PRACTICE

2. a . Code execute and debug programs that Use i/o statements

```
 x=input("enter first number:")
y=input("enter second number:")
i=int(x)
j=int(y)
print("the sum:",i+j)
```

OTPUT:

```
enter first number:10
enter second number:20
the sum: 30
```

2. b. Evaluate expressions and displays formatted output

```
test_str="58*66/5"#initializing string
print("the original string is:"+test_str)
res= eval(test_str)
print("the evaluated result is:"+str(res))
```

OUTPUT:

```
the original string is:58*66/5
the evaluated result is:765.6
```

2. c. Evaluate expressions to examine the operator precedence

```
print(5-7)
print(10-4*2)
#precedence of (or) & and
meal="fruit"
money=2
if meal=="fruit" or meal=="sandwich" and money>=2:
    print("lunch being delivered")
else:
    print("can't deliver lunch")
    #left-right associativity
print(5*2//3)
    #left-right associativity
print(5*(2//3))
    #shows the right-left associativity of **
print(2**3**2)
    #if 2 needs to be exponated first,need to use()
print((2**3)**2)
```

OUTPUT:

```
-2
2
lunch being delivered
3
0
512
64
```

3. Identify and Code, execute and debug programs using conditional statements.

```
a = input('Enter first number: ')                    # Store input numbers
b = input('Enter second number: ')
c = input('Enter third number: ')
d = input('Enter fourth number: ')
if a>b:
  print("A is greater than B")
if a!=b:
  print("A not equal to B")
if a == b:
    print("A is equal to B")
elif a>b:
      print("true")
else:
    print("false")
if a == b and c == d:
        print("Hello")
if a == b or c == d:
        print("world")
if 5 > 2: print("five is greater than two!")
print("Yes") if c > d else print("No")
```

OUTPUT:

```
Enter first number: 10
Enter second number: 20
Enter third number: 30
Enter fourth number: 30
A not equal to B
false
world
five is greater than two!
No
```

4. Code, execute and debug programs using loops and conditional statements

```
for x in range(5):
  if x == 3:
      break
  print("value of x is", x)
count = 0
while count < 5:
  if count == 2:
      break
  print("value of count is", count)
  count += 1
else:
  print("While Loop has ended, value of count is", count)
```

OUTPUT:

```
Value of x is 0
Value of x is 1
Value of x is 2
Value of count is 0
Value of count is 1
```

5. a. Code, execute and debug programs to perform following

- **Set operations**
- **Set comprehension**

```
A = {1, 2, 3, 4, 5}                          # initialize A and B
B = {4, 5, 6, 7, 8}
print("Union of A and B is is: ")
print(A | B)
print("Intersection of A and B is: ")
print(A & B)
print("Difference of the set B is: ")
print(A - B)
print("Symmetric Difference of A and B is: ")
print(A ^ B)
```

OUTPUT:

```
Union of A and B is is:
{1, 2, 3, 4, 5, 6, 7, 8}
Intersection of A and B is:
{4, 5}
Difference of the set B is:
{1, 2, 3}
Symmetric Difference of A and B is:
{1, 2, 3, 6, 7, 8}
```

5. b. Code, execute and debug programs to perform following

- **basic operations on tuples**
- **tuple indexing & slicing**

```
my_tuple = (1, "Hello", 3.4)                # tuple with mixed datatypes
print(my_tuple)
my_tuple = ("mouse", [8, 4, 6], (1, 2, 3))  # nested tuple
print(my_tuple)
my_tuple = ('p','o','l','y','t','e','c','h','n','i','c')
print(my_tuple[0])                  # Accessing tuple using indexing
print(my_tuple[-6])                 # Negative indexing for tuple elements
                                    # Accessing tuple using slicing
print(my_tuple[1:4])                        # elements 2nd to 4th
print(my_tuple[:-7])                        # elements beginning to 2nd
print(my_tuple[7:])                         # elements 8th to end
print(my_tuple[:])                          # elements beginning to end
```

OUTPUT:

```
(1, 'Hello', 3.4)
('mouse', [8, 4, 6], (1, 2, 3))
p
e
('o', 'l', 'y')
('p', 'o', 'l', 'y')
('h', 'n', 'i', 'c')
('p', 'o', 'l', 'y', 't', 'e', 'c', 'h', 'n', 'i', 'c')
```

6. Write code snippet to perform following on List

- **basic operations on List**
- **indexing and slicing**
- **comprehension**

```
my_list = ['p', 'o', 'l', 'y', 't', 'e', 'c', 'h', 'n', 'i', 'c']

print(my_list[0])                                        # first item

print(my_list[-1])                                       # last item

n_list = ["Happy", [2, 0, 1, 5]]                         # Nested List

print(n_list[0][1])                                 # Nested indexing

print(my_list[2:5])                           # elements from index 2 to index 4

print(my_list[5:])                            # elements from index 5 to end

print(my_list[:])                             # elements beginning to end

pow2 = [2 ** x for x in range(10)]

print(pow2)
```

OUTPUT:

```
p

c

a

['l', 'y', 't']

['e', 'c', 'h', 'n', 'i', 'c']

['p', 'o', 'l', 'y', 't', 'e', 'c', 'h', 'n', 'i', 'c']

[1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
```

7.Code, execute and debug programs to perform Dictionary

- **indexing**
- **Iterating**
- **comprehension**

```
my_dict = {'name': 'Jack', 'age': 26}            # get vs [] for retrieving elements
print(my_dict['name'])
print(my_dict)
my_dict['age'] = 27                                    # update value
print(my_dict)
my_dict['address'] = 'Downtown'                        # add item
print(my_dict)
squares = {0: 0, 1: 1, 3: 9, 4: 25, 5: 49, 6: 81}
print(squares.pop(4))            # remove a particular item, returns its value
print(squares)                   # remove an arbitrary item, return (key,value)
print(all(squares))
print(any(squares))
squares = {x: x*x for x in range(6)}    # Dictionary Comprehension
print(squares)
```

OUTPUT:

```
Jack
{'name': 'Jack', 'age': 26}
{'name': 'Jack', 'age': 27}
{'name': 'Jack', 'age': 27, 'address': 'Downtown'}
25
{0: 0, 1: 1, 3: 9, 5: 49, 6: 81}
False
True
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

8. a. Code, execute and debug programs to perform string manipulation

```
my_string = "Hello"
print(my_string)
str = 'PYTHON'
print('str = ', str)
print('str[0] = ', str[0])
print('str[-1] = ', str[-1])
list_enumerate = list(enumerate(str))
print('list(enumerate(str) = ', list_enumerate)
print('len(str) = ', len(str))
```

OUTPUT:

```
Hello
str =  PYTHON
str[0] =  P
str[-1] =  N
list(enumerate(str) =  [(0, 'P'), (1, 'Y'), (2, 'T'), (3, 'H'), (4, 'O'), (5, 'N')]
len(str) =  6
```

8. b. Code, execute and debug programs to perform array manipulation

```
import array as arr
a = arr.array('i', [1, 2, 3, 4, 5, 6])
for i in range (0, 3):
    print (a[i], end =" ")
print()
a.insert(1, 4)
print ("Array after insertion : ", end =" ")
for i in (a):
        print (i, end =" ")
print()
#import array as arr
#a = arr.array('i', [1, 2, 3, 4, 5, 6])
print("Access element is: ", a[0])
```

OUTPUT:

```
1 2 3
Array after insertion :  1 4 2 3 4 5 6
Access element is:  1
```

9. a. Code, execute and debug programs to solve the given problem using built in functions

```
k = [1, 3, 4, 6]
print(all(k))                                    #Python all() Function
x = 8
print(eval('x + 1'))             #Python eval() Function Example
s = sum([1, 2,4 ])               #Python sum() Function Example
print(s)
```

OUTPUT:

```
Absolute value of -20 is: 20
True
0b1010
9
7
```

9. b. Code, execute and debug programs to solve the given problem by defining a function

```
def greet(name):
    print("Hello, " + name + ". Good morning!")
greet('Paul')
```

OUTPUT:

```
Hello, Paul. Good morning!
```

9. c. Code, execute and debug programs to solve the given problem using recursion

```
def tri_recursion(k):
  if(k > 0):
    result = k + tri_recursion(k - 1)
    print(result)
  else:
    result = 0
  return result

print("\n\nRecursion Example Results")
tri_recursion(6)
```

OUTPUT:

```
Recursion Example Results

1

3

6

10

15

21
```

9. d. Define anonymous function and code to solve the given problem

```
def myfunc(n):
  return lambda a : a * n
mydoubler = myfunc(2)
mytripler = myfunc(3)
print(mydoubler(11))
print(mytripler(11))
```

OUTPUT:

```
22

33
```

10. a. Create Modules and Packages

mymodule.py

```
person1 = {
 "name":"John",
 "age":36,
 "country":"Norway"
}
```

another.py

```
import mymodule
a = mymodule.person1["age"]
print(a)
b = mymodule.person1["name"]
print(b)
```

OUTPUT:

```
36
John
```

10. b. Code, execute and debug programs using built in modules

```
import math
x=13  # small value of of x
print('log10(x) is :', math.log10(x))
import random
random.seed(2)
print(random.random())
print(random.random())
print(random.random())
```

OUTPUT:

```
log10(x) is : 1.1139433523068367
0.9560342718892494
0.9478274870593494
0.05655136772680869
```

11. a. Code, execute and debug programs using NumPy module.

Installation of NumPy

If you have Python and PIP already installed on a system, then installation of NumPy is very easy.

```
pip install numpy
```

Save this code in **pythonIDLE** using file extension .py

```
import numpy as np
arr1 = np.array([10, 11, 12, 13, 14, 15])
arr2 = np.array([20, 21, 22, 23, 24, 25])
newarr = np.add(arr1, arr2)
print(newarr)
newarr = np.subtract(arr1, arr2)
print(newarr)
newarr = np.multiply(arr1, arr2)
print(newarr)
newarr = np.divide(arr1, arr2)
print(newarr)
print(np.amin(arr1))
print(np.amax(arr2))
```

OUTPUT:

```
[30 32 34 36 38 40]
[-10 -10 -10 -10 -10 -10]
[200 231 264 299 336 375]
[0.5        0.52380952 0.54545455 0.56521739 0.58333333 0.6       ]
10
25
```

11. b. Code, execute and debug programs using series.

```
import pandas as pd
a = [1, 7, 2]
myvar = pd.Series(a, index = ["x", "y", "z"])
print(myvar[0])
print(myvar["y"])
calories = {"day1": 420, "day2": 380, "day3": 390}
myvar = pd.Series(calories)
print(myvar)
#calories = {"day1": 420, "day2": 380, "day3": 390}
myvar = pd.Series(calories, index = ["day1", "day2"])
print(myvar)
```

OUTPUT:

```
1
7
day1    420
day2    380
day3    390
dtype: int64
day1    420
day2    380
dtype: int64
```

11. c. Code, execute and debug programs using dataframes.

```
import pandas as pd
data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}
#load data into a DataFrame object:
df = pd.DataFrame(data)
print(df)
print(df.loc[0])
```

OUTPUT:

```
  calories  duration
0       420        50
1       380        40
2       390        45
calories    420
duration     50
Name: 0, dtype: int64
```

12. a. write code snippet to perform following operations on different types of files

- **Read file**
- **Write to file**

Creat a new text file demofile.txt

Hello! Welcome to demofile.txt

This file is for testing purposes.

Good Luck!

```
f = open("demofile.txt", "r")
print(f.read())
```

```
f = open("demofile.txt", "a")
f.write("Now the file has more content!")
f.close()
f = open("demofile1.txt", "r")    #open and read the file after the appending:
print(f.read())
```

```
f = open("demofile.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()
#open and read the file after the appending:
f = open("demofile.txt", "r")
print(f.read())
```

OUTPUT:

```
Hello! Welcome to demofile.txt

This file is for testing purposes.

Good Luck!

Now the file has more content!

Hello! Welcome to demofile.txt

This file is for testing purposes.

Good Luck!

Now the file has more content!Now the file has more content!

Woops! I have deleted the content!
```

12.b . Write code to perform file operations using dataframes on different file types.

```
import pandas as pd
data = {
 'CHN': {'COUNTRY': 'China', 'POP': 1_398.72, 'AREA': 9_596.96,
 'GDP': 12_234.78, 'CONT': 'Asia'},
 'IND': {'COUNTRY': 'India', 'POP': 1_351.16, 'AREA': 3_287.26,
 'GDP': 2_575.67, 'CONT': 'Asia', 'IND_DAY': '1947-08-15'},
 'USA': {'COUNTRY': 'US', 'POP': 329.74, 'AREA': 9_833.52,
 'GDP': 19_485.39, 'CONT': 'N.America',
 'IND_DAY': '1776-07-04'}}
columns = ('COUNTRY', 'POP', 'AREA', 'GDP', 'CONT', 'IND_DAY')
df = pd.DataFrame(data=data, index=columns).T
df.to_csv('data.csv')
df = pd.read_csv('data.csv', index_col=0)
print(df)
```

OUTPUT:

```
    COUNTRY     POP     AREA       GDP       CONT     IND_DAY
CHN   China  1398.72  9596.96  12234.78       Asia         NaN
IND   India  1351.16  3287.26   2575.67       Asia  1947-08-15
USA      US   329.74  9833.52  19485.39  N.America  1776-07-04
```

13. a. Integrate exception handling into above code

```
import sys
randomList = ['a', 0, 2]
for entry in randomList:
    try:
        print("The entry is", entry)
        r = 1/int(entry)
        break
    except Exception as e:
        print("Oops!", e.__class__, "occurred.")
        print("Next entry.")
        print()
print("The reciprocal of", entry, "is", r)
```

OUTPUT:

```
The entry is a
Oops! <class 'ValueError'> occurred.
Next entry.
The entry is 0
Oops! <class 'ZeroDivisionError'> occurred.
Next entry.
The entry is 2
The reciprocal of 2 is 0.5
```

13.b. Write code snippet to raise exceptions

```
filename = 'John.txt'
try:
 with open(filename) as f_obj:
  contents = f_obj.read()
except FileNotFoundError:
 msg = "Sorry, the file "+ filename + " does not exist."
 print(msg)
```

OUTPUT:

```
Sorry, the file John.txt does not exist.
```