

CHAPTER 1: Basics of Python Programming

Programming Exercise

1.

```
# getting input from the user and assigning it to user
```

```
height = float(input("Enter height in meters: "))
```

```
weight = float(input("Enter weight in kg: "))
```

```
# the formula for calculating bmi
```

```
bmi = weight/(height**2)
```

```
# ** is the power of operator i.e height*height in this case
```

```
print("Your BMI is: {0} and you are: ".format(bmi), end="")
```

```
#conditions
```

```
if ( bmi < 16):
```

```
    print("severely underweight")
```

```
elif ( bmi >= 16 and bmi < 18.5):
```

```
    print("underweight")
```

```
elif ( bmi >= 18.5 and bmi < 25):
```

```
    print("Healthy")
```

```
elif ( bmi >= 25 and bmi < 30):
```

```
    print("overweight")
```

```
elif ( bmi >=30):
```

```
    print("severely overweight")
```

2.

```
print("Missile Man of India" + " - Sir APJ Abdul Kalam")
```

3. Refer Section 6.9.2

4.

```
char = input('Enter a character : ')  
print('ASCII value of character : %s' % ord(char))
```

5.

```
char = input('Enter alphabet in upper case : ')  
print('Letter in lower case : %c' % char.lower())
```

6.

```
num1 = int(input('Enter first number : '))  
num2 = int(input('Enter second number : '))  
print('Before swapping : num1 : %d, num2 : %d'%(num1,num2))  
temp = num1  
num1 = num2  
num2 = temp  
print('After swapping : num1 : %d, num2 : %d' %(num1,num2))
```

7.

```
address = input('Enter address(use comma as seperator) : ')  
for x in address.split(','):  
    print(x)
```

8.

```
P = float(input('Enter principal amount : '))  
R = float(input('Enter rate of interest : '))  
T = float(input('Enter time : '))  
N = int(input('Enter number of times interest is computed annually : '))  
SI = P + (P*R*T)/100  
print('Simple interest : %.2f' % SI)  
CI = P * pow(1 + (R/N),(N*T))  
print('Compound interest : %.2f' % CI)
```

9.

```
x = int(input('Enter value of x : '))
y = int(input('Enter value of y : '))
print('x ^ y : %d' %(pow(x,y)))
```

10.

```
first_name = input('Enter first name : ')
last_name = input('Enter last name : ')
print('Greetings!!! %s %s' %(first_name,last_name))
```

11.

```
basic_pay = float(input('Enter basic salary : '))
HRA = 0.1 * basic_pay
TA = 0.05 * basic_pay
salary = basic_pay + HRA + TA
print('Total Salary : %f' % salary)
```

12.

```
item_name = input('Enter item name : ')
item_quantity = int(input('Enter item quantity : '))
item_price = float(input('Enter item price : '))
print('*****')
print('Item Name \t Item Quantity \t Item Price')
print('\n\t%s \t\t %d \t\t %f' %(item_name,item_quantity,item_price))
print('*****')
print('Total Amount to be paid : %f' %(item_quantity * item_price))
print('*****')
```

13.

```
m = float(input("Enter the mass : "))
c = float(input("Enter the velocity : "))
e = m*c*c
print("Momentum = ",e)
```

14.

```
print('Number of Seconds in One Day = ', (24*60*60))
```

15.

```
name = input("Enter Name : ")
marks = input("Enter Marks : ")
course = input("Enter Course : ")
fees = float(input("Enter Fees : "))
print("DETAILS OF STUDENT ARE \n *****")
print("%s - %s - %s - %d"%(name,marks,course,fees))
```

16.

```
fname = input("Enter First Name : ")
lname = input("Enter Last Name : ")
print("Hello %s %s \n Welcome to Python !"%(fname,lname))
```

17.

```
num = int(input("Enter a number : "))
rem = num%10
if(rem == 0):
    print("The number is divisible by 10")
else:
    print(rem, " should be subtracted or %d must be added to %d to make it completely divisible by 10"%(10-rem, num))
```

18.

```
age = int(input("Enter the age : "))
if(age>=21 and age<=35):
    print("The candidate is eligible")
else:
    print("The candidate is not eligible")
```

19.

```
angle = int(input('Enter angle(in degrees) : '))
if(angle<0):
    print('Invalid value')
```

```
if(angle >= 0 and angle<=90):
```

```
    print('1st quadrant')
```

```
elif(angle > 90 and angle <=180):
```

```
    print('2nd quadrant')
```

```
elif(angle > 180 and angle <=270):
```

```
    print('3rd quadrant')
```

```
else:
```

```
    print('4th quadrant')
```

20.

Same as above

21.

```
length = int(input("Enter the length of the rectangle : "))
breadth = int(input("Enter the breadth of the rectangle : "))
print("***** MENU *****")
print("1. PERIMETER \n 2. AREA \n 3. Diagonal \n *****")
option = int(input("Enter your option : "))
if(option == 1):
    res = 2*(length + breadth)
    print("PERIMETER = ", res)
if(option == 2):
    res = length * breadth
    print("AREA = ", res)
else:
    res = (length * length + breadth * breadth) ** 0.5
    print("DIAGONAL = ",res)
```

22. Same as solution to programming exercise 3

23.

```
grad_stream = input('Enter stream in grad.(Science/Commerce/Arts) : ')
grad = float(input('Enter graduation marks : '))
pg_stream = input('Enter PG stream : ')
if(pg_stream != grad_stream):
    grad-=5
```

```
if(classX > 80 and classXII > 80 and grad > 70):
```

```
    print('You are eligible for PG course.')
```

```
else:
```

```
    print('Sorry, you are not eligible for the PG course')
```

24.

```
float_num = float(input('Enter floating point number : '))
```

```
int_num = int(input('Enter integer number : '))
```

```
if(float_num > 3.14):
```

```
    int_num+=100
```

```
print('Updated integer : %d' %int_num)
```

25.

```
a = int(input('Enter the first number : '))
```

```
b = int(input('Enter the second number : '))
```

```
remainder = 0
```

```
quotient = 0
```

```
if(a>b):
```

```
    remainder = a % b
```

```
    quotient = float(a)/float(b)
```

```
else:
```

```
    remainder = b % a
```

```
    quotient = float(b)/float(a)
```

```
print('Remainder : ',remainder)
```

```
print('Quotient : ',quotient)
```

26. Use simple subtraction and string formatting**31.**

divisor

res = 100/divisor

print(res)

ZeroDivisionError: division by zero

if(divisor==0):

print("Division by zero isnot possible")

else:

res = 100/divisor

print(res)

26.

cd = int(input("Enter the current date : "))

cm = int(input("Enter the current month : "))

cy = int(input("Enter the current year : "))

bd = int(input("Enter the birth date : "))

bm = int(input("Enter the birth month : "))

by = int(input("Enter the birth year : "))

dy = cy - by

dm = cm - bm

dd = cd - bd

if(dm<0):

dm+=12

dd -= 12

print(dd,"/", dm, "/", dy)

27.

```
amt = float(input("Enter the amount to be paid : "))
```

```
if amt >= 30000:
```

```
    dis = 0.3 * amt
```

```
elif amt >= 20000:
```

```
    dis = 0.2 * amt
```

```
elif amt >= 10000:
```

```
    dis = 0.1 * amt
```

```
else:
```

```
    dis = 0.05
```

```
total = amt - dis
```

```
print(total)
```

28.

```
age = int(input("Enter your age : "))
```

```
if age == 15:
```

```
    height = 170
```

```
    weight = 56
```

```
if age == 16:
```

```
    height = 173
```

```
    weight = 60
```

```
if age == 17:
```

```
    height = 175
```

```
    weight = 64
```

```
if age == 18:
```

```
    height = 176
```

```
    weight = 66
```

```
print("your Height should be %d and Weight should be %d"%(height,weight))
```



29.

```
temp = float(input("ENter the body temperature : "))
if(temp==98.7):
    print("Perfect Body Temperature")
elif(temp<97):
    print("Lody Body Temperature")
elif(temp>99):
    print("High Temperature")
else:
    print("Your temperature is OK ")
```

30.

```
num = float(input("Enter a number : "))
if(num<0):
    print("Square Root of Negative Numbers is Not Defined")
else:
    res = num ** 0.5
    print("%.2f"%res)
```

31.

```
a = int(input("Input the first side : "))
b = int(input("Input the second side : "))
c = int(input("Input the third side : "))
if (a+b>c) or (a+c)> b or (b+c)>a:
    print(a, b, c, " are sides of a triangle")
else:
    print(a, b, c, " are not the sides of a triangle")
```

32. Same as above

33.

```
a = int(input("Input the first angle : "))
b = int(input("Input the second angle : "))
c = int(input("Input the third angle : "))
if (a+b+c) == 180:
    if(a == 90 or b == 90 or c == 90):
        print(a, b, c, " are angles of a right angles triangle")
    elif (a > 90 or b > 90 or c > 90):
        print(a, b, c, " are angles of an obtuse triangle")
    elif a < 90 and b < 90 and c < 90:
        print(a, b, c, " are angles of an acute triangle")
```

34.

```
num = int(input("Enter a number : "))
if(num == 0):
    print(num, " is equal to zero")
elif(num > 0):
    print(num, " is positive")
else:
    print(num, " is negative")
```

36. Same as above, use if-elif statements**37.**

```
wages = float(input("Enter the hourly wages : "))
hrs = int(input("Enter the regular hours worked : "))
overtime_hrs = int(input("Enter the overtime hours worked : "))
total = wages * hrs + (1.5 * wages * overtime_hrs)
print(total)
```

38.

```
units = int(input('Enter units consumed : '))
total_cost = 0
if(units <= 150):
    total_cost = units*3
```

```

elif(units > 150 and units<=350):
    remaining_units = units - 150
    total_cost = 150 * 3 + (remaining_units * 3.75) + 100

elif(units > 350 and units <=450):
    remaining_units = units - 350
    total_cost = 150 * 3 + (200 * 3.75) + 4*remaining_units + 350

elif(units > 450 and units < 600):
    remaining_units = units - 450
    total_cost = 150 * 3 + (200 * 3.75) + (4 * 100) + 4.25 * remaining_units + 650

else:
    remaining_units = units - 600
    total_cost = 150 * 3 + (200 * 3.75) + (4 * 100) + (4.25 * 150) + 5*remaining_units + 1000
print('Your bill amount : ',total_cost)

```

38.

```

dec = int(input("Enter an integer: "))
print("The decimal value of",dec,"is:")
print(bin(dec),"in binary.")
print(oct(dec),"in octal.")
print(hex(dec),"in hexadecimal.")

```

39.

```

num = input("Enter an octal number :")
OctalToDecimal(int(num))

def OctalToDecimal(num):
    decimal_value = 0
    base = 1
    while (num):
        last_digit = num % 10

```

```
num = int(num / 10)

decimal_value += last_digit * base

base = base * 8

print("The decimal value is :", decimal_value)
```

40.

```
m = int(input("Enter the value of m : "))
n = int(input("Enter the value of m : "))

i = 1

while i < n:

    if(i % m == 0):

        print(i, end = " ")

    i = i + 1
```

41.

```
hrs = int(input("Enter number of hours : "))
mins = hrs * 60
print(mins)
mins = int(input("Enter number of minutes : "))
hrs = 0
while(mins != 0):

    mins = int(mins / 60)

    hrs = hrs + 1

    print(mins, " ", hrs)

print(hrs)
```

42.

```
sedan_charge = 17
prime_charge = 12
mini_charge = 10
parking_charge = 0
```

```
vehicle_type = input('Enter vehicle type(s/p/m) : ')
```

```
hours = int(input('Enter parking hours : '))
```

```
if(vehicle_type == 's'):
```

```
    parking_charge = hours * sedan_charge
```

```
elif(vehicle_type == 'p'):
```

```
    parking_charge = hours * prime_charge
```

```
else:
```

```
    parking_charge = hours * mini_charge
```

```
print('Parking charge : ',parking_charge)
```

43.

```
sedan_charge = 30
```

```
prime_charge = 20
```

```
mini_charge = 10
```

```
sedan_extended_charge = 17
```

```
prime_extended_charge = 12
```

```
mini_extended_charge = 10
```

```
conveyance_charge = 0
```

```
vehicle_type = input('Enter vehicle type(s/p/m) : ')
```

```
entry_kms = int(input('Enter rading in odometer during entry : '))
```

```
exit_kms = int(input('Enter rading in odometer during exit : '))
```

```
total_kms = exit_kms - entry_kms
```

```
if(total_kms >= 5):
```

```
    diff_kms =total_kms - 5
```

```
    if(vehicle_type == 's'):
```

```
        con_charge = sedan_charge * 5 + diff_kms * sedan_extended_charge
```

```
        print(con_charge)
```

```
elif(vehicle_type == 'p'):
    con_charge = (prime_charge * 5) + (diff_kms * prime_extended_charge)
    print(con_charge)
else:
    con_charge = mini_charge * 5 + diff_kms * mini_extended_charge
print('Total charge : ',con_charge)
```

44.

```
n = int(input("Enter the value of n : "))
s = 0.0
for i in range(1,n+1):
    a = float(i**i)/i
    s = s+a
print("The sum of the series is",s)
```

45.

```
n = int(input("Enter the value of n : "))
s = 0
for i in range(1,n+1):
    a = i**3
    print(a)
```

46. Same as question 25**47.**

```
str = "python programmming"
print(str.upper())

i = 0

length = len(str)

while(i<length):
    if(i==0):
```

```

    print(str[i].lower(), end = "")
elif(str[i-1]==' '):
    print(str[i].lower(), end = "")
else:
    print(str[i].upper(), end = "")
i+=1
print("\n")
i = 0
length = len(str)
while(i<length):
    if(i==0):
        print(str[i].upper(), end = "")
    elif(str[i-1]==' '):
        print(str[i].upper(), end = "")
    else:
        print(str[i], end = "")
    i+=1

```

48.

```

number = 0
count_even = 0
count_odd = 0
sum_even = 0
sum_odd = 0

```

```

while(number!=-1):
    number = int(input('Enter number(-1 to finish) : '))
    if(number==1):
        break
    if(number%2==0):

```



```
        count_even+=1

        sum_even+=number

    else :

        count_odd+=1

        sum_odd+=number


print('Odd count : %d' %count_odd)
print('Even count : %d' %count_even)
print('Sum even : %d' %sum_even)
print('Sum odd : %d' %sum_odd)
print('Average even : %f' %(sum_even/count_even))
print('Average odd : %f' %(sum_odd/count_odd))
```

49.

```
from math import sin,radians
degree = 0
while(degree <= 360):
    print('sin %d is : %r' %(degree,sin(radians(degree))))
    degree+=45
```

50.

```
for i in range(1,100):
    if(i%7!=0 and i%11!=0):
        print(i)
```

51.

```
number = int(input('Enter number : '))
digit_count = 0
while(number!=0):
    digit_count +=1
    number = int(number/10)

print('Number of digits : ',digit_count)
```


52.

```
for i in range(100,0,-4):
    print(i)
```

53.

```
size = int(input('Enter size : '))
for i in range(1, size):
    for j in range(1,size):
        if(i==1 or i==(size-1) or j==1 or j==(size-1)):
            print('*',end= ' ')
        else:
            print(' ',end= ' ')
    print('\n')
```

54.

```
size = int(input('Enter size : '))
for i in range(1, size):
    for j in range(1,size):
        if(i==j):
            print('$',end= ' ')
        elif(i==1 or i==(size-1) or j==1 or j==(size-1)):
            print('*', end= ' ')
        else:
            print(' ', end= ' ')
    print('\n')
```

55.

```
size = int(input('Enter size : '))
for i in range(1, size):
    for j in range(1,size):
        if(i==j or i+j==size):
            print('$',end= ' ')
        elif(i==1 or i==(size-1) or j==1 or j==(size-1)):
            print('*',end= ' ')
        else:
            print(' ',end= ' ')
    print('\n')
```

56.

```
ch = 'y'

largest = 0

while(ch == 'y'):

    num = int(input('Enter number : '))

    if(num > largest):

        largest = num

    ch = input('cont(y/n)..')

print('Largest number is ',largest)
```

57.

```
nterms = int(input("How many terms? "))

# first two terms
n1, n2 = 0, 1
count = 0

# check if the number of terms is valid
if nterms <= 0:
    print("Please enter a positive integer")
elif nterms == 1:
    print("Fibonacci sequence upto",nterms,":")
    print(n1)
else:
    print("Fibonacci sequence:")
    while count < nterms:
        print(n1)
        nth = n1 + n2
        # update values
        n1 = n2
        n2 = nth
        count += 1
```

58.

```
for i in range (40,0,-2):

    print(i)
```

59.

```

n = int(input("Enter the value of n : "))
x = int(input("Enter the value of x : "))
sum = 0
for i in range(1,n):
    sum += (-1**i * x**i)
print(sum)

```

```

def fact(x):

```

```

    f = 1

```

```

    for i in range(1,x):

```

```

        f = f*i

```

```

    return f

```

```

n = int(input("Enter the value of n : "))

```

```

x = int(input("Enter the value of x : "))

```

```

sum = 1

```

```

for i in range(n):

```

```

    sum += ((-1**i * x**i)/fact(i))

```

```

print(sum)

```

60.

a.

```

def pattern(n):

```

```

    k = 2 * n - 2

```

```

    for i in range(0, n):

```

```

        for j in range(0 , k):

```

```

            print(end=" ")

```

```

        k = k - 1

```

```

        for j in range(0 , i + 1 ):

```

```

            print("* ", end="")

```

```

        print("\r")

```

```

    k = n - 2

```

```

    for i in range(n , -1, -1):

```

```

        for j in range(k , 0 , -1):

```

```

            print(end=" ")

```

```

        k = k + 1

```

```
for j in range(0, i + 1):  
    print("* ", end="")  
print("\r")
```

pattern(5)

b.

def pattern(n):

```
    for i in range(0, n):  
        for j in range(0, i + 1):  
            print("* ", end="")  
        print("\r")
```

```
    for i in range(n, 0, -1):  
        for j in range(0, i + 1):  
            print("* ", end="")  
        print("\r")
```

pattern(5)

c.

for i in range(5):

for j in range(5):

if i + j == 2 or i - j == 2 or i + j == 6 or j - i == 2:

print("*", end="")

else:

print(end=" ")

print()

d.

def pattern(n):

for i in range(0, n):

for j in range(0, i + 1):

if(j==0 or j==i):

print("* ", end="")

else:

print(" ", end="")

print("\r")

```

for i in range(n, 0, -1):
    for j in range(0, i + 1):
        if(j==0 or j==i):
            print("* ", end="")
        else:
            print(" ", end="")
    print("\r")
pattern(5)
e.
row = int(input('Enter how many lines? '))

```

```

a = 64
# Generating pattern
for i in range(1,row+1):

    # for space
    for j in range(1, row+1-i):
        print(' ', end='')

    # for increasing pattern
    for j in range(1,i+1):
        print('%c' %(a+j), end='')

    # for decreasing pattern
    for j in range(i-1,0,-1):
        print('%c' %(a+j), end='')

    # Moving to next line
    print()

```

```

f.
row = int(input('Enter how many lines? '))

```

```

# Generating pattern
for i in range(1,row+1):

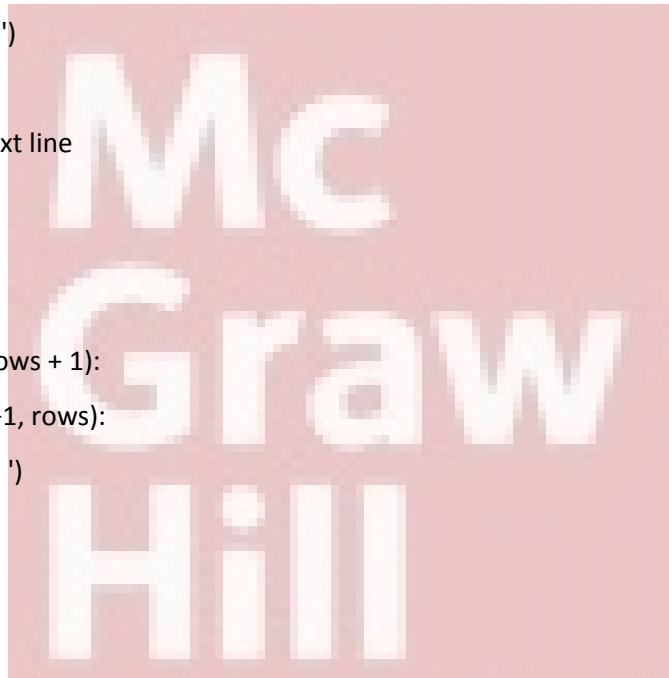
```

```
# for space
for j in range(1, row+1-i):
    print(' ', end='')

# for increasing pattern
for j in range(1,i+1):
    print(j, end='')

#for decreasing pattern
for j in range(i-1,0,-1):
    print(j, end='')

# Moving to next line
print()
g.
rows = 6
for i in range(0, rows + 1):
    for j in range(i+1, rows):
        print(j, end=' ')
    print()
h.
def pattern(n):
    for i in range(1, n):
        for j in range(1, i + 1):
            print(j, end= " ")
        print("\r")
pattern(5)
```



CHAPTER 2: Revisiting Data Structures in Python

1.

```
def find_ch(s, c, pos):
    index = pos
    while(index < len(s)):
        if s[index] == c:
            print(c, "found in string at index : ", index)
            return
        else:
            pass
            index += 1
    print (c, " is not present in the string")
str = input("\n Enter a string : ")
ch = input("\n Enter the character to be searched : ")
pos = int(input("\n Enter the position from which searching needs to be done : "))
find_ch(str, ch, pos)
```

2.

```
def string_length(str1):
    count = 0
    for char in str1:
        count += 1
    return count
str1 = input("\n Enter a string : ")
print(string_length(str1))
```

3.

```
str = input("Enter a string : ")
count_vowels = 0
count_consonants = 0
for char in str:
    if char in "aeiou":
        count_vowels += 1
```

```
else:
    count_consonants += 1
print("Number of consonants = ", count_consonants)
print("Number of vowels = ", count_vowels)
```

4.

```
str = input("Enter a string : ")
count_lower = 0
count_upper = 0
for char in str:
    if char.islower():
        count_lower += 1
    else:
        count_upper += 1
print("Number of lower case characters = ", count_lower)
print("Number of uppercase characters = ", count_upper)
```

5.

```
str = input("Enter a string : ")
count_lower = 0
count_upper = 0
count_space = 0
count_digits = 0
for char in str:
    if char.islower():
        count_lower += 1
    elif char.isupper():
        count_upper += 1
    elif char.isdigit():
        count_digits += 1
    elif char == " ":
        count_space += 1
print("Number of lower case characters = ", count_lower)
print("Number of uppercase characters = ", count_upper)
print("Number of digits = ", count_digits)
print("Number of spaces = ", count_space)
```


6.

```
def string_both_ends(str):
    if len(str) < 2:
        return ""

    return str[0:2] + str[-2:]

print(string_both_ends('Good Morning'))
print(string_both_ends('He'))
```

7.

```
def change_char(str1):
    char = str1[0]
    length = len(str1)
    str1 = str1.replace(char, '$')
    str1 = char + str1[1:]

    return str1

print(change_char('Hello World'))
```

8.

```
def swap(a, b):
    final_a = b[:2] + a[2:]
    final_b = a[:2] + b[2:]

    return final_a + ' ' + final_b

print(swap('abc', 'xyz'))
```

9.

```
def copy(str):
    new_str = ""
    for i in range(len(str)):
        new_str += str[i]
    return new_str

str = input("\n Enter a string : ")
print("\n The copied string is : ", copy(str))
```

10.

```
def modify(str1):  
    length = len(str1)  
  
    if length > 2:  
        if str1[-3:] == 'ing':  
            str1 += 'ly'  
        else:  
            str1 += 'ing'  
  
    return str1  
print(modify('ab'))  
print(modify('abc'))  
print(modify('bring'))
```

11.

```
def not_poor(str1):  
    not = str1.find('not')  
    bad = str1.find('poor')  
  
    if bad > not:  
        str1 = str1.replace(str1[not:(bad+4)], 'good')  
  
    return str1  
print(not_poor('The book is not that poor!'))
```

12.

```
def longest(words_list):  
    word_len = []  
    for n in words_list:  
        word_len.append((len(n), n))  
    word_len.sort()  
    return word_len[-1][1]  
  
print(longest(["Welcome", "to", "Python"]))
```

13.

```
def remove_char(str, n):
    first_part = str[:n]
    last_pasrt = str[n+1:]
    return first_part + last_pasrt
print(remove_char('PythonProgramming', 7))
```

14.

```
def modify(str1):
    return str1[-1:] + str1[1:-1] + str1[:1]

print(modify('PYTHON'))
```

15.

```
def odd_values_string(str):
    result = ""
    for i in range(len(str)):
        if i % 2 == 0:
            result = result + str[i]
    return result

print(odd_values_string('Programming in Python'))
```

16.

```
def word_count(str):
    counts = dict()
    words = str.split()

    for word in words:
        if word in counts:
            counts[word] += 1
        else:
            counts[word] = 1

    return counts

print(word_count('Welcome to the world of Python programming.'))
```

17.

```
items = "Welcome, to, the, world, of, Python, programming "  
words = [word for word in items.split(",")]  
print(", ".join(sorted(list(set(words)))))
```

18.

```
def insert(str, word):  
    return str[:2] + word + str[2:]  
print(insert('---', 'Python'))
```

19.

```
def insert(str):  
    sub_str = str[-2:]  
    return sub_str * 4
```

```
print(insert('Python'))
```

20.

```
def first_three(str):  
    return str[:3] if len(str) > 3 else str
```

```
print(first_three('Hey'))
```

```
print(first_three('Hello World'))
```

21.

```
def first_half(str1):  
    if len(str1) % 2 == 0:  
        pos = int(len(str1)/2)  
        return str1[:pos]
```

```
str1 = "Python"
```

```
print(first_half(str1))
```

22.

```
def reverse(str1):  
    if len(str1) % 4 == 0:  
        return ".join(reversed(str1))  
    return str1
```

```
print(reverse('python'))
```

23.

```
def to_uppercase(str1):
    num_upper = 0
    for letter in str1[:4]:
        if letter.upper() == letter:
            num_upper += 1
    if num_upper >= 2:
        return str1.upper()
    return str1

print(to_uppercase('PyThOnPrOgraMMing'))
```

24. Use the sorted() function

25. Use the rstrip() function

26. Use the startswith() function

27. Same as Q. No. 25, just remove tabs instead of newline characters

28.

```
str1 = "Hello World \n Welcome to Python"
new_str = ""
i = 0
l = len(str1)
while(i<l):
    if(i==0):
        new_str += "NEXT "
    if(str1[i]=='\n'):
        new_str += "\n NEXT"
    i+=1
    new_str += str1[i]
    i+=1
print(new_str)
```

29.

```
str1 = "Hello World \n Welcome to Python"
new_str = ""
i = 0
l = len(str1)
while(i<l):
    if(i==0):
        new_str += "\t"
    if(str1[i]!='\n'):
        new_str += "\n \t"
    i+=1
    new_str += str1[i]
    i+=1
print(new_str)
```

30.

```
x = 3.1415926
print("Formatted Number: "+"{:0.2f}".format(x));
```

31.

```
x = 3.1415926
print("Formatted Number with sign: "+"{:+.2f}".format(x));
```

32.

```
x = 3.1415926
print("Formatted Number with no decimal places: "+"{:0f}".format(x));
```

33.

```
x = 9
print("Formatted Number(left padding, width 2): "+"{:0>2d}".format(x));
```

34.

```
x = 8
print("Formatted Number(right padding, width 2): "+"{:*< 2d}".format(x));
```

35.

```
x = 3000000
print("Formatted Number with comma separator: "+"{:,}".format(x));
```

36.

```
x = 8.2
print("Formatted Number with percentage: "+"{:.2%}".format(x));
```

37.

```
x = 312
print("Left aligned (width 10) :"+"{:< 10d}".format(x));
print("Right aligned (width 10) :"+"{:10d}".format(x));
print("Center aligned (width 10) :"+"{: ^10d}".format(x));
```

38.

```
def strip_chars(str1, chars):
    new_str = ""
    for c in str1:
        if c not in chars:
            new_str+=c
    return new_str

print("\nOriginal String: ")
print("The quick brown fox jumps over the lazy dog.")
print("After stripping a,e,i,o,u")
print(strip_chars("The quick brown fox jumps over the lazy dog.", "aeiou"))
```

39. Find reverse of the string using reverse() and then join it with the original string.

40. Refer to solution 38

41. Compare the original string with its reverse. If equal then the string is a palindrome

42.

```
str1= "Hello There World.. How are yu all?"
str2 = "There"
str3 = ""
str4 = str1.split()
for i in str4:
    if i!="There":
        str3 += i
print(str3)
```

43.

```
str1= "Hello"
str2 = "World"
str3 = ""
for i in str1:
    str3 += i
for i in str2:
    str3 += i
print(str3)
```

44. Same as above

45.

```
str1= "Hello"
str2 = "World"
str3 = ""
str3 = str1
str1 = str2
str2 = str3
print("str1 = ", str1)
print("str2 = ", str2)
```

46.

```
str1= "Welcome to the world of Python"
str2 = "to"
str3 = "TO"
new_str = ""
words = str1.split()
for i in words:
    if(i=='to'):
        new_str += str3
    else:
        new_str += i
print(new_str)
```



47.

```
str1 = "Hello World Welcome to Python"
str2 = "World"
print(str1.replace(str2, ""))
```

48.

```
str1 = "Hello World Welcome to Python"
str2 = "World"
str3 = "Dude"
```

```
i = 0
```

```
l = len(str1)
```

```
while(i<l):
```

```
    if(str1[i] == str2[0]):
```

```
        l2 = len(str2)
```

```
        k = i+1
```

```
        j = 1
```

```
        while(j<l2):
```

```
            if(str1[k] == str2[j]):
```

```
                k +=1
```

```
                j+=1
```

```
            if(j == l2):
```

```
                print(str1.replace(str2, str3))
```

```
                break
```

```
        i += 1
```

49. Use the strip() function

50.

```
str1= "Janak Raj Thareja"
i = 0
l = len(str1)
new_str = str1[0]
```

```
while(i<l):
    if(str1[i] == " "):
        new_str += str1[i+1]
        i+=1
print(new_str.upper())
```

51.

```
str1= "Janak Raj Thareja"
i = 0
l = len(str1)
new_str = str1[0]+"."
while(i<l):
    if(str1[i] == " " and str1.find(" ", i+1, l) != -1):
        new_str += str1[i+1] + "."
    if(str1.find(" ", i+1, l) == -1):
        while(i<l):
            new_str += str1[i]
            i+=1
        i+=1
print(new_str.upper())
```

52. Use the count() function

53. Use the count() function

54.

```
def copy(str,n):
    new_str = ""
    for i in range(n+1):
        new_str += str[i]
    print("\n The copied string is : ", new_str)
str = input("\n Enter a string : ")
n = int(input("Enter number of characters to be copied : "))
copy(str,n)
```

55.

```

n = int(input("Enter n : "))
m = int(input("Enter m : "))
str1 = "Welcome to the world of programming in Python"
i = 0
new_str = ""
l = len(str1)
while(i<n):
    new_str += str1[m]
    m+=1
    i+=1
print(new_str)

```

56.

```

str1 = "Welcome to the world of programming in Python"
i = 0
new_str = ""
l = len(str1)-1
while(i<l):
    new_str += str1[i]
    i+=1
print(new_str)

```

57.

```

str1 = "Welcome to the world of programming in Python"
i = 1
new_str = ""
l = len(str1)
while(i<l):
    new_str += str1[i]
    i+=1
print(new_str)

```

58.

```
str1 = "Welcome to the world of programming in Python"
str1 = str1.upper()
i = 0
new_str = ""
l = len(str1)
while(i<l):
    new_str += chr((ord(str1[i])+ 3))
    i+=1
print(new_str)
```

59. Same as Q.No.54. Just multiply key instead of adding

60.

```
str1 = "hello world"

i = 0
new_str = ""
l = len(str1)
while(i<l):
    if(i==0):
        new_str += str1[i].upper()
    else:
        new_str += str1[i]
    i+=1
print(new_str)
```



61.

```
str1 = "hello world"
i = 0
new_str = ""
l = len(str1)
while(i<l):
    if(i==0):
        new_str += str1[i].upper()
    elif(str1[i-1] == " "):
        new_str += str1[i].upper()
```

```

else:
    new_str += str1[i]
    i+=1
print(new_str)

```

62.

```
str1 = " hElLo WoRlD""
```

```

i = 0
new_str = ""
l = len(str1)
while(i<l):
    if(str1[i].isupper()==True):
        new_str += str1[i].lower()
    else:
        new_str += str1[i].upper()
    i+=1
print(new_str)

```

63.

```

str1 = "hello world"
i = 0
l = len(str1)
new_str = ""
while(i<l):
    if(i==0):
        new_str += str1[0].upper()
    else:
        new_str += str1[i]
    i+=1
print(new_str.replace("world", "friends"))

```

64.

```

no = input("Enter 10 digit phone number : ")
new_no = ""
for i in range(10):

```

```
    if i==3 or i == 6:
        new_no += "-"
    new_no += no[i]

print(new_no)
```

65.

```
i=0
while i<5:
    print(i)
    str = input("Enter a word : ")
    if len(str) < 8:
        print("Enter at least 8 characters")
        continue
    if '0' not in str:
        print("Enter at least one zero")
        continue
    else:
        print(str)
        i += 1
```

66.

```
n = int(input("Enter the value of n : "))
l = []
for i in range(n+1):
    term = 2 ** (i-1)
    l.append(term)
for i in range(n+1):
    if i%2 == 0:
        print(l[i])
```

67.

```
l = [1,2,3,4,5,6,7,8,9,10,9,8,7,6,5,4,3,2,1]
j = len(l)-1
for i in range(len(l)):
    print(l[i],i,j)
    j -= 1
```

68.

```

l1 = []
l2 = []
print("Enter the elements of first list... Enter -123 to exit")
while(1):
    n = int(input())
    if(n== -123):
        break
    l1.append(n)
print("Enter the elements of second list... Enter -123 to exit")
while(1):
    n = int(input())
    if(n== -123):
        break
    l2.append(n)
for i in l1:
    if i in l2:
        print(i)

```

69.

```

list1 = []
str1 = ' abcdefghijklmnopqrstuvwxyz'
length = len(str1)
for i in range(1,length):
    c = (str1[i])*i
    list1.append(c)
print(list1)

```

70.

```

l = []
num = 2
den = 9
n = int(input("Enter the value of n :"))
for i in range(n+1):
    if i%2 == 0:
        term = str(num) + "/" + str(den)

```

```
else:
    term = "-" + str(num) + "/" + str(den)
l.append(term)
num += 3
den += 4

print(l)
```

b.

```
def fact(x):
    res = 1
    for i in range(1,x+1):
        res = res * i

    return res

l = []
n = int(input("Enter the value of n :"))
x = int(input("Enter the value of x :"))
for i in range(1,n+1):
    f = fact(i)
    print(f)
    if i%2 == 0:
        term = (x ** i)/f
    else:
        term = (-x ** i)/f
    l.append(term)
print(l)
```

71.

```
import random
List = []
for i in range(5):
    List.append(int(random.randint(0,100)))
print(List)
```


72.

```
l1 = [1,2,3,4,5]
l2 = [3,5,7,9,2]

for i in range(len(l1)):
    s = 0
    s = l1[i] + l2[i]
    print(s)
```

73.

```
a. List = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
print(List[:3])
b. print(List[2:3])
c.
i = int(input("Enter any index value : "))
print(List[i:])
```

74.

```
import random
List = []
for i in range(20):
    List.append(int(random.randint(0,100)))
print(List)
l = int(len(List)/2)
print(l)
s = 0
for i in List[l:]:
    s += i
print(s)
```

75.

```
import random
List = []
for i in range(20):
    List.append(int(random.randint(0,100)))
print(List)
l = len(List)
```

```
print(l)
s = 0
for i in List:
    s += i
print(s)
```

76.

```
import random
List = []
for i in range(20):
    List.append(int(random.randint(0,100)))
print(List)
l = len(List)
print(l)
s = 0
for i in List:
    if i%2 == 0:
        s += i
print(s)
```

77.

```
List = [12,98,56,45,22,34,90,65]
i=0
l = len(List)
j = l-1
while(i<j):
    temp = List[i]
    List[i] = List[j]
    List[j] = temp
    i+=1
    j-=1
print(List)
```

78.

use in and not in operator

79.

```
List = ['ant', 'bat', 'cat', 'apple', 'bug', 'cake']
c = input("Enter a character (a/b/c) : ")
for word in List:
    if(word.startswith(c)):
        print(word)
```

80.

```
List = []
for i in range(5):
    num = int(input("Enter a number : "))
    if(num>100):
        List.append("EXCESS")
    else:
        List.append(num)
print(List)
```

81.

```
List = [1,2,3,4,1,2,3,4]
i = 0
l = len(List)-1
val = int(input("Enter a number : "))
count = 0
while(i<l):
    if (List[i] == val):
        count+=1
    i+=1
print(count)
```

82.

```
List = [1,2,3,4,6,7,8]
i = 0
l = len(List)
val = int(input("Enter a number : "))
pos = int(input("Enter the position : "))
j = pos
List.append(List[l-1])
```

```
while(j<=l):  
    List[l] = List[l-1]  
    l-=1  
List[pos-1] = val  
print(List)
```

83.

```
List = [12,98,56,45,22,34,90,65]  
print(List)  
for i in List:  
    if i%3==0 or i%6 == 0:  
        print(i)
```

84.

```
List = [12,98,56,45,22,34,90,65]  
print(List)  
for i in List:  
    if i%3==0 or i%6 == 0:  
        print(i)
```

85.

```
import random  
i = random.randrange(1,10)  
List = [12,98,56,45,22,34,90,65]  
print(List[i])
```

86.

```
str = "HELLOWORLD"  
l = list(str)  
k = 0  
for i in range(1,len(l)+1):  
    print()  
    for j in range(1,len(l)):  
        if j == i+1 and j!=len :  
            print(l[k], end = "")  
            k += 1  
        elif j == len(l)-i:  
            print(l[k], end = "")  
            k += 1  
    else:  
        print(" ", end = "")
```

87.

```
List1 = ['a','b','c']
Tup1 = tuple(List1)
print(Tup1)
Tup2 = ('a', 'b','c')
List2 = list(Tup2)
print(List2)
```

88.

```
Tup = (('a','b','c'),)
print(len(Tup))
```

89.

```
List = ['ant', 'bat', 'cat', 'apple', 'bug', 'cake']
c = input("Enter a character (a/b/c) : ")
for word in List:
    if(word.startswith(c)):
        print(word)
```

90.

```
T=((8,9,10),(9,9,9),(6,4,8),(10,10,10),(7,6,8))
total_sum = 0
print(T)
print(T[3])
for i in T:
    print(i[0], end = " ")
print()
for i in T:
    sum = 0
    for j in i:
        sum += j
    avg = sum/3
    total_sum += sum
    print(sum, avg)
print(total_sum, total_sum/15)
for i in T:
    print("MAX ", max(i), "MIN = ", min(i))
```

91.

```
T1 = (1,2,3)
T2 = (4,5,6,7,8)
T1 = T1 + T2
print(T1)
```

92.

```
a.
Tup = ()
L = []
for i in range(1,11):
    L.append(i**2)
Tup = list(L)
print(Tup)

b.
T = ()
for i in range (1,21,2):
    T += (i**3,)
print(T)

c.
T = ()
for i in range (1,11):
    T += (i*7,)
print(T)

d.
T = ()
for i in range(-5,11,2):
    T += (i,)
print(T)
```

93.

```
import random
T = ()
for i in range(10):
    n = random.randint(1,100)
    p = random.randint(1,100)
    T += ((n,p),)
```

```

print(T)
count = 0
for i in T:
    if i[0]%2 == 1 and i[1]%2 == 1:
        count += 1
print(count)

```

94.

```

import random
T1 = ()
T2 = (3,4,6)
for i in range(20):
    n = random.randint(1,10)
    T1 += (n,)
print(T1)
count = 0
l = len(T2)
for i in T2:
    if i in T1:
        count += 1
if count == l:
    print("SUBSET")
else:
    print("NOT A SUBSET")

```

95.

```

Products = {'Pen Drive': 500, 'Mouse':400, 'Keyboard': 600}
sum = 0
for val in Products.values():
    sum += val
print(sum)

Tup = ()
L = []
for i in range(1,11):
    L.append(i**2)
Tup = list(L)
print(Tup)

```

96.

```
Dict={0:0, 1:1}
def fibo(n):
    if n not in Dict:
        val=fibo(n-1)+fibo(n-2)
        Dict[n]=val
    return Dict[n]
n=int(input("Enter the value of n:"))
print("Fibonacci(", n,")= ", fibo(n))
```

97.

```
msg = 'Hello All, Good Morning... Welcome to the World of Python'
msg = msg.lower()
Dict = dict()
for word in msg:
    if word not in Dict:
        Dict[word] = 1
    else:
        Dict[word] = Dict[word] + 1
print(Dict)
for key, val in Dict.items():
    print(key, '\t', '*' * val)
```

98.

```
matrix = [[0,0,0,1,0],
[2,0,0,0,3],
[0,0,0,4,0]]
Dict = {}
print("Sparse Matrix")
for i in range(len(matrix)):
    print("\n")
    for j in range(len(matrix[i])):
        print(matrix[i][j], end = ' ')
        if matrix[i][j]!=0:
            Dict[(i,j)] = matrix[i][j]
    print("\n\nSparse Matrix can be efficiently represented as Dictionary : ")
print(Dict)
```


98.

```

Bdays = {'Arav' : '17/3', 'Manan' : '26/2', 'Pratham' : '5/6'}
print(sorted(Bdays.keys()))
name = input("Enter the name you are looking for : ")
if(name in Bdays):
    print(Bdays[name])
else:
    bday = input("Enter birth date : ")
    Bdays[name] = bday
    print(Bdays)

```

99. Refer to the above solution

100. Same as 98

101.

```

Marks = {'CSA' : 90, 'DS' : 92, 'FOC' : 91, 'C++' : 94, 'C' : 88}
max = -1
min = 111
for val in Marks.values():
    if(val>max):
        max = val
    elif(val<min):
        min = val
print("MAX = ", max)
print("MIN = ", min)

```

102. Duplicate keys are not allowed in dictionary. For values, check the count. If value has count greater than 1 delete the key.

103.

```

Dict = {}
if(len(Dict)==0):
    print("EMPTY")

```

104.

```
Emp = {'Shiv' : {'Desig':'Manager', 'Sal': 100000, 'Deptt': 'IT'},  
      'Sadhvi' : {'Desig':'Director', 'Sal': 200000, 'Deptt': 'SALES'},  
      'Krish' : {'Desig':'Vice President', 'Sal': 300000, 'Deptt': 'MARKETING'}}  
for key, val in Emp.items():  
    print(key, val)
```

105.

```
D1 = {'a': 10, 'b': 20, 'c': 10}  
count = {}  
val = D1.values()  
for i in val:  
    if i not in count:  
        count[i] = 0  
        count[i] += 1  
  
a = False # This is used to check if there is atleast one value which occurs more than once  
for i in count.values():  
    if i>1:  
        a = True  
print(i, "keys have same values")  
if not a: # There are no two keys with same value  
    print('No keys have same values')
```

106.

```
D1 = {'a': 10, 'b': 20, 'c': 10}  
  
count = {}  
val = D1.values()  
for i in val:  
    if i not in count:  
        count[i] = 0  
        count[i] += 1  
  
a = False # This is used to check if there is atleast one value which occurs more than once  
for i in count.values():
```

```

if i==1:
    a = True
print(i, "keys have a different value")
if not a: # There are no two keys with same value
    print('All keys have same value')

```

107.

```

dict1= {'key1':'Geeks', 'key12':'For'}
dict2= {'key1':'Geeks', 'key2':'For', 'key3':'geeks',
        'key4': {'GeekKey1': 12, 'GeekKey2': 22, 'GeekKey3': 32 }}

```

```

for key in dict1.keys():

```

```

    if not key in dict2:

```

```

        # Printing difference in
        # keys in two dictionary
        print(key)

```

108.

```

Marks = {'Neha' : 97, 'Mitul' : 92, 'Shefali' : 67, 'Radhika' :85, 'Tanush' : 100}

```

```

print(Marks['Shefali'])

```

```

for key,val in Marks.items():

```

```

    if val>90:

```

```

        print(key, end = " ")

```

```

print()

```

```

sort_keys = sorted(Marks.keys())

```

```

for i in sort_keys:

```

```

    print(i, " = ",Marks[i])

```

```

m = []

```

```

for i in Marks.values():

```

```

    m.append(i)

```

```

m = sorted(m)

```

```

print(m)

```

```

for i in m:

```

```

    for key,val in Marks.items():

```

```

        if val == i:

```

```

            print(key, " = ", i)

```

```
names = []
for i in Marks.keys():
    names.append(i)
print(names)
marks = []
for i in Marks.values():
    marks.append(i)
print(marks)
```

109. Same a Solution to Question 1

110.

```
tup = ((1,2),(3,4),(5,6),(7,8),(9,10))
Dict = dict(tup)
print(Dict)
```



CHAPTER 3: Functions and Modules

1.

```
def largest(a,b,c):  
    if a>b and a>c:  
        return a  
    if b>a and b>c:  
        return b  
    if c>b and c>a:  
        return c  
a,b,c = input("Enter three numbers : ").split()  
large = largest(a,b,c)  
print(large)
```

2.

```
import time  
start = time.time()  
#do stuff  
print(start - time.time())
```

3.

```
num = int(input("Enter a number : "))  
print(abs(num))
```

4.

```
def is_prime(x):  
    if x==1:  
        print("Neither Prime Nor Composie")  
        exit(0)  
    elif x == 2 or x==3:  
        return(0)  
    else:  
        i=2  
        while(i<x):  
            flag = 0
```

```
        if x%i == 0:
            flag = 1
            break
        i = i+1
    return(flag)
num = int(input("Enter a number : "))
flag = is_prime(num)
if flag == 0:
    print("Prime")
else:
    print("Composite")
```

5.

```
def month(x):
    if(x==1):
        print("JANUARY")
    if(x==2):
        print("FEBRUARY")
    if(x==3):
        print("MARCH")
    if(x==4):
        print("APRIL")
    if(x==5):
        print("MAY")
    if(x==6):
        print("JUNE")
    if(x==7):
        print("JULY")
    if(x==8):
        print("AUGUST")
    if(x==9):
        print("SEPTEMBER")
    if(x==10):
        print("OCTOBER")
    if(x==11):
        print("NOVEMBER")
```



```

if(x==12):
    print("DECEMBER")
if(x<1 and x>12):
    print("INVALID NUMBER")
num = int(input("Enter an integer (1-12) : "))
print("The corresponding month is ", end=" ")
month(num)

```

6.

```

def check(year):
    if((year%4==0 and year %100!=0) or (year%400 == 0)):
        print("Leap Year")
    else:
        print("Not a Leap Year")

year = int(input("Enter any year : "))
check(year)

```

7.

```

str1 = "Hello"
str2 = "World"
def swap():
    global str1, str2
    temp = str1
    str1 = str2
    str2 = temp
    print("AFTER SWAPPING STR1 = ", str1)
    print("AFTER SWAPPING STR2 = ", str2)

print("BEFORE SWAPPING STR1 = ", str1)
print("BEFORE SWAPPING STR2 = ", str2)
swap()

```

8.

```

def add(x,y):
    return(x+y)

```

```
def sub(x,y):
    return(x-y)
def mul(x,y):
    return(x*y)
def div(x,y):
    return(x/y)
a = int(input("Enter the first number : "))
b = int(input("Enter the second number : "))
while(1):
    print("***** MENU *****")
    print("1. ADD \n 2. SUBTRACT \n 3. MULTIPLY \n 4. DIVIDE")
    option = int(input("Enter your option : "))
    if(option ==1):
        res = add(a,b)
        print(res)
    elif option ==2:
        res = sub(a,b)
        print(res)
    elif option == 3:
        res = mul(a,b)
        print(res)
    elif option == 4:
        res = div(a,b)
        print(res)
    else:
        break
```

9.

```
def hyp(a,b):
    return ((a * a) + (b * b))

a = int(input("Enter the first side : "))
b = int(input("Enter the second side : "))
print(hyp(a,b))
```


10.

```
def average():
    ch = 'y'
    sum = 0
    n = 0
    while(1):
        num = int(input("Enter a number : "))
        sum += num
        n += 1
        ch = input("Do you want to enter another number (y/n) : ")
        if(ch == 'n' or ch=='N'):
            break
    avg = sum/n
    return avg
print(average())
```

11.

```
def area(x,y):
    return 0.5 * x * y
a = int(input("Enter the height of the triangle : "))
b = int(input("Enter the base of the triangle : "))
print("AREA OF RIGHT ANGLED TRIANGLE = ", area(a,b))
```

12.

```
def power(x,y):
    res = 1

    while(y>=1):
        res = res * x
        y -= 1
    print(res)
a = int(input("Enter the first number : "))
b = int(input("Enter the second number : "))
print(a, " ** ", b, " = ", end = "")
power(a,b)
```

13.

```
def compound_interest(principle, rate, time):  
    # Calculates compound interest  
    Amount = principle * (pow((1 + rate / 100), time))  
    CI = Amount - principle  
    print("Compound interest is", CI)  
  
compound_interest(10000, 10.25, 5)
```

14.

```
def convert(c):  
    print("%.2f"%(1.8*c+32))  
c = int(input("Enter the temperature in Celsius : "))  
print("Temperature in Fahrenheit = ", end = "")  
convert(c)
```

15.

```
n = int(input('Enter N:'))  
for i in range(n+1):  
    for j in range(n+1):  
        if i == 0:  
            print('*', end = ' ')  
        elif i == n:  
            print('*', end = ' ')  
        elif j == 0:  
            print('\n!', end = ' ')  
        elif j == n:  
            print('!')  
        else:  
            print(' ', end = ' ')
```

16.

```
def printStatus(ch):  
    ''' This is a docstring.  
    The function prints status of a relationship'''  
    if ch == 'S' or ch == 's':  
        print("Separated")
```

```

elif ch == 'M' or ch == 'm':
    print("Married")
elif ch == 'D' or ch == 'd':
    print("Divorced")
elif ch == 'U' or ch == 'u':
    print("Unmarried")
else:
    print("Undefined Status")
ch = input("Enter the status (S,U,M,D) : ")
printStatus(ch)

```

17.

```

def check(x,y,z):
    if(x==0 or y==0 or z==0):
        return 1
    else:
        return 0
a = int(input("Enter a number : "))
b = int(input("Enter a number : "))
c = int(input("Enter a number : "))
print(bool(check(a,b,c)))

```

18.

```

def div_num(m,n):
    i = 1
    while(i<n):
        if(i%m==0):
            print(i, end = " ")
        i+=1

m = int(input("Enter m : "))
n = int(input("Enter n : "))
div_num(m,n)

```



19.

```
def mesg(name):  
    print("Hello, ", name)  
name = input("Enter your name : ")  
mesg(name)
```

20.

```
import math  
# prime  
def primeFactors(n):  
    # no of even divisibility  
    while n % 2 == 0:  
        print (2),  
        n = n / 2  
    # n reduces to become odd  
    for i in range(3,int(math.sqrt(n))+1,2):  
        # while i divides n  
        while n % i== 0:  
            print (i)  
            n = n / i  
    # if n is a prime  
    if n > 2:  
        print (n)  
n = 200  
primeFactors(n)
```

21.

```
def Range(list1):  
    largest = list1[0]  
    lowest = list1[0]  
    largest2 = None  
    lowest2 = None  
    for item in list1[1:]:  
        if item < lowest:  
            lowest2 = lowest  
            lowest = item
```

```
elif lowest2 == None or lowest2 > item:
    lowest2 = item

print("Smallest element is:", lowest)
print("Second Smallest element is:", lowest2)

# Driver Code
list1 = [12, 45, 2, 41, 31, 10, 8, 6, 4]
Range(list1)
```



CHAPTER 4: Data Handling Using Numpy

1.

```
import numpy as np
import random
temps = []
for i in range(11):
    val = random.randint(30,45)
    temps.append(val)
arr = np.array(temps)
print(arr)
```

2.

```
import numpy as np
arr = np.arange(3,100,3, dtype=None)
print(arr)
```

3.

```
import numpy as np
arr = np.arange(100,105,0.5, dtype=None)
print(arr)
```

4.

```
import numpy as np
a = np.full((4,3), 3.9)
print(a)
```

5.

```
import numpy as np
a = np.random.rand(5,5)
print(a)
```

6.

```
import numpy as np
a = np.random.rand(5,5)
print(a)
```

Copyright © 2023 by McGraw Hill Education (India) Private Limited

```

print("Number of Dimensions in a : ",np.ndim(a))
print("Shape of a : ",np.shape(a))
print("Dtype of a : ",a.itemsize)
print("Size of a : ",np.size(a))
print("Size of each item : ",np.dtype(float).itemsize)
print("First 4 Values : ", a.head(4))

```

7.

```

import numpy as np

indi_List =
np.array([[ 'Shyna','Krishnav','Chinmay','Ashnoor','Harpreet','Jiya'],[ 'Ali','Peter','Sukhpreet','Dhru
v','Dev','Sarthak'],[ 'John','Mithila','Karan','Vipul','Manan','Urvi']])

print("Individual List of Students is : ", indi_List)

OneD_arr = indi_List.reshape([1,18])
print("OneD Array is : ", OneD_arr)

```

8.

```

import numpy as np

Marks = np.array([[78,80,82,75],[67,65,70,72],[99,98,100,89],[77,80,85,67],[88,79,85,90]])

max_sub = Marks.max(axis=0)
print(max_sub)

min_stud = Marks.min(axis=1)
print(min_stud)

newMarks = np.array([87,67,88,90])
UpdatedMarks = np.vstack([Marks,newMarks])
print(UpdatedMarks)

newMarks = np.array([75,87,98,90,66,78])
UpdatedColsMarks = np.column_stack([UpdatedMarks,newMarks])
print(UpdatedColsMarks)

del_row_Marks = np.delete(UpdatedColsMarks, 3, 0) #Delete row 3
print(del_row_Marks)

del_col_Marks = np.delete(UpdatedColsMarks, 2, 1) #Delete row 3
print(del_col_Marks)

```

9.

```
import numpy as np
arr3D = np.arange(36).reshape(3,4,3)
print("3D ARRAY IS : \n", arr3D)
print()
print()
for value in range(arr3D.shape[0]):
    if value %2 == 0:
        print(arr3D[value,:])
print()
print()
for value in range(arr3D.shape[1]):
    if value %2 == 1:
        print(arr3D[:,value])
```

10.

```
import numpy as np
arr2D = np.arange(12).reshape(3,4)
print("2D ARRAY IS : \n", arr2D)
print()
print()
print(arr2D[:,1])
```

11.

```
import numpy as np

arr = np.array((1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12))
print(arr)
tup = tuple(arr)
print(tup)
```

12.

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

newarr = arr.reshape(2, 3, 2)

print(newarr)
new_arr = newarr.reshape(-1)
print(new_arr)
```


13.

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

for i in arr:
    if i%2 == 0:
        print(i**3)
```

14.

```
import numpy as np
arr2D = np.arange(12).reshape(3,4)
print("2D ARRAY IS : \n", arr2D)
result = arr2D.sum(axis=0)
print("\nSum of all columns:")
print(result)
j=0
for i in result:
    if i%5 == 0:
        print("Column = ",j," with sum = ",i)
    j+=1
```

15.

```
import numpy as np
arr3D = np.arange(24).reshape(2,4,3)
print("3D ARRAY IS : \n", arr3D)
result = arr3D.sum(axis=1)
print("\nSum of all columns:")
print(result)
```

16.

```
import numpy as np
arr = np.arange(12)
arr2D = arr.reshape(4,3)
print("2D ARRAY IS : \n", arr2D)
```

17.

```
import numpy as np
arr = np.arange(24)
arr2D = arr.reshape(6,4)
print("2D ARRAY IS : \n", arr2D)

arr3D = arr.reshape(2,4,3)
print("3D ARRAY IS : \n", arr3D)

newarr = arr3D.reshape(-1, arr3D.shape[-1])
print("\n \n",newarr)
```



CHAPTER 5: Python Pandas

- a. sort by the columns of 'Year' and 'Brand'
- b. Find out mean value of 'A' of each column 'C'
- c. an unsorted data frame
- d. sorting can be done on the column labels.
- e. col1 values are sorted and the respective col2 value and row index will alter along with col1.
- f. Sort data frame by using mergesort algorithm
- g. Descending Order of index
- h. Get the summary of the balance variable by using .describe()
Get relevant percentiles and see their distribution.
Get the summary of the age variable
- i. Prints quantiles of data in the json file
- j. Creates a data frame filled with random values. Find mean of each column by using apply() function
- k. Rename column names of a data frame.

1.

```
import pandas as pd

Student = {'RollNo':[1,2,3,4,5,6],
           'Name':['Chitra','Mansha','Priyal','Chinmay','Vicky','Mansha'],
           'Marks':[89,90,97,93,86,85],
           'Class':['X','XII','IX','VII','VIII','VI']}

df = pd.DataFrame(Student)

print(df)

df.sort_values(by=['Name','Marks'], ascending = False, inplace = True)

print(df)
```

2.

```
df1 = df.pivot(index='Class',columns = 'Name',values = 'Marks')
print(df1)
```

3.

```
import pandas as pd
import numpy as np
df = pd.DataFrame({'a':[1,2,np.nan], 'b':[np.nan,1,np.nan]})
print(df.isna().sum())
```

4.

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack',
    'Lee','David','Gasper','Betina','Andres']),
    'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
    'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])
}

#Create a DataFrame
df = pd.DataFrame(d)
print(df)

print(df.sum())

print(df.sum(1))

print(df.mean())

print(df.std())

print(df.describe())
```

5.

```

import pandas as pd
import numpy as np

# create a dictionary with five fields each
df = pd.DataFrame(10*np.random.randn(5,3),columns=['A','B','C'])

# Convert the dictionary into DataFrame

print("Original DataFrame:\n", df)

# applying function to each row in the dataframe and storing result in a new column
df['add'] = df.apply(np.sum, axis = 1)

print('\nAfter Applying Function: ')
# printing the new dataframe
print(df)

```

6.

```

import pandas as pd

scores = {"Physics":(37,45,45,46,52,55,55,55,61,75),
          "Chemistry":(45,50,52,52,57,60,67,72,76,78)};

dataFrame = pd.DataFrame(data=scores);
print("Data Frame:");
print(dataFrame);

# Compute the 100th percentile
quantileValues = dataFrame.quantile(1);
print("100th percentile:");
print(quantileValues);

# Compute the 95th percentile
quantileValues = dataFrame.quantile(.95);

```

```
print("95th percentile:");  
print(quantileValues);  
  
# Compute the 50th percentile  
quantileValues = dataframe.quantile(.50);  
print("50th percentile:");  
print(quantileValues);
```

7.

```
import pandas as pd  
df = pd.read_csv("titanic.csv")  
print(df.tail(25))  
print(df.describe())  
print("Sorted Data Frame")  
print(df.sort_values("Fare", ascending = False).head(10))  
print(df.sort_values("Cabin", ascending = True, na_position = 'last'))  
print(df["Sex"].value_counts())  
print(df["Embarked"].nunique())  
embarked_c_mask = df["Embarked"] == "C"  
print(df[embarked_c_mask])  
  
df_fare_mask = df["Fare"] < 100  
df_sex_mask = df["Sex"] == "female"  
print(df[df_fare_mask & df_sex_mask])  
  
df_fare_mask2 = df["Fare"] > 500  
df_age_mask = df["Age"] > 70  
print(df[df_fare_mask2 | df_age_mask])  
  
null_mask = df["Cabin"].isnull()  
print(df[null_mask])  
  
print(df.isnull().sum())
```

```
df = df.drop(labels = ["Cabin"], axis=1).head()
print(df)
```

```
df['Age'] = df['Age'].fillna((df['Age'].median()))
print(df)
```

8.

```
df["User_Mean"] = df.groupby('User_ID')['Purchase'].transform('mean')
```

9.

```
# importing pandas as pd
```

```
import pandas as pd
```

```
# Creating the dataframe
```

```
df = pd.DataFrame({"A": [1, 5, 3, 4, 2],
                  "B": [3, 2, 4, 3, 4],
                  "C": [2, 2, 7, 3, 4],
                  "D": [4, 3, 6, 12, 7]},
                  index=["first", "second", "third", "fourth", "fifth"])
```

```
# Print the dataframe
```

```
df
```

```
# reindexing with new index values
```

```
df.reindex(["first", "dues", "trois", "fourth", "fifth"])
```

```
# filling the missing values by 100
```

```
df.reindex(["first", "dues", "trois", "fourth", "fifth"], fill_value = 100)
```

```
df.reindex(columns=["A", "B", "D", "E"])
```

```
df.reindex(columns=["A", "B", "D", "E"], fill_value = 25)
```

10.

```
# import pandas
```

```
import pandas as pd
```

```
# link to gapminder data
```

```
data_url = 'http://bit.ly/2cLzoxH'
```

```
# read data from url as pandas dataframe
```

```
gapminder = pd.read_csv(data_url)
```

#let us check the names of the columns of the dataframe, the first three rows of the data, using head function.

```
print(gapminder.head(3))
```

#To change the columns of gapminder dataframe, we can assign the list of new column names to gapminder.columns as

```
gapminder.columns = ['country','year','population',  
                    'continent','life_exp','gdp_per_cap']
```

#This will assign the names in the list as column names for the data frame “gapminder”. We can check the dataframe to see that if it has new column names using head() function.

```
gapminder.head(3)
```

```
gapminder.rename(columns={'pop':'population',  
                        'lifeExp':'life_exp',  
                        'gdpPercap':'gdp_per_cap'},  
                inplace=True)
```

```
print(gapminder.columns)
```

```
print(gapminder.head(3))
```

11.

```
import pandas as pd
```

```
# Create a dataframe
```

```
data = {'country': ['Cochice', 'Pima', 'Santa Cruz', 'Maricopa', 'Yuma'],  
        'year': [2012, 2012, 2013, 2014, 2014],  
        'reports': [4, 24, 31, 2, 3]}
```

```
df = pd.DataFrame(data)
```

```
print(df)
```

```
# Change the order (the index) of the rows
```

```
df.reindex([4, 3, 2, 1, 0])
```

```
# Change the order (the index) of the columns
```



```
columnsTitles = ['year', 'reports', 'county']
df = df.reindex(columns=columnsTitles)
print(df)
```

12.

```
# Create a function that
def mean_age_by_group(dataframe, col):
    # groups the data by a column and returns the mean age per group
    return dataframe.groupby(col).mean()

# Create a function that
def uppercase_column_name(dataframe):
    # Capitalizes all the column headers
    dataframe.columns = dataframe.columns.str.upper()
    # And returns them
    return dataframe

# Create a pipeline that applies the mean_age_by_group function
(df.pipe(mean_age_by_group, col='gender')
 # then applies the uppercase column name function
 .pipe(uppercase_column_name)
 )
```

13.

```
data = {'Gender':['m','f','f','m','f','m','m'],'Height':[172,171,169,173,170,175,178]}
df_sample = pd.DataFrame(data)
df_sample
df_sample.groupby('Gender').mean()
```

14.

```
# Create a function that
def mean_age_by_group(dataframe, col):
    # groups the data by a column and returns the mean age per group
    return dataframe.groupby(col).mean()

# Create a function that
def uppercase_column_name(dataframe):
    # Capitalizes all the column headers
```

```
dataframe.columns = dataframe.columns.str.upper()

# And returns them

return dataframe

# Create a pipeline that applies the mean_age_by_group function
(df.pipe(mean_age_by_group, col='gender')
 # then applies the uppercase column name function
 .pipe(uppercase_column_name)
 )
```

15.

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(10, 4),
                  index = pd.date_range('1/1/2000', periods=10),
                  columns = ['A', 'B', 'C', 'D'])
print(df)
print(df.agg(np.sum))
print(df['A'].agg(np.mean))
print(df[['C','D']].agg(np.min))
```

16.

```
import pandas as pd

df1 = pd.DataFrame({'HPI':[80,85,88,85],
                   'Int_rate':[2, 3, 2, 2],
                   'US_GDP_Thousands':[50, 55, 65, 55]},
                  index = [2001, 2002, 2003, 2004])

df2 = pd.DataFrame({'HPI':[80,85,88,85],
                   'Int_rate':[2, 3, 2, 2],
                   'US_GDP_Thousands':[50, 55, 65, 55]},
                  index = [2005, 2006, 2007, 2008])

df3 = pd.DataFrame({'HPI':[80,85,88,85],
```

```

        'Int_rate':[2, 3, 2, 2],
        'Low_tier_HPI':[50, 52, 50, 53]},
        index = [2001, 2002, 2003, 2004])

concat = pd.concat([df1,df2,df3])

print(concat)

df4 = df1.append(df2)

print(df4)

```

17.

```

import pandas as pd
import numpy as np

# First, add the platform and device to the user usage - use a left join this time.
result = pd.merge(user_usage,
                  user_device[['use_id', 'platform', 'device']],
                  on='use_id',
                  how='left')

# At this point, the platform and device columns are included
# in the result along with all columns from user_usage
# Now, based on the "device" column in result, match the "Model" column in devices.
devices.rename(columns={"Retail Branding": "manufacturer"}, inplace=True)

'''result = pd.merge(result,
                    devices[['manufacturer', 'Model']],
                    left_on='device',
                    right_on='Model',
                    how='left')

print(result.head())'''

```

18.

```

import pandas as pd

one = pd.DataFrame({
    'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
    'subject_id': ['sub1', 'sub2', 'sub4', 'sub6', 'sub5'],
    'Marks_scored': [98,90,87,69,78]},
    index=[1,2,3,4,5])

```

```
two = pd.DataFrame({  
    'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],  
    'subject_id': ['sub2', 'sub4', 'sub3', 'sub6', 'sub5'],  
    'Marks_scored': [89, 80, 79, 97, 88]},  
    index=[1, 2, 3, 4, 5])  
print pd.concat([one, two], keys=['x', 'y'], ignore_index=True, axis = 1)
```



CHAPTER 6: Plotting Graphs

1.

```
import numpy as np
import matplotlib.pyplot as plt

medals1 = (12,20,8)
index = np.array([1,2,3])
width = 0.30
plt.bar(index, medals1, width)

plt.ylabel('Medals')
plt.title('Medals Won By Country')

plt.xticks(index + width / 2, ('Bronze', 'Silver', 'Gold'))
plt.legend(loc='best')
plt.show()
```

2.

```
import numpy as np
import matplotlib.pyplot as plt
countries = ['USA', 'GB', 'China', 'Russia', 'Germany']

bronzes = np.array([38, 17, 26, 19, 15])
silvers = np.array([37, 23, 18, 18, 10])
golds = np.array([46, 27, 26, 19, 17])
#ind = [x for x, _ in enumerate(countries)]
y_pos = np.arange(len(countries))

plt.barh(y_pos, golds, label='golds', color='gold')
plt.barh(y_pos, silvers, label='silvers', color='silver')
plt.barh(y_pos, bronzes, label='bronzes', color='#CD853F')
```

```
plt.yticks(y_pos, countries)
plt.ylabel("Countries")
plt.xlabel("Medals")
plt.legend(loc="upper right")
plt.title("2012 Olympics Top Scorers")
plt.show()
```

3.


```
import numpy as np
import matplotlib.pyplot as plt

medals1 = (12,20,8)
medals2 = (15,15, 10)

index = np.array([1,2,3])
width = 0.30
plt.bar(index, medals1, width, label = 'Medals Won By First Country')
plt.bar(index + width, medals2, width, label = 'Medals Won By First Country')

plt.ylabel('Medals')
plt.title('Comparing Medals Won By Country 1 and Country 2')

plt.xticks(index + width / 2, ('Bronze', 'Silver', 'Gold'))
plt.legend(loc='best')
plt.show()
```



4.

```
import numpy as np
import matplotlib.pyplot as plt

countries = ['USA', 'GB', 'China', 'Russia', 'Germany']

bronzes = np.array([38, 17, 26, 19, 15])
silvers = np.array([37, 23, 18, 18, 10])
golds = np.array([46, 27, 26, 19, 17])
ind = [x for x, _ in enumerate(countries)]
```

```
plt.bar(ind, golds, width=0.8, label='golds', color='gold', bottom=silvers+bronzes)
plt.bar(ind, silvers, width=0.8, label='silvers', color='silver', bottom=bronzes)
plt.bar(ind, bronzes, width=0.8, label='bronzes', color='#CD853F')

plt.xticks(ind, countries)
plt.ylabel("Medals")
plt.xlabel("Countries")
plt.legend(loc="upper right")
plt.title("2012 Olympics Top Scorers")
```

```
plt.show()
```

7.

```
import matplotlib.pyplot as plt
import pandas as pd
Launch_Year = ['2000','2004','2008','2011','2015','2019']
#speed = [95, 87, 75, 78, 98, 58]
speed = [120,160,180,180,160,170]
plt.scatter(Launch_Year, speed,label='Speed', color='red',marker = (5,2))

plt.title('Scatter Plot')
plt.xlabel('Launch Year')
plt.ylabel('Speed')
plt.legend()
plt.show()
```

8.

```
import matplotlib.pyplot as plt
import pandas as pd
girls_grades = [89, 90, 70, 89, 100, 80, 90, 100, 80, 34]
boys_grades = [30, 29, 49, 48, 100, 48, 38, 45, 20, 30]
grades_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
plt.scatter(grades_range, girls_grades, color='r')
```

```
plt.scatter(grades_range, boys_grades, color='g')
plt.xlabel('Grades Range')
plt.ylabel('Grades Scored')
plt.show()
```

The graph is clearly telling that girls performed way better than boys. It also tells that there are two outliers, one in guys and other in girls. One boy performed pretty well while a single girl did pretty bad.

```
import matplotlib.pyplot as plt
import numpy as np

weight1=[67,57.2,59.6,59.64,55.8,61.2,60.45,61,56.23,56]
height1=[101.7,197.6,98.3,125.1,113.7,157.7,136,148.9,125.3,114.9]
weight2=[61.9,64,62.1,64.2,62.3,65.4,62.4,61.4,62.5,63.6]
height2=[152.8,155.3,135.1,125.2,151.3,135,182.2,195.9,165.1,125.1]
weight3=[68.2,67.2,68.4,68.7,71,71.3,70.8,70,71.1,71.7]
height3=[165.8,170.9,192.8,135.4,161.4,136.1,167.1,235.1,181.1,177.3]
weight=np.concatenate((weight1,weight2,weight3))
height=np.concatenate((height1,height2,height3))
plt.scatter(weight, height, marker='*')
plt.xlabel('weight', fontsize=16)
plt.ylabel('height', fontsize=16)
plt.title('Group wise Weight vs Height scatter plot',fontsize=20)
plt.show()
```

9.

```
import numpy as np
import matplotlib.pyplot as plt

students = ['Mehek', 'Samurai', 'Neon', 'Jia']
Marks1 = np.array([18,14,17,10])
Marks2 = np.array([8,12,10,19])
Marks3 = np.array([21,9,15,11])
index = students

plt.bar(index, Marks1, width=0.5, label='C++', color='red', bottom = Marks2 + Marks3 )
plt.bar(index, Marks2, width=0.5, label='CSA', color='green', bottom = Marks3)
plt.bar(index, Marks3, width=0.5, label='Data Structures', color='blue')
```



```
plt.xticks(index, students)
plt.ylabel("Marks")
plt.xlabel("Students")
plt.legend(loc="upper right")
plt.title("Marks obtained in weekly tests")

plt.show()
```

10.

```
import numpy as np
import matplotlib.pyplot as plt
students = ['Mehek', 'Samurai', 'Neon', 'Jia']
Marks1 = np.array([18,14,17,10])
Marks2 = np.array([8,12,10,19])
Marks3 = np.array([21,9,15,11])
index = students
data=[Marks1, Marks2, Marks3]
box = plt.boxplot(data,patch_artist=True,notch = True, vert = True, meanline=True,showbox =
True, showmeans=True, labels=['CSA','C++','Data Structures'])
plt.title('STUDENTS PERFORMANCE')
plt.xlabel('STUDENTS')
plt.ylabel('MARKS')

colors = ['cyan', 'lightblue', 'lightgreen']
for patch, color in zip(box['boxes'], colors):
    patch.set_facecolor(color)
plt.show()
```

11.

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('iris')
sb.distplot(df['petal_length'])
plt.show()
```

12.

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('iris')
sb.jointplot(x = 'petal_length', y = 'petal_width', data = df)
plt.show()
```

13.

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('tips')
sb.violinplot(x = "day", y = "total_bill", data=df)
plt.show()
```

14.

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('tips')
sb.violinplot(x = "day", y = "total_bill", hue = 'sex', data = df)
plt.show()
```

15.

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('titanic')
sb.countplot(x = " class ", data = df, palette = "Blues");
plt.show()
```

16.

```
from matplotlib import pyplot as plt
df = sb.load_dataset('iris')
sb.boxplot(data = df, orient = "h")
plt.show()
```

17.

```

importing packages
import seaborn

import matplotlib.pyplot as plt

# loading of a dataframe from seaborn
df = seaborn.load_dataset('tips')

##### Main Section #####

# Form a facetgrid using columns with a hue
graph = seaborn.FacetGrid(df, col = "sex", hue = "day")
# map the above form facetgrid with some attributes
graph.map(plt.scatter, "total_bill", "tip", edgecolor = "w").add_legend()
# show the object
plt.show()

```

18.

```

import seaborn as sns
import matplotlib.pyplot as plt

# loading dataset
data = sns.load_dataset("titanic")

# draw regplot
sns.regplot(x = "age",
            y = "fare",
            data = data,
            dropna = True)

# show the plot
plt.show()

```

19.

```

import pandas as pd
import seaborn as sb

from matplotlib import pyplot as plt

```

```
df = sb.load_dataset('anscombe')
sb.lmplot(x="x", y="y", data=df.query("dataset == 'I'"))
plt.show()
```

20.

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('tips')
g = sb.FacetGrid(df, col = "sex", hue = "smoker")
g.map(plt.scatter, "total_bill", "tip")
plt.show()
```

21.

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('iris')
g = sb.PairGrid(df)
g.map_upper(plt.scatter)
g.map_lower(sb.kdeplot, cmap = "Blues_d")
g.map_diag(sb.kdeplot, lw = 3, legend = False);
plt.show()
```

22.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data_url = 'http://bit.ly/2cLzoxH'
gapminder = pd.read_csv(data_url)
print(gapminder.head(3))
df1 = gapminder[['continent', 'year', 'lifeExp']]
print(df1.head())
# pandas pivot
heatmap1_data = pd.pivot_table(df1, values='lifeExp',
                                index=['continent'],
                                columns='year')
```

```
sns.heatmap(heatmap1_data, cmap="YlGnBu")
plt.show()
```

```
df2 = gapminder[['country', 'continent', 'year', 'lifeExp']]
heatmap2_data = pd.pivot_table(df2, values='lifeExp', index=['country'], columns='year')
heatmap2_data.head(n=5)
sns.heatmap(heatmap2_data, cmap="BuGn")
plt.show()
```

```
df3 = gapminder[['country', 'continent', 'year', 'lifeExp']]
# pandas pivot with multiple variables
heatmap3_data = pd.pivot_table(df3, values='lifeExp', index=['continent', 'country'],
columns='year')
plt.figure(figsize=(8, 12))
sns.heatmap(heatmap3_data, cmap="RdBu")
plt.show()
```

23.

```
from pandas import read_csv
from matplotlib import pyplot
series = read_csv('daily-minimum-temperatures.csv', header=0, index_col=0,
parse_dates=True, squeeze=True)
series.plot()
pyplot.show()
```

```
from pandas import read_csv
from matplotlib import pyplot
series = read_csv('daily-minimum-temperatures.csv', header=0, index_col=0,
parse_dates=True, squeeze=True)
series.hist()
pyplot.show()
from pandas import read_csv
from matplotlib import pyplot
series = read_csv('daily-minimum-temperatures.csv', header=0, index_col=0,
parse_dates=True, squeeze=True)
series.plot(kind='kde')
```

```

pyplot.show()

from pandas import read_csv
from pandas import DataFrame
from pandas import Grouper
from matplotlib import pyplot

series = read_csv('daily-minimum-temperatures.csv', header=0, index_col=0,
parse_dates=True, squeeze=True)

groups = series.groupby(Grouper(freq='A'))

years = DataFrame()

for name, group in groups:
    years[name.year] = group.values

years.boxplot()
pyplot.show()

```

24.

```

from datetime import datetime
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv('path_to_file/stock.csv')
df = pd.DataFrame(data, columns = ['ValueDate', 'Price'])

# Set the Date as Index
df['ValueDate'] = pd.to_datetime(df['ValueDate'])
df.index = df['ValueDate']
del df['ValueDate']

df.plot(figsize=(15, 6))
plt.show()

```

25.

```

from matplotlib import pyplot
from pandas import read_csv
from pandas.tools.plotting import scatter_matrix

path = r"C:\pima-indians-diabetes.csv"

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

```

```
data = read_csv(path, names=names)
scatter_matrix(data)
pyplot.show()
```

26.

```
from datetime import datetime
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv('path_to_file/stock.csv')
df = pd.DataFrame(data, columns = ['ValueDate', 'Price'])

# Set the Date as Index
df['ValueDate'] = pd.to_datetime(df['ValueDate'])
df.index = df['ValueDate']
del df['ValueDate']

df.plot(figsize=(15, 6))
plt.show()
```



CHAPTER 7: File Handling

1.

```
filename = input("Enter the input filename : ")
filename2 = input("Enter the output filename : ")
with open(filename) as file:
    with open(filename2,"w") as file2:
        buf = file.read()
        lines = buf.split('\n')
        l = len(lines)
        new_buf = ""
        for i in range(l-1, 0, -1):
            print(lines[i])
            new_buf += lines[i]
        file2.write(new_buf)
```

2.

```
filename = input("Enter the input filename : ")
with open(filename) as file:
    buf = file.read()
    lines = buf.split('\n')
    for i in lines:
        if(i.find("print", 0, len(i))!=-1):
            print(i)
```

3.

```
filename = input("Enter the input filename : ")
file2 = input("Enter the output filename : ")
with open(filename) as file:
    with open(file2,"w") as ofile:
        buf = file.read()
        lines = buf.split('\n')
        new_buf = ""
```



```

for i in lines:
    for ch in range(2, len(i)):
        new_buf += i[ch]
    new_buf += "\n"
ofile.write(new_buf)

```

4.

```

filename = input("Enter the first filename : ")
filename2 = input("Enter the second filename : ")
with open(filename) as file1:
    with open(filename2) as file2:
        buf1 = file1.read()
        buf2 = file2.read()
        if(buf1==buf2):
            print("SAME")
        else:
            print("NOT SAME")

```

5.

```

filename = input("Enter the first filename : ")
filename2 = input("Enter the second filename : ")
with open(filename) as file1:
    with open(filename2,"w") as file2:
        buf = file1.read()
        for i in buf:
            file2.write(i)

```

6.

Read the file line by line and print

7.

Print count the number of lines in the file and print. (if every record consumes one line)

8.

```

filename = input("Enter the first filename : ")
filename2 = input("Enter the second filename : ")
with open(filename, ) as file1:

```

```
with open(filename2,"w") as file2:
    buf = file1.read()
    buf = buf.upper()
    file2.write(buf)
```

9.

```
filename1 = input("Enter the first filename : ")
filename2 = input("Enter the second filename : ")
filename3 = input("Enter the third filename : ")
with open(filename1) as file1:
    with open(filename2) as file2:
        with open(filename3, "w") as file3:
            buf1 = file1.read()
            buf2 = file2.read()
            buf3 = buf1 + buf2
            file3.write(buf3)
```

10.

```
filename1 = input("Enter the first filename : ")
filename2 = input("Enter the second filename : ")
with open(filename1) as file1:
    with open(filename2,"w") as file2:
        buf = file1.read()
        for c in buf:
            if(c=='.'):
                file2.write(',')
            else:
                file2.write(c)
```

11.

```
filename1 = input("Enter the first filename : ")
filename2 = input("Enter the second filename : ")
with open(filename1, "r") as file1:
    with open(filename2,"r") as file2:
        buf1 = file1.read()
        buf2 = file2.read()
```

```

with open(filename1, "w") as file1:
    with open(filename2, "w") as file2:
        temp = buf1
        buf1 = buf2
        buf2 = temp
        file1.write(buf1)
        file2.write(buf2)

```

12.

```

filename1 = input("Enter the first filename : ")
with open(filename1) as file1:
    buf = file1.read()
print(buf)
with open(filename1, "w") as file1:
    new_buf = ""
    i=0
    l = len(buf)
    while(i<l and buf[i]!='\0'):
        if(buf[i-2]=='.'):
            new_buf += buf[i].upper()
        else:
            new_buf += buf[i]
        i+=1
    print(new_buf)
    file1.write(new_buf)

```

13.

```

import pickle
L = ['a', 'b', 'c', 'd']
with open('datafile.txt', 'wb') as file:
    pickle.dump(L, file)
print("Data Written Successfully .....")
file = open ("datafile.txt", "rb")
L = pickle.load(file)
print(L)

```

14.

```
{
  "ID":["1","2","3","4","5","6","7","8"],
  "Name":["Rocky","Danish","Mikal","Raima","Gauri","Neon","Siya","Garv"],
  "Deptt":["IT","HR","Finance","Sales","Marketing","Accounts","Payroll","Manufacturing"],
  "Desig":["Manager","Trainee","Team Leader","IT Assistant","VP","Director","Coach","Data Analyst"]
}

import pandas as pd
data = pd.read_json('Emp.json')
print (data)
```

15.

```
{
  "book": [
    {
      "id":"01",
      "language": "Java",
      "edition": "third",
      "author": "Herbert Schildt"
    },
    {
      "id":"07",
      "language": "C++",
      "edition": "second",
      "author": "E.Balagurusamy"
    }
  ]
}
```



16.

```
{
  "ID":["1","2","3","4","5","6","7","8"],
  "Name":["Rick","Dan","Michelle","Ryan","Gary","Nina","Simon","Guru"]
}
```

```

"Salary":["623.3","515.2","611","729","843.25","578","632.8","722.5"],

"StartDate":["1/1/2012","9/23/2013","11/15/2014","5/11/2014","3/27/2015","5/21/2013",
            "7/30/2013","6/17/2014"],

"Dept":["IT","Operations","IT","HR","Finance","IT","Operations","Finance"]
}

```

```
import pandas as pd
```

```

data = pd.read_json('path/input.json')
print (data)

```

```

import pandas as pd
data = pd.read_json('path/input.xlsx')

# Use the multi-axes indexing funtion
print (data.loc[[1,3,5],['salary','name']])

```

17.

Python program to write JSON to a file

```

import json

# Data to be written
dictionary = {
    "name" : "sathiyajith",
    "ID" : 56,
    "Rating" : 8.6,
    "phonenummer" : "9976770500"
}

# Serializing json
json_object = json.dumps(dictionary, indent = 4)

# Writing to sample.json
with open("sample.json", "w") as outfile:
    outfile.write(json_object)

```

18.

```
{
    "firstName": "John",
    "lastName": "Smith",
    "isAlive": true,
    "age": 27,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021-3100"
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "office",
            "number": "646 555-4567"
        },
        {
            "type": "mobile",
            "number": "123 456-7890"
        }
    ],
    "children": [],
    "spouse": null
}

with open('person_list.json', 'w') as f:
    json.dump(persons, f)
open('person_list.json', 'r').read()
```

CHAPTER 8: Interfacing Python with MySQL

Question 1

- a) CREATE TABLE Employee(FIRST_NAME varchar(10), LAST_NAME varchar(10), AGE int, SEX char(2), INCOME decimal(10,2));
- b) INSERT INTO Employee VALUES('Tarini', 'Patel', '20', 'F', '2000');
INSERT INTO Employee VALUES('Tara', 'Mani', '19', 'F', '900');
INSERT INTO Employee VALUES('Sachin', 'Kapoor', '19', 'M', '1000');
INSERT INTO Employee VALUES('Tarun', 'Khurana', '19', 'M', '800');
INSERT INTO Employee VALUES('Rohan', 'Saxena', '20', 'M', '1200');
INSERT INTO Employee VALUES('Vani', 'Tandon', '19', 'F', '1500');
INSERT INTO Employee VALUES('Anya', 'Tripathi', '18', 'F', '900');
INSERT INTO Employee VALUES('Arti', 'Gupta', '21', 'F', '1700');
INSERT INTO Employee VALUES('Dilip', 'Yadav', '25', 'M', '1800');
INSERT INTO Employee VALUES('Ishvina', 'Kapoor', '23', 'F', '2100');
INSERT INTO Employee VALUES('Sonia', 'Seth', '49', 'F', '5200');

```
mysql> SELECT * FROM Employee;
```

FIRST_NAME	LAST_NAME	AGE	SEX	INCOME
Tarini	Patel	20	F	2000.00
Tara	Mani	19	F	900.00
Sachin	Kapoor	19	M	1000.00
Tarun	Khurana	19	M	800.00
Rohan	Saxena	20	M	1200.00
Vani	Tandon	19	F	1500.00
Anya	Tripathi	18	F	900.00
Arti	Gupta	21	F	1700.00
Dilip	Yadav	25	M	1800.00
Ishvina	Kapoor	23	F	2100.00
Sonia	Seth	49	F	5200.00

FIRST_NAME	LAST_NAME	AGE	SEX	INCOME
Tarini	Patel	20	F	2000.00
Tara	Mani	19	F	900.00
Sachin	Kapoor	19	M	1000.00
Tarun	Khurana	19	M	800.00
Rohan	Saxena	20	M	1200.00
Vani	Tandon	19	F	1500.00

Anya	Tripathi	18	F	900.00	
Arti	Gupta	21	F	1700.00	
Dilip	Yadav	25	M	1800.00	
Ishvina	Kapoor	23	F	2100.00	
Sonia	Seth	49	F	5200.00	
+-----+-----+-----+-----+-----+					

c) `SELECT * FROM Employee WHERE INCOME > 1000;`

FIRST_NAME	LAST_NAME	AGE	SEX	INCOME	
Tarini	Patel	20	F	2000.00	
Rohan	Saxena	20	M	1200.00	
Vani	Tandon	19	F	1500.00	
Arti	Gupta	21	F	1700.00	
Dilip	Yadav	25	M	1800.00	
Ishvina	Kapoor	23	F	2100.00	
Sonia	Seth	49	F	5200.00	
+-----+-----+-----+-----+-----+					

FIRST_NAME	LAST_NAME	AGE	SEX	INCOME	
Tarini	Patel	20	F	2000.00	
Rohan	Saxena	20	M	1200.00	
Vani	Tandon	19	F	1500.00	
Arti	Gupta	21	F	1700.00	
Dilip	Yadav	25	M	1800.00	
Ishvina	Kapoor	23	F	2100.00	
Sonia	Seth	49	F	5200.00	
+-----+-----+-----+-----+-----+					

d) `UPDATE Employee SET AGE = AGE+1 WHERE SEX='M';`

`SELECT * FROM Employee;`

FIRST_NAME	LAST_NAME	AGE	SEX	INCOME	
Tarini	Patel	20	F	2000.00	
Tara	Mani	19	F	900.00	
Sachin	Kapoor	20	M	1000.00	
Tarun	Khurana	20	M	800.00	
Rohan	Saxena	21	M	1200.00	
Vani	Tandon	19	F	1500.00	
Anya	Tripathi	18	F	900.00	
Arti	Gupta	21	F	1700.00	
Dilip	Yadav	26	M	1800.00	
Ishvina	Kapoor	23	F	2100.00	
Sonia	Seth	49	F	5200.00	
+-----+-----+-----+-----+-----+					

FIRST_NAME	LAST_NAME	AGE	SEX	INCOME
Tarini	Patel	20	F	2000.00
Tara	Mani	19	F	900.00
Sachin	Kapoor	20	M	1000.00
Tarun	Khurana	20	M	800.00
Rohan	Saxena	21	M	1200.00
Vani	Tandon	19	F	1500.00
Any	Tripathi	18	F	900.00
Arti	Gupta	21	F	1700.00
Dilip	Yadav	26	M	1800.00
Ishvina	Kapoor	23	F	2100.00
Sonia	Seth	49	F	5200.00

e) DELETE FROM Employee WHERE AGE>20;

SELECT * FROM Employee;

FIRST_NAME	LAST_NAME	AGE	SEX	INCOME
Tarini	Patel	20	F	2000.00
Tara	Mani	19	F	900.00
Sachin	Kapoor	20	M	1000.00
Tarun	Khurana	20	M	800.00
Vani	Tandon	19	F	1500.00
Any	Tripathi	18	F	900.00

FIRST_NAME	LAST_NAME	AGE	SEX	INCOME
Tarini	Patel	20	F	2000.00
Tara	Mani	19	F	900.00
Sachin	Kapoor	20	M	1000.00
Tarun	Khurana	20	M	800.00
Vani	Tandon	19	F	1500.00
Any	Tripathi	18	F	900.00

Question 2**a) Creating the table LibraryBooks:**

```
CREATE TABLE LibraryBooks(Accession_Number int PRIMARY KEY, Title varchar(25), Author
varchar(25), Department varchar(25), PurchaseDate date, Price decimal(10,2));
```

Inserting values into the table LibraryBooks :

```
INSERT INTO LibraryBooks VALUES(1910,'Oracle Database','S B Navathe','CS','2002-03-01',450);
INSERT INTO LibraryBooks VALUES(1205,'Discrete Maths','M Freeman','MATHS','2005-04-18',559);
INSERT INTO LibraryBooks VALUES(9971,'Data Structures','S B Navathe','CS','2003-07-24',670);
INSERT INTO LibraryBooks VALUES(1312,'Intro To CS','S Taneja','CS','1998-02-23',789);
INSERT INTO LibraryBooks VALUES(2406,'Database System Concepts','M Hume','CS','2005-12-
08',939);
```

Displaying the Table LibraryBooks :

```
SELECT * FROM LibraryBooks;
```

Accession_Number	Title	Author	Department	PurchaseDate	Price
1205	Discrete Maths	M Freeman	MATHS	2005-04-18	559.00
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00
2406	Database System Concepts	M Hume	CS	2005-12-08	939.00
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00

Accession_Number	Title	Author	Department	PurchaseDate	Price
1205	Discrete Maths	M Freeman	MATHS	2005-04-18	559.00
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00
2406	Database System Concepts	M Hume	CS	2005-12-08	939.00
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00

Creating the table IssuedBooks :

```
CREATE TABLE IssuedBooks(Accession_Number int, Borrower varchar(25) PRIMARY KEY,
FOREIGN KEY(Accession_Number) REFERENCES LibraryBooks(Accession_Number));
```

Inserting values into the table IssuedBooks :

```
INSERT INTO IssuedBooks VALUES(1205, 'Rob');
INSERT INTO IssuedBooks VALUES(9971, 'Robert');
INSERT INTO IssuedBooks VALUES(1312, 'Katie');
INSERT INTO IssuedBooks VALUES(2406, 'Mark');
INSERT INTO IssuedBooks VALUES(1910, 'Daisy');
```

Displaying the Table IssuedBooks:

```
SELECT * FROM IssuedBooks;
```

Accession_Number	Borrower
1205	Rob
1312	Katie
1910	Daisy
2406	Mark
9971	Robert

Accession_Number	Borrower
1205	Rob
1312	Katie
1910	Daisy
2406	Mark
9971	Robert

b) Delete the record of book titled “Database System Concepts”

```
DELETE FROM LibraryBooks WHERE Title = 'Database System Concepts';
```

Accession_Number	Title	Author	Department	PurchaseDate	Price
1205	Discrete Maths	M Freeman	MATHS	2005-04-18	559.00
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00

c) Change the Department of the book titled “Discrete Maths” to CS

```
UPDATE LibraryBooks SET Department = 'CS' WHERE Title = 'Discrete Maths';
SELECT * FROM LibraryBooks;
```

Accession_Number	Title	Author	Department	PurchaseDate	Price
1205	Discrete Maths	M Freeman	CS	2005-04-18	559.00
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00

Accession_Number	Title	Author	Department	PurchaseDate	Price
1205	Discrete Maths	M Freeman	CS	2005-04-18	559.00
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00

d) List all books that belongs to “CS” Department

```
SELECT * FROM LibraryBooks WHERE Department = 'CS';
```

Accession_Number	Title	Author	Department	PurchaseDate	Price
1205	Discrete Maths	M Freeman	CS	2005-04-18	559.00
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00

Accession_Number	Title	Author	Department	PurchaseDate	Price
1205	Discrete Maths	M Freeman	CS	2005-04-18	559.00
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00

e) List all books that belongs to “CS” department and are written by author “Navathe”

```
SELECT * FROM LibraryBooks WHERE Department = 'CS' AND Author = 'S B Navathe';
```

Accession_Number	Title	Author	Department	PurchaseDate	Price
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00

f) List all Computer (Department = “CS”) that have been issued

```
SELECT * FROM LibraryBooks, IssuedBooks WHERE LibraryBooks.Accession_Number =  
IssuedBooks.Accession_Number AND Department = 'CS';
```

Accession_Number	Title	Author	Department	PurchaseDate	Price	Accession_Number	Borrower
1205	Discrete Maths	M Freeman	CS	2005-04-18	559.00	1205	Rob
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00	1312	Katie
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00	1910	Daisy
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00	9971	Robert

Accession_Number	Title	Author	Department	PurchaseDate	Price	Accession_Number	Borrower
1205	Discrete Maths	M Freeman	CS	2005-04-18	559.00	1205	Rob
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00	1312	Katie
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00	1910	Daisy
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00	9971	Robert

g) List all books which have a price less than 500 or purchased between "01/01/1999" and "01/01/2004"

```
SELECT * FROM LibraryBooks, IssuedBooks WHERE Price < 500 OR PurchaseDate BETWEEN '1999-01-01' AND '2004-01-01';
```

Accession_Number	Title	Author	Department	PurchaseDate	Price	Accession_Number	Borrower
1205	Discrete Maths	M Freeman	CS	2005-04-18	559.00	1205	Rob
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00	1205	Rob
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00	1205	Rob
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00	1205	Rob
1205	Discrete Maths	M Freeman	CS	2005-04-18	559.00	1312	Katie
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00	1312	Katie
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00	1312	Katie
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00	1312	Katie
1205	Discrete Maths	M Freeman	CS	2005-04-18	559.00	1910	Daisy
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00	1910	Daisy
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00	1910	Daisy
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00	1910	Daisy
1205	Discrete Maths	M Freeman	CS	2005-04-18	559.00	9971	Robert
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00	9971	Robert
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00	9971	Robert
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00	9971	Robert

Accession_Number	Title	Author	Department	PurchaseDate	Price	Accession_Number	Borrower
1205	Discrete Maths	M Freeman	CS	2005-04-18	559.00	1205	Rob
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00	1205	Rob
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00	1205	Rob
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00	1205	Rob
1205	Discrete Maths	M Freeman	CS	2005-04-18	559.00	1312	Katie
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00	1312	Katie
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00	1312	Katie
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00	1312	Katie
1205	Discrete Maths	M Freeman	CS	2005-04-18	559.00	1910	Daisy
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00	1910	Daisy
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00	1910	Daisy
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00	1910	Daisy
1205	Discrete Maths	M Freeman	CS	2005-04-18	559.00	9971	Robert
1312	Intro To CS	S Taneja	CS	1998-02-23	789.00	9971	Robert
1910	Oracle Database	S B Navathe	CS	2002-03-01	450.00	9971	Robert
9971	Data Structures	S B Navathe	CS	2003-07-24	670.00	9971	Robert

Question 3

a) C

Creating Table Personal_Info:

```
CREATE TABLE Personal_Infor(C_RNo int PRIMARY KEY, Name varchar(25), DOB date, Address
varchar(25),B_Marks int, P_No int(10));
```

Inserting values into table Personal_Info :

```
INSERT INTO Personal_Infor VALUES(1191,'James','2001-03-14', 'Delhi',89,'889904321');
INSERT INTO Personal_Infor VALUES(1192,'Mark','2001-06-12', 'Pune',77, 987654321);
INSERT INTO Personal_Infor VALUES(1193,'Carol','2001-11-04', 'Pune',67, 976976976);
INSERT INTO Personal_Infor VALUES(1194,'Albert','2001-09-23', 'Mumbai',95, 987567678);
INSERT INTO Personal_Infor VALUES(1195,'Alex','2001-12-09', 'Delhi',68, 957376654);
```

Displaying the table Personal_Infor :

```
SELECT * FROM Personal_Infor;
```

C_RNo	Name	DOB	Address	B_Marks	P_No
1191	James	2001-03-14	Delhi	89	889904321
1192	Mark	2001-06-12	Pune	77	987654321
1193	Carol	2001-11-04	Pune	67	976976976
1194	Albert	2001-09-23	Mumbai	95	987567678
1195	Alex	2001-12-09	Delhi	68	957376654

C_RNo	Name	DOB	Address	B_Marks	P_No
1191	James	2001-03-14	Delhi	89	889904321
1192	Mark	2001-06-12	Pune	77	987654321
1193	Carol	2001-11-04	Pune	67	976976976
1194	Albert	2001-09-23	Mumbai	95	987567678
1195	Alex	2001-12-09	Delhi	68	957376654

Creating Table Paper_Detailss:

```
CREATE TABLE Paper_Detailss(P_Code varchar(25) PRIMARY KEY, Paper_Name varchar(25));
```

Inserting values into Paper_Detailss:

```
INSERT INTO Paper_Detailss VALUES(1, 'Paper 1');
INSERT INTO Paper_Detailss VALUES(2, 'Paper 2');
INSERT INTO Paper_Detailss VALUES(3, 'Paper 3');
INSERT INTO Paper_Detailss VALUES(4, 'Paper 4');
INSERT INTO Paper_Detailss VALUES(5, 'Paper 5');
```

Displaying the table Paper_Detailss:

```
SELECT * FROM Paper_Detailss;
```

P_Code	Paper_Name
1	Paper 1
2	Paper 2
3	Paper 3
4	Paper 4
5	Paper 5

P_Code	Paper_Name
1	Paper 1
2	Paper 2
3	Paper 3
4	Paper 4
5	Paper 5

Creating the table Acad_Attendnce:

```
CREATE TABLE Acad_Attendnce(C_RNo int, P_Code varchar(25) PRIMARY KEY, Attendance int,
Home_Marks int, FOREIGN KEY(C_RNo) REFERENCES Personal_Infor(C_RNo));
```

Inserting values into Acad_Attendnce:

```
INSERT INTO Acad_Attendnce VALUES(1191,1,74, 79);
INSERT INTO Acad_Attendnce VALUES(1192, 2,90, 95);
INSERT INTO Acad_Attendnce VALUES(1193,3,74, 79);
INSERT INTO Acad_Attendnce VALUES(1194, 4,94, 99);
INSERT INTO Acad_Attendnce VALUES(1195,5,90, 67);
```

Displaying the table Acad_Attendnce:

```
SELECT * FROM Acad_Attendnce;
```

```
+-----+-----+-----+-----+
| C_RNo | P_Cod | Attendance | Home_Marks |
+-----+-----+-----+-----+
| 1191  | 1     | 74         | 79          |
| 1192  | 2     | 90         | 95          |
| 1193  | 3     | 74         | 79          |
| 1194  | 4     | 94         | 99          |
| 1195  | 5     | 90         | 67          |
+-----+-----+-----+-----+
```

```
+-----+-----+-----+-----+
| C_RNo | P_Code | Attendance | Home_Marks |
+-----+-----+-----+-----+
| 1191  | 1     | 74         | 79          |
| 1192  | 2     | 90         | 95          |
| 1193  | 3     | 74         | 79          |
| 1194  | 4     | 94         | 99          |
| 1195  | 5     | 90         | 67          |
+-----+-----+-----+-----+
```

- b) `SELECT Paper_Name, Name FROM Personal_Infor, Paper_Detailss, Acad_Attendnce WHERE Attendance > 75 AND Home_Marks > 65 AND Paper_Name = 'Paper 2' AND Personal_Infor.C_RNo = Acad_Attendnce.C_RNo AND Paper_Detailss.P_Code = Acad_Attendnce.P_Code;`

```
+-----+-----+
| Paper_Name | Name |
+-----+-----+
| Paper 2    | Mark |
+-----+-----+
```

```
+-----+-----+
| Paper_Name | Name |
+-----+-----+
| Paper 2    | Mark |
+-----+-----+
```


- c) SELECT Name, Address FROM Personal_Infor, Paper_Detailss, Acad_Attendnce WHERE Address = 'Delhi' AND Paper_Name = 'Paper 1' AND Personal_Infor.C_RNo = Acad_Attendnce.C_RNo AND Paper_Detailss.P_Code = Acad_Attendnce.P_Code;

```
+-----+-----+
| Name | Address |
+-----+-----+
| James | Delhi   |
+-----+-----+
```

```
+-----+-----+
| Name | Address |
+-----+-----+
| James | Delhi   |
+-----+-----+
```

- d) SELECT C_RNo, SUM(Attendance) AS Total_Attendance, SUM(Home_Marks) AS Total_Marks FROM Acad_Attendnce GROUP BY C_RNo;

```
+-----+-----+-----+
| C_RNo | Total_Attendance | Total_Marks |
+-----+-----+-----+
| 1191 | 74 | 79 |
| 1192 | 90 | 95 |
| 1193 | 74 | 79 |
| 1194 | 94 | 99 |
| 1195 | 90 | 67 |
+-----+-----+-----+
```

```
+-----+-----+-----+
| C_RNo | Total_Attendance | Total_Marks |
+-----+-----+-----+
| 1191 | 74 | 79 |
| 1192 | 90 | 95 |
| 1193 | 74 | 79 |
| 1194 | 94 | 99 |
| 1195 | 90 | 67 |
+-----+-----+-----+
```

- e) SELECT Name, Home_Marks AS Max_Marks FROM Personal_Infor, Acad_Attendnce WHERE Personal_Infor.C_RNo = Acad_Attendnce.C_RNo AND Home_Marks IN (SELECT MAX(Home_Marks) FROM Acad_Attendnce, Paper_Detailss WHERE Paper_Name = 'Paper 2' AND Paper_Detailss.P_Code = Acad_Attendnce.P_Code);

```
+-----+-----+
| Name | Max_Marks |
+-----+-----+
| Mark | 95 |
+-----+-----+
```

Name	Max_Marks
Mark	95

Question 4

a) create table Customer(CustID varchar(4),email varchar(30),Name varchar(20),Phone int,ReferrerID varchar(10));

Query OK, 0 rows affected (0.07 sec)

mysql> create table Bicycle(BicycleID varchar(4),DatePurchased date,Color varchar(10),CustID varchar(4),ModelNo varchar(7));

Query OK, 0 rows affected (0.05 sec)

mysql> create table BicycleModel(ModelNo varchar(7),Manufacturer varchar(20),Style varchar(10));

Query OK, 0 rows affected (0.06 sec)

mysql> create table Service(StartDate date, BicycleID varchar(4),EndDate date);

Query OK, 0 rows affected (0.05 sec)

insert into Customer values("C1","abc@gmail.com","abc",98102,"C1-10"),("C2","def@gmail.com","def",89012,"C2-20"),("C3","qwe@gmail.com","qwe",67890,"C3-30"),("C4","vfg@gmail.com","vfg",78965,"C3-40"),("C5","xyz@gmail.com","xyz",67895,"C5-50");

Query OK, 5 rows affected (0.01 sec)

Records: 5 Duplicates: 0 Warnings: 0

insert into Bicycle values("B1","2017-10-08","Red","C1","M1-10"),("B2","2019-12-09","Red","C3","M2-50"),("B3","2016-03-22","Blue","C1","M6-20"),("B4","2021-07-13","Yellow","C4","C3-40"),("B5","2020-12-12","Purple","C2","M5-90");

Query OK, 5 rows affected (0.01 sec)

Records: 5 Duplicates: 0 Warnings: 0

insert into BicycleModel values("M1-10","Hero","Road"),("M2-50","Honda","Mountain"),("M6-20","Atlas","Foalading"),("C3-40","Hero","Road"),("M5-90","Honda","Foalading");

Query OK, 5 rows affected (0.01 sec)

Records: 5 Duplicates: 0 Warnings: 0

```
insert into Service values("2017-10-08","B1","2018-06-18"),("2019-12-09","B2","2020-08-29"),("2016-03-22","B3","2016-10-14"),("2021-07-13","B4","2022-01-26"),("2020-12-12","B5","2021-06-12");
```

Query OK, 5 rows affected (0.02 sec)

Records: 5 Duplicates: 0 Warnings: 0

```
mysql> select * from Customer;
+-----+-----+-----+-----+-----+
| CustID | email          | Name | Phone | ReferrerID |
+-----+-----+-----+-----+-----+
| C1     | abc@gmail.com  | abc  | 98102 | C1-10      |
| C2     | def@gmail.com  | def  | 89012 | C2-20      |
| C3     | qwe@gmail.com  | qwe  | 67890 | C3-30      |
| C4     | vfg@gmail.com  | vfg  | 78965 | C3-40      |
| C5     | xyz@gmail.com  | xyz  | 67895 | C5-50      |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select * from Bicycle;
+-----+-----+-----+-----+-----+
| BicycleID | DatePurchased | Color | CustID | ModelNo |
+-----+-----+-----+-----+-----+
| B1        | 2017-10-08    | Red   | C1     | M1-10    |
| B2        | 2019-12-09    | Red   | C3     | M2-50    |
| B3        | 2016-03-22    | Blue  | C1     | M6-20    |
| B4        | 2021-07-13    | Yellow| C4     | C3-40    |
| B5        | 2020-12-12    | Purple| C2     | M5-90    |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select * from BicycleModel;
+-----+-----+-----+
| ModelNo | Manufacturer | Style |
+-----+-----+-----+
| C3-40    | Hero         | Road  |
| M1-10    | Hero         | Road  |
| M2-50    | Honda        | Mountain |
| M5-90    | Honda        | Foalading |
| M6-20    | Atlas        | Foalading |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select * from Service;
+-----+-----+-----+
| StartDate | BicycleID | EndDate |
+-----+-----+-----+
| 2017-10-08 | B1        | 2018-06-18 |
| 2019-12-09 | B2        | 2020-08-29 |
| 2016-03-22 | B3        | 2016-10-14 |
| 2021-07-13 | B4        | 2022-01-26 |
| 2020-12-12 | B5        | 2021-06-12 |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> alter table Customer add primary key (CustID);
```

Query OK, 0 rows affected (0.15 sec)

Records: 0 Duplicates: 0 Warnings: 0

```
mysql> alter table Bicycle add primary key (BicycleID);
```

Query OK, 0 rows affected (0.10 sec)

Records: 0 Duplicates: 0 Warnings: 0

```
mysql> alter table BicycleModel add primary key (ModelNo);
```

Query OK, 0 rows affected (0.07 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table Service add primary key (BicycleID);

Query OK, 0 rows affected (0.07 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> alter table Bicycle add foreign key (CustID) references Customer(CustID);

Query OK, 5 rows affected (0.17 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> alter table Service add foreign key (BicycleID) references Bicycle(BicycleID);

Query OK, 5 rows affected (0.11 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> alter table Bicycle add foreign key (CustID) references Customer(CustID);

Query OK, 5 rows affected (0.08 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> alter table Bicycle add foreign key (ModelNo) references BicycleModel(ModelNo);

Query OK, 5 rows affected (0.09 sec)

Records: 5 Duplicates: 0 Warnings: 0

b) mysql> select c.CustId, c.Name, b.ModelNo, m.manufacturer from Bicycle b join BicycleModel m on b.ModelNo=M.ModelNo join Customer c on c.CustID=b.CustID where m.Manufacturer="Honda";

CustId	Name	ModelNo	manufacturer
C3	qwe	M2-50	Honda
C2	def	M5-90	Honda

2 rows in set (0.01 sec)

c) mysql> select c.CustId, c.Name, b.BicycleID, b.DatePurchased, b.color, m.ModelNo from Bicycle b join BicycleModel m on b.ModelNo=M.ModelNo join Customer c on c.CustID=b.CustID where c.ReferrerId="C1-10";

```

+-----+-----+-----+-----+-----+
| CustId | Name | BicycleID | DatePurchased | color | ModelNo |
+-----+-----+-----+-----+-----+
| C1     | abc  | B1       | 2017-10-08    | Red   | M1-10   |
| C1     | abc  | B3       | 2016-03-22.   | Blue  | M6-20   |
+-----+-----+-----+-----+-----+

```

2 rows in set (0.00 sec)

d) mysql> select b.color,m.Manufacturer from Bicycle b join BicycleModel m on
b.ModelNo=M.ModelNo where b.color="Red";

```

+-----+-----+
| color | Manufacturer |
+-----+-----+
| Red   | Hero         |
| Red   | Honda        |
+-----+-----+

```

2 rows in set (0.00 sec)

e) mysql> select b.ModelNo from Bicycle b join Service s on b.BicycleID=s.BicycleID where
b.BicycleID=s.BicycleID;

ERROR 1054 (42S22): Unknown column 'b.BicycleID' in 'where clause'

mysql> select b.ModelNo from Bicycle b join Service s on b.BicycleID=s.BicycleID where
b.BicycleID=s.BicycleID;

```

+-----+
| ModelNo |
+-----+
| M1-10   |
| M2-50   |
| M6-20   |
| C3-40   |
| M5-90   |
+-----+

```

5 rows in set (0.00 sec)

Question 5:

a) `mysql> create table Company(Company_Name varchar(20),City varchar(20),primary key(Company_Name));`

Query OK, 0 rows affected (0.05 sec)

`mysql> insert into Company values("Samba Bank","Bangaluru"),("NCB Bank","Kolkata"),("ICICI Bank","Lucknow"),("HDFC Bank","Ahemdabad"),("Reseve Bank","Delhi");`

Query OK, 5 rows affected (0.01 sec)

Records: 5 Duplicates: 0 Warnings: 0

`mysql> create table Employee(Person_Name varchar(20),Street varchar(10),City varchar(10),Primary key(Person_Name));`

Query OK, 0 rows affected (0.05 sec)

`mysql> insert into Employee values("Ramesh","abc","Ahemdabad"),("Suresh","xyz","Jaipur"),("Mahesh","def","Kolkata"),("Nina","kgf","Lucknow"),("Sunanda","qwe","Chennai");`

Query OK, 5 rows affected (0.02 sec)

Records: 5 Duplicates: 0 Warnings: 0

`mysql> create table Works(Person_Name varchar(20),Company_Name varchar(20),Salary int,Foreign key(Person_Name) references Employee(Person_Name),Foreign key(Company_Name) references Company(Company_Name));`

Query OK, 0 rows affected (0.09 sec)

`mysql> insert into Works values("Ramesh","Samba Bank",50000);`

Query OK, 1 row affected (0.01 sec)

`mysql> insert into Works values("Suresh","HDFC Bank",1200000);`

Query OK, 1 row affected (0.01 sec)

`mysql> insert into Works values("Mahesh","NCB Bank",60000);`

Query OK, 1 row affected (0.01 sec)

```
mysql> insert into Works values("Nina","ICICI Bank",70000);
```

Query OK, 1 row affected (0.01 sec)

```
mysql> insert into Works values("Rashi", "Reserve Bank",80000);
```

Query OK, 1 row affected (0.03 sec)

```
mysql> create table Manages(Person_Name varchar(20),Manager_Name varchar(20),Foreign
key(Person_Name) references Employee(Person_Name),Foreign key(Manager_Name) references
Employee(Person_Name));
```

Query OK, 0 rows affected (0.048 sec)

```
mysql> insert into manages
values("Sunanda","Aakash"),("Nina","Akhilesh"),("Mahesh","Khushi"),("Ramesh","Kishore"),("Suresh",
"Nandini");
```

Query OK, 5 rows affected (0.01 sec)

Records: 5 Duplicates: 0 Warnings: 0

```
mysql> select * from Employee;
+-----+-----+-----+
| Person_Name | Street | City |
+-----+-----+-----+
| Mahesh      | def    | Kolkata |
| Nina        | kgf    | Lucknow |
| Ramesh      | abc    | Ahemdabad |
| Sunanda     | qwe    | Chennai |
| Suresh      | xyz    | Jaipur  |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select * from Works;
+-----+-----+-----+
| Person_Name | Company_Name | Salary |
+-----+-----+-----+
| Ramesh      | Samba Bank   | 50000  |
| Suresh      | HDFC Bank    | 120000 |
| Mahesh      | NCB Bank     | 60000  |
| Nina        | ICICI Bank   | 70000  |
| Sunanda     | Reserve Bank | 80000  |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select * from Company;
+-----+-----+
| Company_Name | City |
+-----+-----+
| HDFC Bank    | Ahemdabad |
| ICICI Bank   | Lucknow   |
| NCB Bank     | Kolkata   |
| Reseve Bank  | Delhi     |
| Samba Bank   | Bangaluru |
+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select * from manages;
+-----+-----+
| Person_Name | Manager_Name |
+-----+-----+
| Sunanda     | Aakash       |
| Nina        | Akhilesh     |
| Mahesh      | khushi       |
| Ramesh      | kishore      |
| Suresh      | Nandini      |
+-----+-----+
5 rows in set (0.00 sec)
```

b)

```
mysql> alter table Employee add email varchar(20);
```

Query OK, 0 rows affected (0.06 sec)

Records: 0 Duplicates: 0 Warnings: 0

c) mysql> select m.Manager_Name,m.Person_Name from Manages m join works w on m.Person_Name=w.Person_Name where w.Company_Name="Samba Bank" or w.Company_Name="NCB Bank";

```
+-----+-----+
| Manager_Name | Person_Name |
+-----+-----+
| kishore      | Ramesh      |
| khushi       | Mahesh      |
+-----+-----+
```

4 rows in set (0.00 sec)

d) mysql> select e.Person_Name,e.Street,e.City,w.Salary from Employee e join works w on e.Person_Name=w.Person_Name where w.Company_Name="Samba Bank" and w.Salary>10000;

```
+-----+-----+-----+-----+
| Person_Name | Street | City      | Salary |
+-----+-----+-----+-----+
| Ramesh      | abc    | Ahemdabad | 50000  |
+-----+-----+-----+-----+
```

1 row in set (0.00 sec)

e) mysql> select e.Person_Name,c.City from Employee e join works w on e.Person_Name=w.Person_Name join Company c on w.Company_Name=c.Company_Name where e.City=c.City;

```
+-----+-----+
| Person_Name | City      |
+-----+-----+
| Mahesh      | Kolkata   |
| Nina        | Lucknow   |
+-----+-----+
```

2 rows in set (0.00 sec)

f) mysql> select Company_Name,max(Salary),min(Salary),avg(Salary) from works group by Company_Name;

Company_Name	max(Salary)	min(Salary)	avg(Salary)
Samba Bank	50000	50000	50000.0000
HDFC Bank	1200000.	1200000	1200000.0000
NCB Bank	60000	60000	60000.0000
ICICI Bank	70000	70000	70000.0000.
Reserve Bank	80000	80000	80000.0000

5 rows in set (0.01 sec)

g) mysql> select Company_Name,max(Salary),min(Salary),avg(Salary) from works group by Company_Name;

Company_Name	max(Salary)	min(Salary)	avg(Salary).
Samba Bank	50000	50000	50000.0000
HDFC Bank	1200000	1200000	1200000.0000
NCB Bank	60000	60000	60000.0000
ICICI Bank.	70000	70000.	70000.0000
Reserve Bank.	80000	80000.	80000.0000.

5 rows in set (0.01 sec)

h) mysql> select Company_Name,max(Salary)from works;

Company_Name	max(Salary)
Samba Bank	1200000

1 row in set (0.01 sec)

Ques 6: a) mysql> create table Suppliers(Sno int primary key,Sname varchar(10),Status int,SCity varchar(20));

Query OK, 0 rows affected (0.05 sec)

mysql> insert into Suppliers
values(001,"S1",67,"Paris"),(002,"S2",12,"Jaipur"),(003,"S3",25,"Chennai"),(004,"S4",30,"Ahemdabad"),(005,"S5",70,"Delhi");

Query OK, 5 rows affected (0.03 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> create table Parts(Pno int primary key,Pname varchar(10),Colour varchar(10),Weight int,City varchar(20));

Query OK, 0 rows affected (0.06 sec)

mysql> insert into Parts
Values(100,"P1","Red",15,"Paris"),(200,"P2","Blue",10,"Jaipur"),(300,"P3","Red",20,"London"),(400,"P4","Black",25,"Paris"),(500,"P5","Blue",30,"Japan");

Query OK, 5 rows affected (0.01 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> create table Project(Jno int primary key, Jname varchar(10),JCity varchar(20));

Query OK, 0 rows affected (0.05 sec)

mysql> insert into Project
values(101,"J1","Paris"),(102,"J2","London"),(103,"J3","Paris"),(104,"J4","Jaipur"),(105,"J5","London");

Query OK, 5 rows affected (0.01 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> create table Shipment(Sno int,Pno int,Jno int,Quantity int,foreign key (Sno) references Suppliers(Sno),foreign key (Pno) references Parts(Pno),foreign key(Jno) references Project(Jno));

Query OK, 0 rows affected (0.08 sec)

mysql> insert into Shipment
values(001,100,101,300),(002,200,102,750),(003,300,103,600),(004,400,104,250),(005,500,105,700);

Query OK, 5 rows affected (0.03 sec)

```
mysql> select * from Suppliers;
+-----+-----+-----+-----+
| Sno | Sname | Status | SCity |
+-----+-----+-----+-----+
| 1 | S1 | 27 | Paris |
| 2 | S2 | 12 | Jaipur |
| 3 | S3 | 25 | Chennai |
| 4 | S4 | 30 | Ahemdabad |
| 5 | S5 | 70 | Delhi |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select * from Parts;
+-----+-----+-----+-----+-----+
| Pno | Pname | Colour | Weight | City |
+-----+-----+-----+-----+-----+
| 100 | P1 | Red | 15 | Paris |
| 200 | P2 | Blue | 10 | Jaipur |
| 300 | P3 | Red | 20 | London |
| 400 | P4 | Black | 25 | Paris |
| 500 | P5 | Blue | 30 | Japan |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select * from Project;
+-----+-----+-----+
| Jno | Jname | JCity |
+-----+-----+-----+
| 101 | J1 | Paris |
| 102 | J2 | London |
| 103 | J3 | Paris |
| 104 | J4 | Jaipur |
| 105 | J5 | London |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select * from shipment;
+-----+-----+-----+-----+
| Sno | Pno | Jno | Quantity |
+-----+-----+-----+-----+
| 1 | 100 | 101 | 300 |
| 2 | 200 | 102 | 750 |
| 3 | 300 | 103 | 600 |
| 4 | 400 | 104 | 250 |
| 5 | 500 | 105 | 700 |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

b) mysql> select Sno, Status, Scity from Suppliers where SCity="Paris" and Status>20;

```
+-----+-----+-----+
| Sno | Status | Scity |
+-----+-----+-----+
| 1 | 27 | Paris |
+-----+-----+-----+
```

1 row in set (0.00 sec)

c) mysql> select suppliers.Sno, Suppliers.Sname from Suppliers, Parts, Shipment where Parts.Pname="P2" and Suppliers.Sno=Shipment.Sno and Parts.Pno=Shipment.Pno group by Sno;

```
+-----+-----+
| Sno | Sname |
+-----+-----+
| 2 | S2 |
+-----+-----+
```

1 row in set (0.00 sec)

d) `mysql> select Suppliers.Sno,Suppliers.Sname from Suppliers,Parts,Shipment where Suppliers.Sno=Shipment.Sno and Parts.Pno= Shipment.Pno and not Parts.Pname="P2";`

```
+-----+-----+
| Sno | Sname |
+-----+-----+
| 1   | S1    |
| 3   | S3    |
| 4   | S4    |
| 5   | S5    |
+-----+-----+
```

4 rows in set (0.00 sec)

e) `mysql> select sno,Parts.Pno, Jno, Parts.Weight*Shipment.Quantity as 'Total_Shipment' from Shipment,Parts where Parts.Pno=Shipment.Pno;`

```
+-----+-----+-----+-----+
| sno | Pno | Jno | Total_Shipment |
+-----+-----+-----+-----+
| 1   | 100 | 101 | 4500           |
| 2   | 200 | 102 | 7500           |
| 3   | 300 | 103 | 12000          |
| 4   | 400 | 104 | 6250           |
| 5   | 500 | 105 | 21000          |
+-----+-----+-----+-----+
```

5 rows in set (0.01 sec)

f) `mysql> select * from Shipment where Quantity between 300 and 750;`

```
+-----+-----+-----+-----+
| Sno | Pno | Jno | Quantity |
+-----+-----+-----+-----+
| 1   | 100 | 101 | 300      |
| 2   | 200 | 102 | 750      |
| 3   | 300 | 103 | 600      |
| 5   | 500 | 105 | 700      |
+-----+-----+-----+-----+
```

4 rows in set (0.00 sec)

g) mysql> select distinct Pno from Parts, Suppliers where Parts.Weight > 16 or Suppliers.Sname='S2';

```
+-----+
```

```
| Pno |
```

```
+-----+
```

```
| 500 |
```

```
| 400 |
```

```
| 300 |
```

```
| 200 |
```

```
| 100 |
```

```
+-----+
```

5 rows in set (0.00 sec)

h) mysql> select Pname, City, Colour from Parts, Shipment where Parts.Pno= Shipment.Pno and Parts.Colour="Red" and Shipment.Quantity > 5;

```
+-----+-----+-----+
```

```
| Pname | City    | Colour |
```

```
+-----+-----+-----+
```

```
| P1    | Paris   | Red    |
```

```
| P3    | London  | Red    |
```

```
+-----+-----+-----+
```

2 rows in set (0.00 sec)

i) mysql> select Parts.Pno, Pname, Colour, City, Weight from Parts, Shipment, Suppliers where Suppliers.Sno= Shipment.Sno and Parts.Pno= Shipment.Pno and Suppliers.SCity="Delhi";

```
+-----+-----+-----+-----+-----+
```

```
| Pno | Pname | Colour | City | Weight |
```

```
+-----+-----+-----+-----+-----+
```

```
| 500 | P5    | Blue   | Delhi | 30    |
```

```
+-----+-----+-----+-----+-----+
```

1 row in set (0.01 sec)

j) mysql> select Parts.Pno from Parts, Suppliers, Shipment, Project where Parts.City="Chennai" and Suppliers.SCity="Allahabad" and Suppliers.Sno=Shipment.Sno and Parts.Pno= Shipment.Pno and Project.Jno= Shipment.Jno;

Empty set (0.01 sec)

k) mysql> select count(Project.Jno) from Suppliers, Project, Shipment where Suppliers.Sno= Shipment.Sno and Project.Jno= Shipment.Jno and Suppliers.Sname="S1" group by Project.Jno;

```
+-----+
| count(Project.Jno) |
+-----+
|                1 |
+-----+
```

1 row in set (0.00 sec)

l) mysql> select sum(Quantity) from Suppliers, Parts, Shipment where Suppliers.Sname="S1" and Parts.Pname="P1" group by Suppliers.Sno;

```
+-----+
| sum(Quantity) |
+-----+
|           2600 |
+-----+
```

1 row in set (0.00 sec)

7. Inserting Data

To insert data into MongoDB we use the insert() method which is available in the database environment. First we connect to the db using python code shown below and then we provide the document details in form of a series of key-value pairs.

Import the python libraries

```
from pymongo import MongoClient
```

```
from pprint import pprint
```

Choose the appropriate client

```
client = MongoClient()
```

Connect to the test db

```
db=client.test
```

Use the employee collection

```

employee = db.employee
employee_details = {
    'Name': 'Raj Kumar',
    'Address': 'Sears Streer, NZ',
    'Age': '42'
}

```

Use the insert method

```
result = employee.insert_one(employee_details)
```

Query for the inserted document.

```

Queryresult = employee.find_one({'Age': '42'})
pprint(Queryresult)

```

When we execute the above code, it produces the following result.

```

{'u'Address': u'Sears Streer, NZ',
  u'Age': u'42',
  u'Name': u'Raj Kumar',
  u'_id': ObjectId('5adc5a9f84e7cd3940399f93')}

```

Updating Data

Updating an existing MongoDB data is similar to inserting. We use the update() method which is native to mongoDB. In the below code we are replacing the existing record with new key-value pairs. Please note how we are using the condition criteria to decide which record to update.

Import the python libraries

```
from pymongo import MongoClient
```

```
from pprint import pprint
```

Choose the appropriate client

```
client = MongoClient()
```

Connect to db

```
db=client.test
```

```
employee = db.employee
```

```
# Use the condition to choose the record
```

```
# and use the update method
```

```
db.employee.update_one(  
    {"Age":'42'},  
    {  
        "$set": {  
            "Name": "Srinidhi",  
            "Age": '35',  
            "Address": "New Omsk, WC"  
        }  
    }  
)
```

```
Queryresult = employee.find_one({'Age': '35'})
```

```
pprint(Queryresult)
```

When we execute the above code, it produces the following result.

```
{u'Address': u'New Omsk, WC',  
u'Age': u'35',  
u'Name': u'Srinidhi',  
u'_id': ObjectId('5adc5a9f84e7cd3940399f93')}
```

Deleting Data

Deleting a record is also straight forward where we use the delete method. Here also we mention the condition which is used to choose the record to be deleted.

```
# Import the python libraries
```

```
from pymongo import MongoClient
```

```
from pprint import pprint
```

```
# Choose the appropriate client
```

```
client = MongoClient()
```



```
# Connect to db

db=client.test

employee = db.employee

# Use the condition to choose the record

# and use the delete method

db.employee.delete_one({"Name":"Prajakta"})
```

```
Queryresult = employee.find_one({"Name":"Prajakta"})
```

```
pprint(Queryresult)
```

When we execute the above code, it produces the following result.

None

So we see the particular record does not exist in the db any more.

```
myQuery = { "COMPANY": { "$gt": "R" } }
```

```
val = tab.find(myQuery)
```

```
for x in val:
```

```
    print(x)
```

```
tab = mydb["customers"]
```

```
for x in tab.find():
```

```
    print(x)
```

CHAPTER 9: Introduction to Machine Learning

1.

```
import pandas as pd
df = pd.read_csv('mtcars.csv')
print(df.head,10)
print(df.shape)
#Check Outliers
import seaborn as sns
sns.boxplot(data=df,x=df['hp'])

Q1=df['hp'].quantile(0.25)Q3=df['hp'].quantile(0.75)IQR=Q3-
Q1print(Q1)print(Q3)print(IQR)Lower_Whisker = Q1-1.5*IQRUpper_Whisker =
Q3+1.5*IQRprint(Lower_Whisker, Upper_Whisker)
df = df[df['hp']< Upper_Whisker]
print(df.shape)
```

2.

```
import pandas as pd
df = pd.read_csv('car-mpg.csv')
print(df.head,10)
print(df.shape)
from scipy import stats
z=np.abs(stats.zscore(df.hp))print(z)
threshold=3print(np.where(z>3))
df1=df[(z< 3)]print(df1)
```

3.

```
import numpy as np

data = [1, 2, 2, 2, 3, 1, 1, 15, 2, 2, 2, 3, 1, 1, 2]
mean = np.mean(data)
std = np.std(data)
print('mean of the dataset is', mean)
print('std. deviation is', std)
```

```

threshold = 3
outlier = []
for i in data:
    z = (i-mean)/std
    if z > threshold:
        outlier.append(i)
print('outlier in dataset is', outlier)

```

4.

```

# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Reading the data
df = pd.read_csv("data_out.csv")
print(df.shape)
print(df.info())
df.describe()
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR)) print(df['Income'].skew())
df['Income'].describe()
plt.boxplot(df["Loan_amount"])
plt.show()

df.boxplot(column='Loan_amount', by='approval_status')
df.Income.hist()

fig, ax = plt.subplots(figsize=(12,6))
ax.scatter(df['Income'], df['Loan_amount'])
ax.set_xlabel('Income of applicants in USD')
ax.set_ylabel('Loan amount applied for in USD')
plt.show()

print(df['Income'].quantile(0.10))
print(df['Income'].quantile(0.90))

```

```

df["Income"] = np.where(df["Income"] < 2960.0, 2960.0, df['Income'])
df["Income"] = np.where(df["Income"] > 12681.0, 12681.0, df['Income'])
print(df['Income'].skew())
index = df[(df['Age'] >= 100) | (df['Age'] <= 18)].index
df.drop(index, inplace=True)
df['Age'].describe()
df_out = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axis=1)]
print(df_out.shape)
df["Log_Loanamt"] = df["Loan_amount"].map(lambda i: np.log(i) if i > 0 else 0)
print(df['Loan_amount'].skew())
print(df['Log_Loanamt'].skew())
print(df['Loan_amount'].quantile(0.50))
print(df['Loan_amount'].quantile(0.95))
df['Loan_amount'] = np.where(df['Loan_amount'] > 325, 140, df['Loan_amount'])
df.describe()

```

5.

```

from sklearn.preprocessing import MinMaxScaler
x = baseball[['age', 'height_in', 'weight_lb']]
mms = MinMaxScaler()
xn = mms.fit_transform(x)
print(xn.min(axis=0))
print(xn.max(axis=0))
print(xn.std(axis=0))

```

6.

```

from pandas import read_csv
from numpy import set_printoptions
from sklearn import preprocessing
path = r'C:\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(path, names=names)
array = dataframe.values
data_scaler = preprocessing.MinMaxScaler(feature_range=(0,1))
data_rescaled = data_scaler.fit_transform(array)
print("\nScaled data:\n", data_rescaled[0:10])

```

7.

```

from sklearn.preprocessing import StandardScaler
from pandas import read_csv
from numpy import set_printoptions
path = r'C:\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(path, names=names)
array = dataframe.values

Now, we can use StandardScaler class to rescale the data.

data_scaler = StandardScaler().fit(array)
data_rescaled = data_scaler.transform(array)
set_printoptions(precision=2)
print ("\nRescaled data:\n", data_rescaled [0:5])

```

8.

```

from sklearn.datasets import load_boston
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
boston = load_boston()
print(boston)
x = boston.data
y = boston.target
columns = boston.feature_names#create the dataframe
boston_df = pd.DataFrame(boston.data)
fig, ax = plt.subplots(figsize=(16,8))
ax.scatter(boston_df['INDUS'], boston_df['TAX'])
ax.set_xlabel('Proportion of non-retail business acres per town')
ax.set_ylabel('Full-value property-tax rate per $10,000')
plt.show()

```

CHAPTER 10: Regression Analysis in Machine Learning

1.

```
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)

model.fit(x[:, np.newaxis], y)

xfit = np.linspace(0, 10, 1000)
yfit = model.predict(xfit[:, np.newaxis])

plt.scatter(x, y)
plt.plot(xfit, yfit);
print("Model slope: ", model.coef_[0])
print("Model intercept:", model.intercept_)
```

2.

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

#importing datasets
data_set= pd.read_csv('Position_Salaries.csv')

#Extracting Independent and dependent Variable
x= data_set.iloc[:, 1:2].values
y= data_set.iloc[:, 2].values

#Fitting the Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin_regs= LinearRegression()
lin_regs.fit(x,y)

#Fitting the Polynomial regression to the dataset
```

Copyright © 2023 by McGraw Hill Education (India) Private Limited

```

from sklearn.preprocessing import PolynomialFeatures

poly_regs= PolynomialFeatures(degree= 2)
x_poly= poly_regs.fit_transform(x)
lin_reg_2 =LinearRegression()
lin_reg_2.fit(x_poly, y)
#Visulaizing the result for Linear Regression model
mtp.scatter(x,y,color="blue")
mtp.plot(x,lin_regs.predict(x), color="red")
mtp.title("Bluff detection model(Linear Regression)")
mtp.xlabel("Position Levels")
mtp.ylabel("Salary")
mtp.show()
#Visulaizing the result for Polynomial Regression
mtp.scatter(x,y,color="blue")
mtp.plot(x, lin_reg_2.predict(poly_regs.fit_transform(x)), color="red")
mtp.title("Bluff detection model(Polynomial Regression)")
mtp.xlabel("Position Levels")
mtp.ylabel("Salary")
mtp.show()
in_pred = lin_regs.predict([[6.5]])
print(lin_pred)
poly_pred = lin_reg_2.predict(poly_regs.fit_transform([[6.5]]))
print(poly_pred)

```

3.

```

# importing libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# for calculating mean_squared error
from sklearn.metrics import mean_squared_error

# creating a dataset with curvilinear relationship

```

```
x=10*np.random.normal(0,1,70)
y=10*(-x**2)+np.random.normal(-100,100,70)

# plotting dataset
plt.figure(figsize=(10,5))
plt.scatter(x,y,s=15)
plt.xlabel('Predictor',fontsize=16)
plt.ylabel('Target',fontsize=16)
plt.show()

# Importing Linear Regression
from sklearn.linear_model import LinearRegression

# Training Model
lm=LinearRegression()
lm.fit(x.reshape(-1,1),y.reshape(-1,1))
Let's see how linear regression performs on this dataset:
y_pred=lm.predict(x.reshape(-1,1))

# plotting predictions
plt.figure(figsize=(10,5))
plt.scatter(x,y,s=15)
plt.plot(x,y_pred,color='r')
plt.xlabel('Predictor',fontsize=16)
plt.ylabel('Target',fontsize=16)
plt.show()

# importing libraries for polynomial transform
from sklearn.preprocessing import PolynomialFeatures

# for creating pipeline
from sklearn.pipeline import Pipeline

# creating pipeline and fitting it on data
Input=[('polynomial',PolynomialFeatures(degree=2)),('model',LinearRegression())]
pipe=Pipeline(Input)

pipe.fit(x.reshape(-1,1),y.reshape(-1,1)) poly_pred=pipe.predict(x.reshape(-1,1))
```



```

#sorting predicted values with respect to predictor
sorted_zip = sorted(zip(x,poly_pred))
x_poly, poly_pred = zip(*sorted_zip)
#plotting predictions
plt.figure(figsize=(10,6))
plt.scatter(x,y,s=15)
plt.plot(x,y_pred,color='r',label='Linear Regression')
plt.plot(x_poly,poly_pred,color='g',label='Polynomial Regression')
plt.xlabel('Predictor',fontsize=16)
plt.ylabel('Target',fontsize=16)
plt.legend()
plt.show()
print('RMSE for Polynomial Regression=>',np.sqrt(mean_squared_error(y,poly_pred)))

```

4.

```

from sklearn.preprocessing import PolynomialFeatures

def create_polynomial_regression_model(degree):
    "Creates a polynomial regression model for the given degree"

    poly_features = PolynomialFeatures(degree=degree)

    # transforms the existing features to higher degree features.
    X_train_poly = poly_features.fit_transform(X_train)

    # fit the transformed features to Linear Regression
    poly_model = LinearRegression()
    poly_model.fit(X_train_poly, Y_train)

    # predicting on training data-set
    y_train_predicted = poly_model.predict(X_train_poly)

    # predicting on test data-set
    y_test_predict = poly_model.predict(poly_features.fit_transform(X_test))

```

```

# evaluating the model on training dataset

rmse_train = np.sqrt(mean_squared_error(Y_train, y_train_predicted))
r2_train = r2_score(Y_train, y_train_predicted)

# evaluating the model on test dataset

rmse_test = np.sqrt(mean_squared_error(Y_test, y_test_predict))
r2_test = r2_score(Y_test, y_test_predict)

print("The model performance for the training set")
print("-----")
print("RMSE of training set is {}".format(rmse_train))
print("R2 score of training set is {}".format(r2_train))

print("\n")

print("The model performance for the test set")
print("-----")
print("RMSE of test set is {}".format(rmse_test))
print("R2 score of test set is {}".format(r2_test))

```

5.

```

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
import matplotlib

matplotlib.rcParams.update({'font.size': 12})

from sklearn.datasets import load_boston
#from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge

boston=load_boston()

boston_df=pd.DataFrame(boston.data,columns=boston.feature_names)

#print boston_df.info()# add another column that contains the house prices which in scikit
learn datasets are considered as target

```

```

boston_df['Price']=boston.target
#print boston_df.head(3)
newX=boston_df.drop('Price',axis=1)
print(newX[0:3]) # check
newY=boston_df['Price']#print type(newY)# pandas core frame
X_train,X_test,y_train,y_test=train_test_split(newX,newY,test_size=0.3,random_state=3)
print(len(X_test), len(y_test))
lr = LinearRegression()
lr.fit(X_train, y_train)
rr = Ridge(alpha=0.01) # higher the alpha value, more restriction on the coefficients; low alpha
> more generalization,
# in this case linear and ridge regression resembles
rr.fit(X_train, y_train)
rr100 = Ridge(alpha=100) # comparison with alpha value
rr100.fit(X_train, y_train)
train_score=lr.score(X_train, y_train)
test_score=lr.score(X_test, y_test)
Ridge_train_score = rr.score(X_train,y_train)
Ridge_test_score = rr.score(X_test, y_test)
Ridge_train_score100 = rr100.score(X_train,y_train)
Ridge_test_score100 = rr100.score(X_test, y_test)
plt.plot(rr.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge;
$\alpha = 0.01$',zorder=7)
plt.plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'Rid
ge; $\alpha = 100$')
plt.plot(lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear
Regression')
plt.xlabel('Coefficient Index',fontsize=16)
plt.ylabel('Coefficient Magnitude',fontsize=16)
plt.legend(fontsize=13,loc=4)
plt.show()

```

6.

```

from numpy import mean
from numpy import std
from numpy import absolute

```

```

from pandas import read_csv

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
from sklearn.linear_model import Lasso

# load the dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'
dataframe = read_csv(url, header=None)
data = dataframe.values
X, y = data[:, :-1], data[:, -1]

# define model
model = Lasso(alpha=1.0)

# define model evaluation method
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)

# evaluate model
scores = cross_val_score(model, X, y, scoring='neg_mean_absolute_error', cv=cv, n_jobs=-1)

# force scores to be positive
scores = absolute(scores)
print('Mean MAE: %.3f (%.3f)' % (mean(scores), std(scores)))

```

7.

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Loading pre-defined Boston Dataset
boston_dataset = datasets.load_boston()
print(boston_dataset.DESCR)

# Load the dataset into Pandas Dataframe
boston_pd = pd.DataFrame(boston_dataset.data)
boston_pd.columns = boston_dataset.feature_names
boston_pd_target = np.asarray(boston_dataset.target)
boston_pd['House Price'] = pd.Series(boston_pd_target)

```

```

# input
X = boston_pd.iloc[:, :-1]

#output
Y = boston_pd.iloc[:, -1]

print(boston_pd.head())
x_train, x_test, y_train, y_test = train_test_split(
    boston_pd.iloc[:, :-1], boston_pd.iloc[:, -1],
    test_size = 0.25)

print("Train data shape of X = % s and Y = % s :"%(
    x_train.shape, y_train.shape))

print("Test data shape of X = % s and Y = % s :"%(
    x_test.shape, y_test.shape))

# Apply multiple Linear Regression Model
lreg = LinearRegression()
lreg.fit(x_train, y_train)

# Generate Prediction on test set
lreg_y_pred = lreg.predict(x_test)

# calculating Mean Squared Error (mse)
mean_squared_error = np.mean((lreg_y_pred - y_test)**2)
print("Mean squared Error on test set : ", mean_squared_error)

# Putting together the coefficient and their corresponding variable names
lreg_coefficient = pd.DataFrame()
lreg_coefficient["Columns"] = x_train.columns
lreg_coefficient['Coefficient Estimate'] = pd.Series(lreg.coef_)
print(lreg_coefficient)

```

8.

```
import pandas as pd
import numpy as np
from sklearn import model_selection
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
df = pd.read_csv('economics.csv')
print(df.shape)
print(df.describe())
target_column = ['unemploy']
predictors = list(set(list(df.columns))-set(target_column))
print(predictors)
#df[predictors] = float(df[predictors])/float(df[predictors].max())
df.describe()
```

```
X = df[predictors].values
```

```
y = df[target_column].values
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=40)
```

```
print(X_train.shape); print(X_test.shape)
```

```
lr = LinearRegression()
```

```
lr.fit(X_train, y_train)
```

```
pred_train_lr= lr.predict(X_train)
```

```
print(np.sqrt(mean_squared_error(y_train,pred_train_lr)))
```

```
print(r2_score(y_train, pred_train_lr))

pred_test_lr= lr.predict(X_test)
print(np.sqrt(mean_squared_error(y_test,pred_test_lr)))
print(r2_score(y_test, pred_test_lr))
```

9.

```
from pandas import read_csv
from sklearn.linear_model import Lasso
# load the dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'
dataframe = read_csv(url, header=None)
data = dataframe.values
X, y = data[:, :-1], data[:, -1]
# define model
model = Lasso(alpha=1.0)
# fit model
model.fit(X, y)
# define new data
row = [0.00632,18.00,2.310,0,0.5380,6.5750,65.20,4.0900,1,296.0,15.30,396.90,4.98]
# make a prediction
yhat = model.predict([row])
# summarize prediction
print('Predicted: %.3f' % yhat)
```

10.

```
from numpy import arange
from pandas import read_csv
from sklearn.linear_model import LassoCV
from sklearn.model_selection import RepeatedKFold
# load the dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'
dataframe = read_csv(url, header=None)
data = dataframe.values
X, y = data[:, :-1], data[:, -1]
```

```
# define model evaluation method
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
# define model
model = LassoCV(alphas=arange(0, 1, 0.01), cv=cv, n_jobs=-1)
# fit model
model.fit(X, y)
# summarize chosen configuration
print('alpha: %f' % model.alpha_)
```

11.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Loading pre-defined Boston Dataset
boston_dataset = datasets.load_boston()
print(boston_dataset.DESCR)
# Load the dataset into Pandas Dataframe
boston_pd = pd.DataFrame(boston_dataset.data)
boston_pd.columns = boston_dataset.feature_names
boston_pd_target = np.asarray(boston_dataset.target)
boston_pd['House Price'] = pd.Series(boston_pd_target)

# input
X = boston_pd.iloc[:, :-1]

#output
Y = boston_pd.iloc[:, -1]

print(boston_pd.head())
x_train, x_test, y_train, y_test = train_test_split(
    boston_pd.iloc[:, :-1], boston_pd.iloc[:, -1],
    test_size = 0.25)
```



```

print("Train data shape of X = % s and Y = % s :"%(
    x_train.shape, y_train.shape))

print("Test data shape of X = % s and Y = % s :"%(
    x_test.shape, y_test.shape))

# import model
from sklearn.linear_model import ElasticNet

# Train the model
e_net = ElasticNet(alpha = 1)
e_net.fit(x_train, y_train)

# calculate the prediction and mean square error
y_pred_elastic = e_net.predict(x_test)
mean_squared_error = np.mean((y_pred_elastic - y_test)**2)
print("Mean Squared Error on test set", mean_squared_error)

e_net_coeff = pd.DataFrame()
e_net_coeff["Columns"] = x_train.columns
e_net_coeff["Coefficient Estimate"] = pd.Series(e_net.coef_)
print(e_net_coeff )

```

12.

```

import time

import matplotlib.pyplot as plt

import numpy as np

from sklearn.datasets import fetch_openml
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.utils import check_random_state

```

```

t0 = time.time()

train_samples = 5000

# Load data from https://www.openml.org/d/554
X, y = fetch_openml('mnist_784', version=1, return_X_y=True, as_frame=False)

random_state = check_random_state(0)
permutation = random_state.permutation(X.shape[0])
X = X[permutation]
y = y[permutation]
X = X.reshape((X.shape[0], -1))

X_train, X_test, y_train, y_test = train_test_split(
    X, y, train_size=train_samples, test_size=10000)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Turn up tolerance for faster convergence
clf = LogisticRegression(
    C=50. / train_samples, penalty='l1', solver='saga', tol=0.1
)
clf.fit(X_train, y_train)

sparsity = np.mean(clf.coef_ == 0) * 100
score = clf.score(X_test, y_test)
# print('Best C %.4f' % clf.C_)
print("Sparsity with L1 penalty: %.2f%%" % sparsity)
print("Test score with L1 penalty: %.4f" % score)

coef = clf.coef_.copy()
plt.figure(figsize=(10, 5))
scale = np.abs(coef).max()

for i in range(10):

```

```

l1_plot = plt.subplot(2, 5, i + 1)
l1_plot.imshow(coef[i].reshape(28, 28), interpolation='nearest',
               cmap=plt.cm.RdBu, vmin=-scale, vmax=scale)
l1_plot.set_xticks(())
l1_plot.set_yticks(())
l1_plot.set_xlabel('Class %i' % i)
plt.suptitle('Classification vector for...')

run_time = time.time() - t0
print('Example run in %.3f s' % run_time)
plt.show()

```

13.

```

import pandas as pd
import numpy as np
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
df = pd.read_csv('Student-Pass-Fail-Data.csv')
x = df.drop('Pass_Or_Fail', axis = 1)
y = df.Pass_Or_Fail
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=4)
logistic_regression = LogisticRegression()
logistic_regression.fit(x_train, y_train)
y_pred = logistic_regression.predict(x_test)
accuracy = metrics.accuracy_score(y_test, y_pred)
accuracy_percentage = 100 * accuracy
print(accuracy_percentage)

```

14.

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

Loading dataset – User_Data
dataset = pd.read_csv('User_Data.csv')

```

```
# input
x = dataset.iloc[:, [2, 3]].values

# output
y = dataset.iloc[:, 4].values
from sklearn.cross_validation import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(
    x, y, test_size = 0.25, random_state = 0)
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(xtrain)
xtest = sc_x.transform(xtest)

print (xtrain[0:10, :])
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(xtrain, ytrain)
y_pred = classifier.predict(xtest)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(ytest, y_pred)

print ("Confusion Matrix : \n", cm)
```

15.

```
import pandas
from sklearn import linear_model

df = pandas.read_csv("cars.csv")

X = df[['Weight', 'Volume']]
y = df['CO2']

regr = linear_model.LinearRegression()
regr.fit(X, y)

#predict the CO2 emission of a car where the weight is 2300kg, and the volume is 1300cm3:
predictedCO2 = regr.predict([[2300, 1300]])
print(predictedCO2)

print(regr.coef_)
```

16.

```

import pandas
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()

df = pandas.read_csv("cars2.csv")

X = df[['Weight', 'Volume']]
y = df['CO2']

scaledX = scale.fit_transform(X)

regr = linear_model.LinearRegression()
regr.fit(scaledX, y)

scaled = scale.transform([[2300, 1.3]])

predictedCO2 = regr.predict([scaled[0]])
print(predictedCO2)

```

17.

```

import numpy as np
from sklearn.linear_model import LinearRegression

x = [[0, 1], [5, 1], [15, 2], [25, 5], [35, 11], [45, 15], [55, 34], [60, 35]]
y = [4, 5, 20, 14, 32, 22, 38, 43]
x, y = np.array(x), np.array(y)

>>> print(x)
>>> print(y)

model = LinearRegression().fit(x, y)

>>> r_sq = model.score(x, y)
>>> print('coefficient of determination:', r_sq)
coefficient of determination: 0.8615939258756776
>>> print('intercept:', model.intercept_)
intercept: 5.52257927519819
>>> print('slope:', model.coef_)
>>> y_pred = model.predict(x)
>>> print('predicted response:', y_pred, sep='\n')
>>> y_pred = model.intercept_ + np.sum(model.coef_ * x, axis=1)

```

```
>>> print('predicted response:', y_pred, sep='\n')
>>> x_new = np.arange(10).reshape((-1, 2))
>>> print(x_new)
>>> y_new = model.predict(x_new)
>>> print(y_new)
```

18.

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
y = np.array([15, 11, 2, 8, 25, 32])
# transform the input array x to contain the additional column(s) with the values of  $x^2$  (and
eventually more features).
transformer = PolynomialFeatures(degree=2, include_bias=False)

# Before applying transformer, you need to fit it with .fit():
transformer.fit(x)
x_ = transformer.transform(x)
model = LinearRegression().fit(x_, y)
r_sq = model.score(x_, y)
print('coefficient of determination:', r_sq)
print('intercept:', model.intercept_)
print('coefficients:', model.coef_)
y_pred = model.predict(x_)
print('predicted response:', y_pred, sep='\n')
```

CHAPTER 11: Classification and Clustering

1.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import seaborn as sns
sns.set()
breast_cancer = load_breast_cancer()
X = pd.DataFrame(breast_cancer.data, columns=breast_cancer.feature_names)
X = X[['mean area', 'mean compactness']]
y = pd.Categorical.from_codes(breast_cancer.target, breast_cancer.target_names)
y = pd.get_dummies(y, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print(confusion_matrix(y_test, y_pred))
```

2.

```
#Import scikit-learn dataset library
from sklearn import datasets

#Load dataset
wine = datasets.load_wine()
# print the names of the features
print(wine.feature_names)
# print the label species(class_0, class_1, class_2)
print(wine.target_names)
```

Copyright © 2023 by McGraw Hill Education (India) Private Limited

```
# print the wine labels (0:Class_0, 1:Class_1, 2:Class_3)
print(wine.target)
# print data(feature)shape
print(wine.data.shape)
# print target(or label)shape
print(wine.target.shape)
# Import train_test_split function
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target, test_size=0.3)
#Import knearest neighbors Classifier model
from sklearn.neighbors import KNeighborsClassifier

#Create KNN Classifier
knn = KNeighborsClassifier(n_neighbors=5)

#Train the model using the training sets
knn.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = knn.predict(X_test)
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

3.

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
```



```

X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

#standardize values
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting classifier to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

```

4.

```

# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']
# load dataset
pima = pd.read_csv("diabetes.csv", header=1, names=col_names, sep=";", skiprows=1)
print(pima.head())

#split dataset in features and target variable
feature_cols = ['pregnant', 'insulin', 'bmi', 'age', 'glucose', 'bp', 'pedigree']

```

```

X = pima[feature_cols] # Features
y = pima.label # Target variable
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1) # 70%
training and 30% test
# Create Decision Tree classifier object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifier
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

```

5.

```

# Importing the required packages
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

```

```

# Function importing Dataset
def importdata():
    balance_data = pd.read_csv(
'https://archive.ics.uci.edu/ml/machine-learning-'+
'databases/balance-scale/balance-scale.data',
    sep= ',', header = None)

# Printing the dataset shape
print ("Dataset Length: ", len(balance_data))

print ("Dataset Shape: ", balance_data.shape)

```

```

# Printing the dataset obseravtions
print ("Dataset: ",balance_data.head())
return balance_data

# Function to split the dataset
def splitdataset(balance_data):

    # Separating the target variable
    X = balance_data.values[:, 1:5]
    Y = balance_data.values[:, 0]

    # Splitting the dataset into train and test
    X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size = 0.3, random_state = 100)

    return X, Y, X_train, X_test, y_train, y_test

# Function to perform training with giniIndex.
def train_using_gini(X_train, X_test, y_train):

    # Creating the classifier object
    clf_gini = DecisionTreeClassifier(criterion = "gini",
    random_state = 100,max_depth=3, min_samples_leaf=5)

    # Performing training
    clf_gini.fit(X_train, y_train)
    return clf_gini

# Function to perform training with entropy.
def tarin_using_entropy(X_train, X_test, y_train):

    # Decision tree with entropy
    clf_entropy = DecisionTreeClassifier(
        criterion = "entropy", random_state = 100,
        max_depth = 3, min_samples_leaf = 5)

```

```
# Performing training
clf_entropy.fit(X_train, y_train)
return clf_entropy

# Function to make predictions
def prediction(X_test, clf_object):

    # Predicton on test with giniIndex
    y_pred = clf_object.predict(X_test)
    print("Predicted values:")
    print(y_pred)
    return y_pred

# Function to calculate accuracy
def cal_accuracy(y_test, y_pred):

    print("Confusion Matrix: ",
          confusion_matrix(y_test, y_pred))

    print ("Accuracy : ",
            accuracy_score(y_test,y_pred)*100)

    print("Report : ",
          classification_report(y_test, y_pred))

# Driver code
def main():

    # Building Phase
    data = importdata()
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
    clf_gini = train_using_gini(X_train, X_test, y_train)
    clf_entropy = tarin_using_entropy(X_train, X_test, y_train)
```

```

# Operational Phase

print("Results Using Gini Index:")

# Prediction using gini
y_pred_gini = prediction(X_test, clf_gini)
cal_accuracy(y_test, y_pred_gini)

print("Results Using Entropy:")
# Prediction using entropy
y_pred_entropy = prediction(X_test, clf_entropy)
cal_accuracy(y_test, y_pred_entropy)

# Calling main function
if __name__=="__main__":
    main()

```

6.

```

from sklearn import tree
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
x = cancer.data
y = cancer.target

clf = DecisionTreeClassifier(max_depth = 1000)
x_train,x_test,y_train,y_test = train_test_split(x,y)

fig = clf.fit(x_train,y_train)
tree.plot_tree(fig)
plt.show()

```

7.

```

import os
import pandas as pd
import numpy as np

```

```

from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.tree import DecisionTreeRegressor
def MAPE(Y_actual,Y_Predicted):
    Mape = np.mean(np.abs((Y_actual - Y_Predicted)/Y_actual))*100
    return Mape
bike = pd.read_csv("Bike.csv")
X = bike.drop(['cnt'],axis=1)
Y = bike['cnt']
# Splitting the dataset into 80% training data and 20% testing data.
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=.20, random_state=0)
DT_model = DecisionTreeRegressor(max_depth=5).fit(X_train,Y_train)
DT_predict = DT_model.predict(X_test) #Predictions on Testing data
DT_MAPE = MAPE(Y_test,DT_predict)
Accuracy_DT = 100 - DT_MAPE
print("MAPE: ",DT_MAPE)
print('Accuracy of Decision Tree model: {:.2f}%'.format(Accuracy_DT))
print("Predicted values:\n",DT_predict)

```

8.

```

import os
import numpy as np
import pandas as pd
import numpy as np, pandas as pd
import matplotlib.pyplot as plt

from sklearn import tree, metrics

data
=pd.read_csv('car.data',names=['buying','maint','doors','persons','lug_boot','safety','class'])
data.head()

data['class'],class_names = pd.factorize(data['class'])
print(class_names)
print(data['class'].unique())

data['buying'],_ = pd.factorize(data['buying'])
data['maint'],_ = pd.factorize(data['maint'])
data['doors'],_ = pd.factorize(data['doors'])
data['persons'],_ = pd.factorize(data['persons'])

```

```

data['lug_boot'],_ = pd.factorize(data['lug_boot'])
data['safety'],_ = pd.factorize(data['safety'])
data.head()
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
# split data randomly into 70% training and 30% test
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.3,
random_state=0)
# train the decision tree
dtree = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
dtree.fit(X_train, y_train)
# use the model to make predictions with the test data
y_pred = dtree.predict(X_test)
# how did our model perform?
count_misclassified = (y_test != y_pred).sum()
print('Misclassified samples: {}'.format(count_misclassified))
accuracy = metrics.accuracy_score(y_test, y_pred)
print('Accuracy: {:.2f}'.format(accuracy))

```

9.

```

#Import scikit-learn dataset library
from sklearn import datasets
import pandas as pd
import numpy as np
#Load dataset
dataset = pd.read_csv("bill_authentication.csv")
X = dataset.iloc[:, 0:4].values
y = dataset.iloc[:, 4].values
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
# Feature Scaling
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

```

```

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators=20, random_state=0)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
Y_Pred = (y_pred>=0.5).astype("int").ravel()

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,Y_Pred))
print(classification_report(y_test,Y_Pred))
print(accuracy_score(y_test, Y_Pred))

```

10.

```

from pandas import DataFrame
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

Data = {'x':
[25,34,22,27,33,33,31,22,35,34,67,54,57,43,50,57,59,52,65,47,49,48,35,33,44,45,38,43,51,46],
      'y':
[79,51,53,78,59,74,73,57,69,75,51,32,40,47,53,36,35,58,59,50,25,20,14,12,20,5,29,27,8,7]
      }

df = DataFrame(Data,columns=['x','y'])

kmeans = KMeans(n_clusters=4).fit(df)
centroids = kmeans.cluster_centers_
print(centroids)

plt.scatter(df['x'], df['y'], c= kmeans.labels_.astype(float), s=50, alpha=0.5)
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=50)
plt.show()

```


11.

```

#Import scikit-learn dataset library
from sklearn import datasets
from sklearn.model_selection import train_test_split
#Load dataset
df = pd.read_csv('Naïve-Bayes-Classification-Data.csv')
x = df.drop('diabetes', axis = 1)
y = df['diabetes']
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target,
test_size=0.3,random_state=109) # 70% training and 30% test
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB
#Create a Gaussian Classifier
gnb = GaussianNB()
#Train the model using the training sets
gnb.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = gnb.predict(X_test)
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

```

12.

```

from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn import datasets
from sklearn.metrics import confusion_matrix
iris = datasets.load_iris()
gnb = GaussianNB()
mnb = MultinomialNB()
y_pred_gnb = gnb.fit(iris.data, iris.target).predict(iris.data)
cnf_matrix_gnb = confusion_matrix(iris.target, y_pred_gnb)

```

```
print(cnf_matrix_gnb)

y_pred_mnb = mnb.fit(iris.data, iris.target).predict(iris.data)
cnf_matrix_mnb = confusion_matrix(iris.target, y_pred_mnb)
print(cnf_matrix_mnb)
```

13.

```
import pandas as pd

data = pd.read_csv('titanic.csv')

print(data.head())

y = list(map(lambda v: 'yes' if v == 1 else 'no', data['Survived'].values)) # target values as string

# We won't use the 'Name' nor the 'Fare' field

X = data[['Pclass', 'Sex', 'Age', 'Siblings/Spouses Aboard', 'Parents/Children Aboard']].values #
features values
print(len(y)) # >> 887

# We'll take 600 examples to train and the rest to the validation process
y_train = y[:600]
y_val = y[600:]

X_train = X[:600]
X_val = X[600:]

## Creating the Naive Bayes Classifier instance with the training data

nbc = NaiveBayesClassifier(X_train, y_train)

total_cases = len(y_val) # size of validation set

# Well classified examples and bad classified examples
good = 0
bad = 0
```

```

for i in range(total_cases):
    predict = nbc.classify(X_val[i])
    # print(y_val[i] + ' ----- ' + predict)
    if y_val[i] == predict:
        good += 1
    else:
        bad += 1

```

```

print('TOTAL EXAMPLES:', total_cases)
print('RIGHT:', good)
print('WRONG:', bad)
print('ACCURACY:', good/total_cases)

```

14.

#Loading the Dataset

```
from sklearn.datasets import load_breast_cancer
```

```
data_loaded = load_breast_cancer()
```

```
X = data_loaded.data
```

```
y = data_loaded.target
```

#Splitting the dataset into training and testing variables

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(data_loaded.data, data_loaded.target,
                                                    test_size=0.2, random_state=20)
```

#keeping 80% as training data and 20% as testing data

```
from sklearn.naive_bayes import GaussianNB
```

#Calling the Class

```
naive_bayes = GaussianNB()
```

#Fitting the data to the classifier

```
naive_bayes.fit(X_train, y_train)
```

```
#Predict on test data
y_predicted = naive_bayes.predict(X_test)
#Import metrics class from sklearn
from sklearn import metrics
print(metrics.accuracy_score(y_predicted , y_test))
```

15.

```
#Loading the Dataset
from sklearn.datasets import load_breast_cancer

data_loaded = load_breast_cancer()
X = data_loaded.data
y = data_loaded.target

#Splitting the dataset into training and testing variables
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data_loaded.data, data_loaded.target,
test_size=0.2, random_state=20)

#keeping 80% as training data and 20% as testing data

from sklearn.naive_bayes import GaussianNB
#Calling the Class
naive_bayes = GaussianNB()

#Fitting the data to the classifier
naive_bayes.fit(X_train , y_train)

#Predict on test data
y_predicted = naive_bayes.predict(X_test)
#Import metrics class from sklearn
from sklearn import metrics
print(metrics.accuracy_score(y_predicted , y_test))
```

16.

```

#prepare data downloaded from UCL
import csv

import pandas as pd

# add header names
headers = ['age', 'sex', 'chest_pain', 'resting_blood_pressure',
           'serum_cholesterol', 'fasting_blood_sugar', 'resting_ecg_results',
           'max_heart_rate_achieved', 'exercise_induced_angina', 'oldpeak', 'slope of the peak',
           'num_of_major_vessels', 'thal', 'heart_disease']

heart_df = pd.read_csv('heart.dat', sep=' ', names=headers)
print(heart_df.shape)

# check missing values
print(heart_df.isna().sum())

# check if all columns are numeric
print(heart_df.dtypes)

import numpy as np
import warnings
warnings.filterwarnings("ignore") #suppress warnings
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

#convert input to numpy arrays
X = heart_df.drop(columns=['heart_disease'])

#replace target class with 0 and 1
#1 means "have heart disease" and 0 means "do not have heart disease"
heart_df['heart_disease'] = heart_df['heart_disease'].replace(1, 0)
heart_df['heart_disease'] = heart_df['heart_disease'].replace(2, 1)

y_label = heart_df['heart_disease'].values.reshape(X.shape[0], 1)

```

```
#split data into train and test set
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y_label, test_size=0.2, random_state=2)

#standardize the dataset
sc = StandardScaler()
sc.fit(Xtrain)
Xtrain = sc.transform(Xtrain)
Xtest = sc.transform(Xtest)
print(f"Shape of train set is {Xtrain.shape}")
print(f"Shape of test set is {Xtest.shape}")
print(f"Shape of train label is {ytrain.shape}")
print(f"Shape of test labels is {ytest.shape}")

class NeuralNet():
    """
    A two layer neural network
    """

    def __init__(self, layers=[13,8,1], learning_rate=0.001, iterations=100):
        self.params = {}
        self.layers = layers
        self.learning_rate = learning_rate
        self.iterations = iterations
        self.loss = []
        self.sample_size = None
        self.X = None
        self.y = None

    def init_weights(self):
        """
        Initialize the weights from a random normal distribution
        """
        np.random.seed(1) # Seed the random number generator
        self.params["W1"] = np.random.randn(self.layers[0], self.layers[1])
```

```

self.params['b1'] = np.random.randn(self.layers[1],)
self.params['W2'] = np.random.randn(self.layers[1],self.layers[2])
self.params['b2'] = np.random.randn(self.layers[2],)

```

```
def relu(self,Z):
```

```
    """
```

```

    The ReLu function performs a threshold
    operation to each input element where values less
    than zero are set to zero.
    """
```

```
    return np.maximum(0, Z)
```

```
def sigmoid(self,Z):
```

```
    """
```

```

    The sigmoid function takes in real numbers in any range and
    squashes it to a real-valued output between 0 and 1.
    """
```

```
    return 1.0 / (1.0 + np.exp(-Z))
```

```
def entropy_loss(self,y, yhat):
```

```
    nsample = len(y)
```

```

    loss = -1/nsample * (np.sum(np.multiply(np.log(yhat), y) + np.multiply((1 - y), np.log(1 -
    yhat))))

```

```
    return loss
```

```
def forward_propagation(self):
```

```
    """
```

```

    Performs the forward propagation
    """
```

```
    Z1 = self.X.dot(self.params['W1']) + self.params['b1']
```

```
    A1 = self.relu(Z1)
```

```
    Z2 = A1.dot(self.params['W2']) + self.params['b2']
```

```
    yhat = self.sigmoid(Z2)
```

```
    loss = self.entropy_loss(self.y,yhat)
```

```
# save calculated parameters
```

```
self.params['Z1'] = Z1
```

```
self.params['Z2'] = Z2
```

```
self.params['A1'] = A1
```

```
return yhat,loss
```

```
def back_propagation(self,yhat):
```

```
'''
```

```
Computes the derivatives and update weights and bias according.
```

```
'''
```

```
def dRelu(x):
```

```
    x[x<=0] = 0
```

```
    x[x>0] = 1
```

```
    return x
```

```
dl_wrt_yhat = -(np.divide(self.y,yhat) - np.divide((1 - self.y),(1-yhat)))
```

```
dl_wrt_sig = yhat * (1-yhat)
```

```
dl_wrt_z2 = dl_wrt_yhat * dl_wrt_sig
```

```
dl_wrt_A1 = dl_wrt_z2.dot(self.params['W2'].T)
```

```
dl_wrt_w2 = self.params['A1'].T.dot(dl_wrt_z2)
```

```
dl_wrt_b2 = np.sum(dl_wrt_z2, axis=0)
```

```
dl_wrt_z1 = dl_wrt_A1 * dRelu(self.params['Z1'])
```

```
dl_wrt_w1 = self.X.T.dot(dl_wrt_z1)
```

```
dl_wrt_b1 = np.sum(dl_wrt_z1, axis=0)
```

```
#update the weights and bias
```

```
self.params['W1'] = self.params['W1'] - self.learning_rate * dl_wrt_w1
```

```
self.params['W2'] = self.params['W2'] - self.learning_rate * dl_wrt_w2
```

```
self.params['b1'] = self.params['b1'] - self.learning_rate * dl_wrt_b1
```

```
self.params['b2'] = self.params['b2'] - self.learning_rate * dl_wrt_b2
```



```

def fit(self, X, y):
    """
    Trains the neural network using the specified data and labels
    """
    self.X = X
    self.y = y
    self.init_weights() #initialize weights and bias

    for i in range(self.iterations):
        yhat, loss = self.forward_propagation()
        self.back_propagation(yhat)
        self.loss.append(loss)

def predict(self, X):
    """
    Predicts on a test data
    """
    Z1 = X.dot(self.params['W1']) + self.params['b1']
    A1 = self.relu(Z1)
    Z2 = A1.dot(self.params['W2']) + self.params['b2']
    pred = self.sigmoid(Z2)
    return np.round(pred)

def acc(self, y, yhat):
    """
    Calculates the accuracy between the predicted values and the truth labels
    """
    acc = int(sum(y == yhat) / len(y) * 100)
    return acc

def plot_loss(self):
    """
    Plots the loss curve
    """

```

```

plt.plot(self.loss)
plt.xlabel("Iteration")
plt.ylabel("logloss")
plt.title("Loss curve for training")
plt.show()
NeuralNet()

```

17.

```

import sys, os
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.model_selection import train_test_split, GridSearchCV
# Read data
x, labels = read_data("points_class_0.txt", "points_class_1.txt")
# Split data to train and test on 80-20 ratio
X_train, X_test, y_train, y_test = train_test_split(x, labels, test_size = 0.2, random_state=0)
# Plot training and test data
plot_data(X_train, y_train, X_test, y_test)
# Create a linear SVM classifier
clf = svm.SVC(kernel='linear')
# Train classifier
clf.fit(X_train, y_train)
# Plot decision function on training and test data
plot_decision_function(X_train, y_train, X_test, y_test, clf)
# Make predictions on unseen test data
clf_predictions = clf.predict(X_test)
print("Accuracy: {}".format(clf.score(X_test, y_test) * 100 ))
# Linearly separable data with noise
# Create a linear SVM classifier with C = 1
clf = svm.SVC(kernel='linear', C=1)
# Create SVM classifier based on RBF kernel.
clf = svm.SVC(kernel='rbf', C = 10.0, gamma=0.1)
# Grid Search
param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001, 0.00001, 10]}

```

```
# Make grid search classifier
clf_grid = GridSearchCV(svm.SVC(), param_grid, verbose=1)
# Train the classifier for searching the best parameter for our classifier.
clf_grid.fit(X_train, y_train)
# clf = grid.best_estimator_()
print("Best Parameters:\n", clf_grid.best_params_)
print("Best Estimators:\n", clf_grid.best_estimator_)
```

18.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
bankdata = pd.read_csv("bill_authentication.csv")
print(bankdata.shape)
X = bankdata.drop('Class', axis=1)
y = bankdata['Class']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

19.

```
import pandas as pd
data = pd.read_csv("apples_and_oranges.csv")
from sklearn.model_selection import train_test_split
training_set, test_set = train_test_split(data, test_size = 0.2, random_state = 1)
X_train = training_set.iloc[:,0:2].values
Y_train = training_set.iloc[:,2].values
X_test = test_set.iloc[:,0:2].values
Y_test = test_set.iloc[:,2].values
```

```

from sklearn.svm import SVC

classifier = SVC(kernel='rbf', random_state = 1)

classifier.fit(X_train,Y_train)

#Predicting the classes for test set
Y_pred = classifier.predict(X_test)

#Attaching the predictions to test set for comparing
test_set["Predictions"] = Y_pred

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(Y_test,Y_pred)

accuracy = float(cm.diagonal().sum())/len(Y_test)

print("\nAccuracy Of SVM For The Given Dataset : ", accuracy)

```

20.

```

import numpy as np
from sklearn import preprocessing, cross_validation, neighbors, svm
import pandas as pd

df = pd.read_csv('breast-cancer-wisconsin.data.txt')
df.replace('?',-99999, inplace=True)
df.drop(['id'], 1, inplace=True)

X = np.array(df.drop(['class'], 1))
y = np.array(df['class'])

X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y, test_size=0.2)

clf = svm.SVC()

clf.fit(X_train, y_train)

confidence = clf.score(X_test, y_test)

print(confidence)

example_measures = np.array([[4,2,1,1,1,2,3,2,1]])

example_measures = example_measures.reshape(len(example_measures), -1)

prediction = clf.predict(example_measures)

print(prediction)

```

CHAPTER 12: Advanced Learning

1.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
from keras import initializers
from keras import regularizers
from keras import constraints
model = Sequential()
model.add(Dense(32, input_shape=(16,), kernel_initializer = 'he_uniform',
    kernel_regularizer = None, kernel_constraint = 'MaxNorm', activation = 'relu'))
model.add(Dense(16, activation = 'relu'))
model.add(Dense(8))
where,
```

- Line 1-5 imports the necessary modules.
- Line 7 creates a new model using Sequential API.
- Line 9 creates a new Dense layer and add it into the model. Dense is an entry level layer provided by Keras, which accepts the number of neurons or units (32) as its required parameter. If the layer is first layer, then we need to provide Input Shape, (16,) as well. Otherwise, the output of the previous layer will be used as input of the next layer. All other parameters are optional.
 - o First parameter represents the number of units (neurons).
 - o input_shape represent the shape of input data.
 - o kernel_initializer represent initializer to be used. he_uniform function is set as value.
 - o kernel_regularizer represent regularizer to be used. None is set as value.
 - o kernel_constraint represent constraint to be used. MaxNorm function is set as value.
 - o activation represent activation to be used. relu function is set as value.
- Line 10 creates second Dense layer with 16 units and set relu as the activation function.
- Line 11 creates final Dense layer with 8 units.

2.

```
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

```

from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
import numpy as np
import keras as K
np.random.seed(4)
model = K.models.Sequential()
model.add(K.layers.Dense(units=7, input_dim=4,
    activation='tanh'))
model.add(K.layers.Dense(units=3, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='sgd', metrics=['accuracy'])

```

3.

```

import pandas as pd

import numpy as np

df = pd.read_csv('dbfs://FileStore/tables/dgshzbe11508270992746/insurance_claims.csv')

feats =
['policy_state', 'insured_sex', 'insured_education_level', 'insured_occupation', 'insured_hobbies',
'insured_relationship', 'collision_type', 'incident_severity', 'authorities_contacted', 'incident_stat
e', 'incident_city', 'incident_location', 'property_damage', 'police_report_available', 'auto_make',
auto_model', 'fraud_reported', 'incident_type']

df_final = pd.get_dummies(df, columns=feats, drop_first=True)

from sklearn.model_selection import train_test_split

X =
df_final.drop(['fraud_reported_Y', 'policy_csl', 'policy_bind_date', 'incident_date'], axis=1).values
y = df_final['fraud_reported_Y'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

```

```
X_test = sc.transform(X_test)
```

```

import keras
from keras.models import Sequential

from keras.layers import Dense
classifier = Sequential()
classifier.add(
    Dense(3, kernel_initializer = 'uniform',
          activation = 'relu', input_dim=5))
classifier.add(
    Dense(3, kernel_initializer = 'uniform',
          activation = 'relu'))
classifier.add(
    Dense(1, kernel_initializer = 'uniform',
          activation = 'sigmoid'))
classifier.compile(optimizer= 'adam',
                  loss = 'binary_crossentropy',
                  metrics = ['accuracy'])
classifier.fit(X_train, y_train, batch_size = 10, epochs = 100)
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

```

4.

```

a.
from keras.models import Sequential
from keras.layers import Activation, Dense
from keras import initializers

my_init = initializers.Zeros()
model = Sequential()
model.add(Dense(512, activation = 'relu', input_shape = (784,),
               kernel_initializer = my_init))

```

b.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
from keras import initializers
my_init = initializers.Ones()
model.add(Dense(512, activation = 'relu', input_shape = (784,),
    kernel_initializer = my_init))
```

c.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
from keras import initializers
```

```
my_init = initializers.Constant(value = 0) model.add(
    Dense(512, activation = 'relu', input_shape = (784,), kernel_initializer = my_init)
)
```

d.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
from keras import initializers

my_init = initializers.RandomNormal(mean=0.0,
stddev = 0.05, seed = None)
model.add(Dense(512, activation = 'relu', input_shape = (784,),
    kernel_initializer = my_init))
```

e.

```
from keras import initializers

my_init = initializers.RandomUniform(minval = -0.05, maxval = 0.05, seed = None)
model.add(Dense(512, activation = 'relu', input_shape = (784,),
    kernel_initializer = my_init))
```

f.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
from keras import initializers
```



```
my_init = initializers.TruncatedNormal(mean = 0.0, stddev = 0.05, seed = None)
model.add(Dense(512, activation = 'relu', input_shape = (784,),
    kernel_initializer = my_init))
```

g.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
from keras import initializers
```

```
my_init = initializers.VarianceScaling(
    scale = 1.0, mode = 'fan_in', distribution = 'normal', seed = None)
model.add(Dense(512, activation = 'relu', input_shape = (784,),
    kernel_initializer = my_init))
```

h.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
from keras import initializers
```

```
my_init = initializers.Orthogonal(gain = 1.0, seed = None)
model.add(Dense(512, activation = 'relu', input_shape = (784,),
    kernel_initializer = my_init))
```

i.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
from keras import initializers
```

```
my_init = initializers.Identity(gain = 1.0) model.add(
    Dense(512, activation = 'relu', input_shape = (784,), kernel_initializer = my_init)
)
```

j.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
from keras import initializers
```

```
my_init = initializers.Identity(gain = 1.0) model.add(  
    Dense(512, activation = 'relu', input_shape = (784,)),  
    kernel_initializer = my_init)  
)  
k.
```

```
from keras.models import Sequential  
from keras.layers import Activation, Dense  
from keras import constraints
```

```
my_constrain = constraints.UnitNorm(axis = 0)  
model = Sequential()  
model.add(Dense(512, activation = 'relu', input_shape = (784,)),  
    kernel_constraint = my_constrain))  
l.
```

```
from keras.models import Sequential  
from keras.layers import Activation, Dense  
from keras import constraints
```

```
my_constrain = constraints.MaxNorm(max_value = 2, axis = 0)  
model = Sequential()  
model.add(Dense(512, activation = 'relu', input_shape = (784,)),  
    kernel_constraint = my_constrain))  
m.
```

```
from keras.models import Sequential  
from keras.layers import Activation, Dense  
from keras import constraints
```

```
my_constrain = constraints.MinMaxNorm(min_value = 0.0, max_value = 1.0, rate = 1.0, axis =  
0)  
model = Sequential()  
model.add(Dense(512, activation = 'relu', input_shape = (784,)),  
    kernel_constraint = my_constrain))  
n.
```

```
from keras.models import Sequential
```

```
from keras.layers import Activation, Dense
from keras import regularizers
```

```
my_regularizer = regularizers.l1(0.)
model = Sequential()
model.add(Dense(512, activation = 'relu', input_shape = (784,),
    kernel_regularizer = my_regularizer))
```

o.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
from keras import regularizers
```

```
my_regularizer = regularizers.l2(0.)
model = Sequential()
model.add(Dense(512, activation = 'relu', input_shape = (784,),
    kernel_regularizer = my_regularizer))
```

p.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
```

```
model = Sequential()
model.add(Dense(512, activation = 'linear', input_shape = (784,)))
```

q.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
```

```
model = Sequential()
model.add(Dense(512, activation = 'elu', input_shape = (784,)))
```

r.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
```

```
model = Sequential()
model.add(Dense(512, activation = 'selu', input_shape = (784,)))
```

s.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
```

```
model = Sequential()
model.add(Dense(512, activation = 'softplus', input_shape = (784,)))
```

t.

```
from keras.models import Sequential
from keras.layers import Activation, Dense
```

```
model = Sequential()
model.add(Dense(512, activation = 'softsign', input_shape = (784,)))
```

5.

```
# TensorFlow and tf.keras
import tensorflow as tf

# Only necessary if you're using Keras (obviously)
from tensorflow import keras

# Helper libraries
import numpy as np
import pandas as pd
import math
import pprint as pp

data = pd.read_csv("zoo.csv")

# Shuffle
data = data.sample(frac=1).reset_index(drop=True)

# Split
data_total_len = data[data.columns[0]].size
data_train_frac = 0.6
split_index = math.floor(data_total_len*data_train_frac)
training_data = data.iloc[:split_index]
evaluation_data = data.iloc[split_index:];
column_count = 18;
```

```

label_column_index = 17 # Zero based index (so this is the 18th column)

def preprocess(data):
    X = data.iloc[:, 1:column_count-1]
    y = data.iloc[:, label_column_index]
    y = y-1 # shift label value range from 1-7 to 0-6
    return X, y
X, y = preprocess(data);
(train_data, train_labels) = preprocess(training_data);
(eval_data, eval_labels) = preprocess(evaluation_data);
feature_columns = [
    tf.feature_column.categorical_column_with_vocabulary_list(
        key = col_name,
        vocabulary_list = data[col_name].unique()
    ) for col_name in X.columns
]
deep_features = [tf.feature_column.indicator_column(col) for col in feature_columns]
model = tf.estimator.DNNClassifier(
    feature_columns = deep_features,
    hidden_units = [30,20,10],
    n_classes = 7
)
model = keras.Sequential()
model.add(keras.layers.Dense(30, input_shape=(16,)))
model.add(keras.layers.Dense(20, activation=tf.nn.relu))
model.add(keras.layers.Dense(7, activation=tf.nn.softmax))
epochs = 150
batch_size = 12
def train_input_fn():
    dataset = tf.data.Dataset.from_tensor_slices((dict(train_data), train_labels))
    # shuffle, repeat, and batch the examples
    dataset = dataset.shuffle(1000)
    dataset = dataset.repeat(epochs).batch(batch_size)
    return dataset

```

```
''' keras version
model.train(input_fn = train_input_fn)
model.compile(
    optimizer=tf.train.AdamOptimizer(),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)'''

def eval_input_fn():
    dataset = tf.data.Dataset.from_tensor_slices((dict(eval_data), eval_labels))
    # repeat, and batch the examples
    # NOTE: do not Shuffle
    dataset = dataset.repeat(1).batch(batch_size)
    return dataset

model.evaluate(input_fn=eval_input_fn)

''' keras version
model.evaluate(eval_data, eval_labels)
animal_type = ['Mammal', 'Bird', 'Reptile', 'Fish', 'Amphibian', 'Bug', 'Invertebrate']
prediction_data = evaluation_data
def predict_input_fn():
    dataset = tf.data.Dataset.from_tensor_slices((dict(prediction_data), eval_labels))
    # repeat, and batch the examples
    # NOTE: do not Shuffle
    dataset = dataset.repeat(1).batch(batch_size)
    return dataset

predictions = model.predict(input_fn = predict_input_fn)

model.fit(train_data, train_labels, epochs = epochs, batch_size = batch_size)
# for keras, write predictions = model.predict(prediction_data)
for i, prediction in enumerate(predictions):
    predicted_animal = animal_type[prediction.argmax(axis=-1)]
    correct_animal = animal_type[eval_labels.iloc[i]]
    print("Predicted: {} \n Actual answer: {} \n Probabilities: {} \n".format(predicted_animal,
correct_animal, prediction))
```

6.

```

# iris_nn.py
# Iris classification Keras 2.1.4 TensorFlow 1.4.0

import numpy as np
import keras as K

import os
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'

def main():
    print("Iris dataset using Keras/TensorFlow ")

    print("Loading Iris data into memory \n")
    data_file = ".\\Data\\iris_data.txt"
    train_x = np.loadtxt(data_file, usecols=[0,1,2,3],
        delimiter=",", skiprows=0, dtype=np.float32)
    train_y = np.loadtxt(data_file, usecols=[4,5,6],
        delimiter=",", skiprows=0, dtype=np.float32)

    np.random.seed(4)
    model = K.models.Sequential()
    model.add(K.layers.Dense(units=7, input_dim=4,
        activation='tanh'))
    model.add(K.layers.Dense(units=3, activation='softmax'))
    model.compile(loss='categorical_crossentropy',
        optimizer='sgd', metrics=['accuracy'])

    print("Starting training \n")
    h = model.fit(train_x, train_y, batch_size=1,
        epochs=12, verbose=1) # 1 = very chatty
    print("\nTraining finished \n")

    eval = model.evaluate(train_x, train_y, verbose=0)

```

```
print("Evaluation: loss = %0.6f accuracy = %0.2f%% \n" \
      % (eval[0], eval[1]*100) )
```

```
np.set_printoptions(precision=4)
unknown = np.array([[6.1, 3.1, 5.1, 1.1]],
                   dtype=np.float32)
predicted = model.predict(unknown)
print("Using model to predict species for features: ")
print(unknown)
print("\nPredicted species is: ")
print(predicted)
```

```
if __name__=="__main__":
    main()
```

