

Chapter 14

A. Multiple Choice Answers

1. C
2. B
3. C
4. B
5. B
6. B
7. A

B. Exercise Questions

1. Explain the need of exception handling.

- When programmer writes program, then there is a probability that he or she may always ends with some kind of errors. An error is something that can produce incorrect or irrelevant output or even be responsible for the system to crash. It is, therefore, very important to find out these errors and fix them so that the program will not terminate or crash during execution. Therefore there is need of exception handling to handle errors at run time.

2. Demonstrate with suitable example, how we can handle exceptions within a program?

The following program demonstrate how we can handle exception handling.

```
def demo():
    try:
        num = int(input('Enter the number:'))
        print('Entered number is :',num)
        return num
    except Exception as e:
        print(e)

while(1):
    n = demo()
    print(type(n))
    if(isinstance(n,int)):
        break
    else:
        continue
print(n)
```

Output:

```
Enter the number:f
invalid literal for int() with base 10: 'f'
<class 'NoneType'>
Enter the number:8
Entered number is : 8
<class 'int'>
8
```

3. Write a program to handle arithmetic exceptions.

Program: The following program demonstrate about division by zero i.e. how to handle arithmetic exception.

```
def demo():
    try:
        num1 = int(input('Enter the number:'))

        num2 = int(input('Enter the number:'))

        print('num1:', num1)
        print('num2:', num2)
        print('Quotient: ', num1/num2)

    except Exception as e:
        print(e)

demo()
```

Output:

```
Enter the number:5
Enter the number:0
num1: 5
num2: 0
division by zero
```

4. Justify, if we can have multiple except blocks while handling exceptions. If yes, how?

Yes we can write multiple except blocks, the following program demonstrate the same.

```
def demo():
    try:
        num1 = int(input('Enter the number:'))
```

```
num2 = int(input('Enter the number:'))

print('num1:', num1)
print('num2:', num2)
print('Quotient: ', num1/num2)
except ZeroDivisionError as e:
    print('Can not divide by zero:', e)

except ValueError as e:
    print(e)
demo()
```

5. State the uses of raise keyword with suitable program

The program on raise keyword is as follows

```
try:
    age = int(input("Enter the age:"))
    if age < 18:
        raise ValueError;
    else:
        print("The age is valid")
except ValueError:
    print("The age is not valid")
```

1. Write a program that will ask the user for an input and tries to handle the error when the programmer will try to type cast the input to an int using try and except blocks.

```
try:
    ch = (input("Enter the Character:"))
    ch = int(ch)
    print(ch)
except ValueError as e:
    print(e)
```

Output:

```
Enter the Character: u
invalid literal for int() with base 10: 'u'
```

2. Write a function which will take two arguments to concatenate two strings. Assume the type of first argument is unknown but the second argument is the string. Write a Program to handle errors if the programmer will try to concatenate unknown input (unknown input can be of integer types or float) type to the string using try and except blocks.

```
def two_str_conc(str1, str2):  
    return str1 + str2  
  
try:  
    print(two_str_conc(10, 'abc'))  
except TypeError as e:  
    print(e)
```

Output:

unsupported operand type(s) for +: 'int' and 'str'

3. Write a Program that will try to handle the error when you will try to divide the float as the unknown input being entered by the user using try-except block.

```
try:  
    num1 = int(input('Please Enter the number:'))  
    num2 = int(input('Please Enter the second number:'))  
    ans = num1 / num2  
except ValueError as e:  
    print(e)
```

Output:

Please Enter the number:45
Please Enter the second number: p
invalid literal for int() with base 10: 'p'

4. Write a function which will take three parameters. The first parameter to the function will be list, the second parameter will be the index position and the third parameter will be string. The function will insert the string at the given index position within the list. In case of a failure, your function should return to the original list. Write a function which will perform this task using the try and except blocks

```
def list_manipulationn(tmp_list, ind, str1):  
    tmp_list[ind]=str1  
    return tmp_list  
  
try:  
  
    tmp_list = [10,20,30,40,50]  
    ind = int(input('Index position to place string in  
list:'))  
    str1 = input('Please Enter the String to Insert:')  
    print(list_manipulationn([1,2,3,5,6],ind,str1))  
except IndexError as e:  
    print(e)
```

Output:

```
Index position to place string in list:10  
Please Enter the String to Insert: India  
list assignment index out of range
```