Section A: Theory (10 Questions)

1. What is difference between mutable data type and immutable data type.

- Mutable data type: this data types are not allowed to make any changes in original data

Eg. String,Tuple

```
s='la'
s[1]="M"
```

Output:-

```
  File "C:\Lavanya_Code\Pyton_Lectures\Weekly_assignment.py", line 42, in <mo
    s[1]="M"
    ~^^^
TypeError: 'str' object does not support item assignment
```

- Immutable data type: this data types are allowed to make any changes in original data

Eg. List,dictionary

```
s=[1,2]
s[1]="M"
print(s)
```

Output:-
[1,"M']

2. What is floor division.

Floor: is a mathematical operation that rounds down the result of a division to the nearest integer

Eg:

A=4.6

Print(floor(A))

Output:

4

3. explain Break, Continue, Pass statement in python.

Break: The break statement in Python is used to terminate the loop

Continue: continue statement is opposite to that of the break statement, instead of terminating the loop, it forces to execute the next iteration of the loop. Current iteration is skipped and next itration is start

Pass: The Python pass statement serves as a placeholder in situations where a statement is syntactically necessary, but no actual code is needed. The pass statement is a null operation that returns nothing when executed in a code block.

it simply useful for code readability

4. What is difference between for loop and while loop.

for loops are used when the number of iterations is known beforehand, such as when iterating over elements in a list or any iterable object.

```
numbers = [1, 2, 3, 4, 5]

print("Iterating over the list using a for loop:")
for num in numbers:
    print(num)
```

While loops are used when the number of iterations is not known beforehand, and the loop continues as long as a certain condition is true. The loop variable is typically initialized before the loop, and the condition is checked before each iteration.

```
n='y'
while(n!='n'):

    n=input("enter y if want to continue n if not: ")
    print('welcome')
print("Bye")
```

output:

enter y if want to continue n if not: y

welcome

enter y if want to continue n if not: y

welcome enter y if want to continue n if not: n welcome Bye

5. What is dynamically typed langugae.

in statically typed data we have to specify type of data variable is holding like interger or string so that the specific amount memory is allocated .

Python is a **dynamically typed** language. It doesn't know about the type of the variable until the code is run. So declaration is of no use. What it does is, It stores that value at some memory location and then binds that variable name to point that memory . And makes the contents of the that memory accessible through that variable name. So the data type does not matter. As it will get to know the type of the value at run-time.

6. What is built in data types in python.

1. **Numerical type:** floats,int,complex

Int: in you can store integers type data

e.g no=23

float: in float you can store decimal type data.

e.g no=4.87

Complex: in this you can store complex type data real and imaginary

e.g 2+3j

2. Sequential Type:- list,tuple,dictionary,string

String: string in python is sequential collection of one or more **Unicode characters**

List: list is collection of heterogeneous data, it is ordered and indexed,

It is mutable, list can be defined with help of [ ]

Tuple: tuple is collection of heterogeneous data,it is ordered and indexed,

It is immutable,tuple can be defined with help of ( )

Dictionary: it is maping data type which stores data in the form of key value pairs

Keys in dictionary are unique ,dictionary is defined using {}

Set: set is collection of heterogeneous data ,it is unordered and unindexed, in set you can add or remove elements, empty can be defined using set using set() ,set elements are defined in {}

7. what is dictionary comprehension? give an example.

Dictionary comprehension is a technique for creating Python dictionaries in one line. The method creates dictionaries from iterable objects, such as a list, tuple, or another dictionary. Dictionary comprehension also allows filtering and modifying key-value pairs based on specified conditions.

Syntax

result = {key: value for item in iterable}

e.g:

```
my_dictionary = {"a": 1, "b": 2, "c": 3, "d": 4}
new_dictionary = {my_dictionary[key]:key for key in my_dictionary}
print(new_dictionary)
```

output:

{1:'a',2:'b',3:'c',4:'d'}

8. Difference between Shallow copy and Deep copy.

Deep Copy:

data is remains same no new data is created after copy,

the reference of data is given to the new object hence now the 2 objects are pointing to same memory location

hence changes in new object reflects to original object

eg:

```
l=[1,2,3]
l2=l
print(id(l))
print(id(l2))
```

l2[0]=90
print(l)
print(l2)
output:
2576617201728
2576617201728
[90, 2, 3]
[90, 2, 3]

Shallow Copy: data is same but new copy of that data is created in another location
Hence new_object is pointing to another data location which newly created
Hence changes in new_object doesn't affetect to original object
e.g:
l=[1,2,3]
l2=l.copy()
print(id(l))
print(id(l2))
l2[0]=90
print(l)
print(l2)
output:
2938346561600
2938346709504
[1, 2, 3]
[90, 2, 3]

9. What is difference between append() and extend() methods.

4-append and extend,

```
8
9  list = [10, 20, 9.5, "lava", True, 40, 80]
10
11 # ---------------------append()-------------------------
12 #it add one or iterable at end of list
13 # it adds an entire iterable as a single element on one index
14 # list length is increased by 1
15 list.append([8,9])
16 print(list,len(list))
17
18 # -------------------------extend()-------------------------
19 # it add one or iterable at emd of list
20 # it adds an entrire iterable on different index
21 # list length is increased by number of elements
22 list.extend([10,20,40,67])
23 print(list,len(list))
24
25
```

input

```
[10, 20, 9.5, 'lava', True, 40, 80, [8, 9]] 8
[10, 20, 9.5, 'lava', True, 40, 80, [8, 9], 10, 20, 40, 67] 12
```

10. What are python modules?

Python Module is a file that contains functions, classes and variables.
We can import the functions, and classes defined in a module to another python sourse file using the **import statement** to the top
Syntax
Import <module_name>
This will only import the module
To import functions or classes of module we have to use dot(.) oprator
e.g
import math
math.floor(4.9)

Python From import statement: without importing whole module you can import particular function or class from module
Eg:
from math import floor
print(floor(4.9))
output:
4

11. Explain all file processing modes.

In Python, the 'open()' function accepts various modes for working with files. Each mode determines how the file will be opened and what operations are allowed. Here are the commonly used file modes:
1. **Read Mode ('r'):** This is the default mode for opening files. It allows you to read the contents of a file.

> *Example:*
>
> ```
> file = open('file.txt', 'r')
> content = file.read()
> print(content)
> file.close()
> ```

2. **Write Mode ('w'):** This mode opens a file for writing. If the file exists, its contents are overwritten. If it doesn't exist, a new file is created.

> *Example:*
>
> ```
> file = open('file.txt', 'w')
> file.write("Hello, World!\n")
> file.write("This is a new line.")
> file.close()
> ```

3. **Append Mode ('a'):** This mode opens a file for appending new content. The new content is added to the end of the file. If the file doesn't exist, a new file is created.

> ***Example:***
>
> ```
> file = open('file.txt', 'a')
> file.write("This line will be appended.\n")
> file.write("So will this one.")
> file.close()
> ```

4. **Binary Mode ('b'):** This mode is used for working with binary files, such as images, audio files, etc. It's often combined with read (`'rb'`) or write ('wb') modes.

> ***Example (Reading a binary file):***
>
> ```
> file = open('image.jpg', 'rb')
> data = file.read()
> # Process the binary data
> file.close()
> ```

5. **Read and Write Mode ('r+'):** This mode allows reading from and writing to a file. It doesn't toverrite the file, preserving its existing contents.

> ***Example:***
>
> ```
> file = open('file.txt', 'r+')
> content = file.read()
> file.write("This line is added in read and write mode.")
> file.close()
> ```

6. **Write and Read Mode ('w+'):** This mode is similar to 'r+', but it also truncates the file, removing its existing contents.

> ***Example:***
>
> ```
> file = open('file.txt', 'w+')
> file.write("This line is added in write and read mode.")
> content = file.read()
> file.close()
> ```

Section B: Correct the Code (10 Questions)

```
# 1.Fix the code to display the correct sum of x and y
x = "10"
y = 20
ANS:
print(int(x)+y)

# 2.  Fix the code to close the file properly
# file = open("example.txt", "r")
# file.write("Hello, World!")
# file.close()
# print("File written successfully!")
# ANS:
# file = open("example.txt", "w")
# file.write("Hello, World!")
# file.close()
# print("File written successfully!")


# 3. Fix this code to properly handle division by zero:
# def divide(a, b):
#     return a / b
# print(divide(10, 0))

# ANS
# def divide(a, b):
#    try:
#      return a / b
#    except:
#      return 'Divison by zero is not possible'
# print(divide(10, 0))


# 4. Fix this code to count the number of vowels in a string:
# def count_vowels(string):
#    vowels = "aeiou"
#    count = 0
#    for char in string:
#        if char not in vowels:
```

```
#             count += 1
#     return count

# print(count_vowels("hello"))
# ANS:
# def count_vowels(string):
#     vowels = "aeiou"
#     count = 0
#     for char in string:
#         if char in vowels:
#             count += 1
#     return count
# print(count_vowels("hello"))


# 5. Fix the function to calculate the area of a rectangle
# def calculate_area(length, breadth):
#     area = length * breadth
#     print("The area of the rectangle is: ", area)
# calculate_area(10)
# ANS:
# def calculate_area(length, breadth=10):
#     area = length * breadth
#     print("The area of the rectangle is: ", area)
# calculate_area(10)
# ANS:
# The area of the rectangle is:  100


# 6.  Fix this code to handle both integers and strings in a list
# data = [1, 2, "three", 4]
# total = 0
# for item in data:
#     total += item
# print(total)
# ANS:
# data = [1, 2, "three", 4]
# total = 0
# for item in data:
#     try:
#         total += int(item)
#     except:
```

```
#         print("Entered string insted of number",item)
# print(total)
# output:
# Entered string insted of number three
# 7

# 7.
# data = [1, "banana", 3, "apple", 2]
# data.sort()
# print(data)
# ANS:
# data=[1,'banana',3,'apple',2]
# try:
#     data.sort()
# except:
#     print("only collections of  intergers or string can be sorted ")
# output:
# only collections of  intergers or string can be sorted



# 8. Fix this code to handle both integers and strings in a list

# data = [1, 2, "three", 4]
# total = 0
# for item in data:
#     total += item
# print(total)
# ANS:
# data = [1, 2, "three", 4]
# total = 0
# for item in data:
#     try:
#         total += int(item)
#     except:
#         print("Entered string insted of number",item)
# print(total)
# output:
# Entered string insted of number three
# 7
```

```
# 9.
# def factorial(n):
#     result = 1
#     for i in range(n):
#         result *= i
#     return result
# print(factorial(5))
# ANS:
# def factorial(n):
#     result = 1
#     for i in range(1,n+1):
#         result *= i
#     return result
# print(factorial(5))
# output:120

# 10. Fix this nested loop structure to create a diamond pattern correctly:
# *****
# ****
# ***
# **
# *
# **
# ***
# ****
# *****
# n = 5
# for i in range(n,0,-1):
#     for j in range(i):
#         print("*", end="")
#     print()
# for i in range(1, n):
#     for j in range(i + 1):
#         print("*", end="")
#     print()
```

Section C: Write Code For (10 Questions)

```
# 1. Write a python program to swap two variables without temp variable.
# n1=int(input("Enter no1 "))
# n2=int(input("Enter no2 "))
# print("befor swap",n1,n2)
# n1=n1+n2
# n2=n1-n2
# n1=n1-n2
# print("After swap",n1,n2)
# output:
# Enter no1 10
# Enter no2 20
# befor swap 10 20
# After swap 20 10


# 2. Write a python program to check if number is positive, negative, zero.
# n1=int(input("enter no "))
# if n1==0:
#     print("Zero")
# elif n1>0:
#     print("Number is positive")
# else:
#     print("Number is negative")
# output:
# enter no -19
# Number is negative


# 3. Write a Python function to return the Second maximum number from list.
# lst=[10,20,40,50,60,3,5,6]
# l=list(set(lst))

# for i in range(len(l)):
#     for j in range(len(l)-1):
#         t=0
#         if l[j]>l[j+1]:
#             temp=l[j]
#             l[j]=l[j+1]
#             l[j+1]=temp
# new_lst=l[::-1]
```

```
# print(new_lst[1])
# output:
# 50


# 4. Write a Python program to print the first 10 numbers of the Fibonacci sequence using a loop.
# n1=0
# n2=1
# print(n1,end=" ")
# print(n2,end=" ")
# for i in range(8):
#     c=n1+n2
#     print(c,end=" ")
#     n1=n2
#     n2=c
# output:
# 0 1 1 2 3 5 8 13 21 34


# 5. Write a Python program that opens a file, reads the first line, and then appends a new line to the file.
# with open('lect.txt','r') as f:
#   print(f.readline())
# with open('lect.txt','a') as f:
#   f.write("I am new line")
#   f.close()
# # output:
# # Hello World


# 6. Write a Python program using a loop that prints a multiplication table for a given number.
# n=int(input("Enter no "))
# for i in range(1,11):
#     print(n,'X',i,'=',n*i)
# output:
# Enter no 2
# 2 X 1 = 2
# 2 X 2 = 4
# 2 X 3 = 6
# 2 X 4 = 8
# 2 X 5 = 10
# 2 X 6 = 12
```

```
# 2 X 7 = 14
# 2 X 8 = 16
# 2 X 9 = 18
# 2 X 10 = 20


# 7. Write a Python function that accepts a string and returns the number of vowels in the string.
# v='aioueAIOUE'
# s=input("Enter string ")
# c=0
# for i in s:
#     if i in v:
#         c+=1
# print(c)
# output:
# Enter string samarthe
# 3


# 8. Write a Python program to print the following half-diamond star pattern:
# *
# **
# ***
# ****
# *****
# ****
# ***
# **
# *
# n=int(input("enter no "))
# for i in range(n):
#     for j in range(i+1):
#         print("*",end="")
#     print()
# for i in range(n-1,0,-1):
#     for j in range(i):
#         print('*',end="")
#     print()
```

```
# 9. Write a Python program to print a list of prime numbers between 200-250.

# for i in range(200,250):
#     p=1
#     for j in range(2,i):
#         if i%j==0:
#             p=0
#     if p==1:
#         print(i)
# output:
# 211
# 223
# 227
# 229
# 233
# 239
# 241


# 10. Write a Python program that checks if a given number is a perfect number (a number is perfect if it is equal to the sum of its proper divisors).
# n=int(input("Enter no "))
# sum=0
# for i in range(1,n):
#     if n%i==0:
#         sum+=i

# if sum==n:
#     print('Perfect number')
# else:
#     print('Not a perfect number')
# output:
# Enter no 23
# Not a perfect number
# Enter no 28
# Perfect number
# Enter no 6
# Perfect number
```