# Python Assignment

## Module – 1 (SDLC)

### 1. What is Software ?

Software is a set of programs (sequence of instructions) that allows the users to perform a well-defined function or some specified task.

A Software is instructions that tell a computer what to do. Software comprises the entire set of programs, procedures, and routines associated with the operation of a computer system.

### 2. What are the types of Applications ?

There are 2 types of applications:

Standalone Application and Web Applications

I) Standalone Application :

The application we are installing on our computer is called a standalone application. To work with any application, if you install that software into your computer then it is called a standalone application.

II) Web Applications :

Without installing any software, we are working with the software called a web application

### 3. What is programing ?

Programming is the process of creating a set of instructions that tell a computer how to perform a task. Programming can be done using a variety of computer programming languages, such as JavaScript, Python, and C++.
Computer programming is the process of designing and writing computer programs. As a skill set, it includes a wide variety of different tasks and techniques.

### 4. What is Python ?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum. It was originally released in 1991.
Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.

## Module – 2 (Fundamentals of python)

1. **Write a Python program to check if a number is positive, negative or zero.**

```
num = float(input("Enter Number: "))
if num > 0:
  print("This Number is Positive")
elif num == 0:
  print("This Number is Zero")
else:
  print("This Number is Negative")
```

2. **Write a Python program to get the Factorial number of given number.**

```
num = int(input("Enter Number: "))
factorial = 1
if num < 0:
  print(" This is not factorial no")
elif num == 0:
  print("The factorial of 0 is 1")
else:
  for i in range(1,num + 1):
     factorial = factorial*i
  print(f'The factorial of {num} is {factorial}')
```

3. **Write a Python program to get the Fibonacci series of given range.**

```
num = int(input("Enter number for Fibonacci series: "))
num1 = 0
num2 = 1
another_number = 0
```

```
count = 1
while(count <= num):
    print(another_number, end=" ")
    count += 1
    num1 = num2
    num2 = another_number
    another_number = num1 + num2
    add = num1 + num2
```

**4. How memory is managed in Python?**

Memory management in Python involves a private heap containing all Python objects and data structures. The management of this private heap is ensured internally by the Python memory manager. The Python memory manager has different components which deal with various dynamic storage management aspects, like sharing, segmentation, preallocation or caching.

**5. What is the purpose continue statement in python?**

The continue keyword is used to end the current iteration in a for loop (or a while loop), and continues to the next iteration.

Program control is passed from the continue statement to the end of the loop body. A continue statement can only appear within the body of an iterative statement, such as do , for , or while.

**6. Write python program that swap two number with temp variable and without temp variable.**

```
a=int(input("enter value of a :"))
b=int(input("enter value of b :"))
b=a+b
a=b-a
b=b-a
print("a =",a)
print("b =",b)
```

7. **Write a Python program to find whether a given number is even or odd, print out an appropriate message to the user.**

```
num = int(input("Enter Number: "))
if (num % 2) == 0:
   print("Number is Even")
else:
   print("Number is Odd")
```

8. **Write a Python program to test whether a passed letter is a vowel or not.**

```
char = input("Enter a character: ")
vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
if char in vowels:
    print(f'The character {char} is a vowel')
else:
    print(f'The character {char} is a consonant')
```

9. **Write a Python program to sum of three given integers. However, if two values are equal sum will be zero.**

```
num1 = int(input("Enter first Number: "))
num2 = int(input("Enter second Number: "))
num3 = int(input("Enter third Number: "))

if num1 == num2 or num2 == num3 or num3 == num1:
    sum = 0
else:
```

```
    sum = (num1 + num2 + num3)
print(f'The sum is : {sum}')
```

**10. Write a Python program that will return true if the two given integer values are equal or their sum or difference is 5.**

```
num1 = int(input("Enter first Number: "))
num2 = int(input("Enter second Number: "))
def result(num1, num2):
    if(num1 == num2 or abs(num1-num2) == 5 or abs(num1+num2) == 5):
        return True
    else:
        return False
print(result(num1, num2))
```

**11. Write a python program to sum of the first n positive integers.**

```
num = int(input("Enter a positive integer: "))
sum = num * (num+1) / 2
print(f'positive integers :{sum}')
```

**12. Write a Python program to calculate the length of a string.**

```
string = input("Enter string to check length: ")
count = 0
for s in string:
    count = count+1
print("Length of string is: ", count)
```

**13. Write a Python program to count the number of characters (character frequency) in a string**

```
string = input("Enter String: ")
string = string.lower()
str={}
for i in string:
    if i in str:
        str[i]+=1
    else:
        str[i]=1
print(str)
```

**14. What are negative indexes and why are they used?**

Negative Indexing is used to in Python to begin slicing from the end of the string (“indexing from the end”).

This means the last value of a sequence has an index of -1, the second last -2, and so on. You can use negative indexing as your advantage when you want to pick values from the end (right side) of an iterable.

**15. Write a Python program to count occurrences of a substring in a string.**

```
str1 = 'I am learning python from Tops Technologies'
print('Number of occurrence of string is :',str1.count("o"))
```

**16. Write a Python program to count the occurrences of each word in a given sentence**

```
string=input("Enter string : ")
```

```
word=input("Enter word : ")
a=[]
count=0
a=string.split(" ")
for i in range(0,len(a)):
    if(word==a[i]):
        count=count+1
print("Count of the word is : ",count)
```

**17. Write a Python program to get a single string from two given strings, separated by a space and swap the first two characters of each string.**

```
str1= input("enter value of a :")
str2= input("enter value of b :")
char1 = str2[:2] + str1[2:]
char2 = str1[:2] + str2[2:]
print("After Swaping String :",char1," ",char2
```

**18. Write a Python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing' then add 'ly' instead if the string length of the given string is less than 3, leave it unchanged.**

```
string = input("Enter String :")
if len(string) < 3:
  print(string)
elif string[-3:] == 'ing':
  print(string + 'ly')
else:
  print(string + 'ing')
```

**19. Write a Python program to find the first appearance of the substring 'not' and 'poor' from a given string, if 'not' follows the 'poor', replace the whole 'not'...'poor' substring with 'good'. Return the resulting string.**

```
string = input("enter a string : ")
str_not = string.find('not')
str_poor = string.find('poor')
if str_not < str_poor and str_not != -1:
    str_new = string[:str_not] + 'good' + string[str_poor+4:]
else:
    str_new = string
print(str_new)
```

**20. Write a Python function that takes a list of words and returns the length of the longest one.**

```
list_of_words = input("Enter Words : ")
list = list_of_words.split()
max_length = 0
for word in list:
    if len(word) > max_length:
        max_length = len(word)
print(f'length of the longest one : {max_length}')
```

**21. Write a Python function to reverses a string if its length is a multiple of 4.**

```
string=input("Enter String : ")
if(len(string)%4==0):
    print(f'Reverse string is : {string[::-1]}')
else:
    print("can nor reverse")
```

**22. Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return instead of the empty string.**

```
string=input("Enter String : ")
count = 0
for i in string:
    count = count + 1
newStr = string[ 0:2 ] + string [count - 2: count ]

print("New String : "+ newStr
```

**23. Write a Python function to insert a string in the middle of a string.**

```
string=input("Enter String : ")
middle_string = "Maitri"
temp = string.split()
middle_position = len(temp) // 2
result = temp[:middle_position] + [middle_string] + temp[middle_position:]
result = ' '.join(result)
print("New String : " + str(result))
```

## Module – 3 (Collections, functions and Modules)

1. **What is List? How will you reverse a list?**

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage. list is an ordered and changeable collection of data objects.

The .reverse() method in Python is a built-in method that reverses the list in place.

This simple and quick way to reverse a list in Python requires little memory.

Syntax: list_name. reverse() Here, list_name means you have to write the list's name, which has to be reversed.

2. **How will you remove last object from a list? Suppose list1 is [2, 33, 222, 14, and 25], what is list1 [-1]?**

The method pop() can be used to remove and return the last value from the list or the given index value. If the index is not given, then the last element is popped out and removed.

To delete the last element, we can use the negative index -1. The use of the negative index allows us to delete the last element, even without calculating the length of the list.

25 beacause -1 corresponds to the last index in the list.

3. **Differentiate between append () and extend () methods?**

The append() method in the Python programming language adds an item to a list that already exists. This method is used to add a single element to the end of a list. It takes an argument and appends it as a single item to the list. whereas

the extend() method adds each of the iterable elements which is supplied as a parameter to the end of the original list. This method is used to add multiple elements to the end of a list. It takes an iterable (such as a list, tuple, or string) as an argument and appends each item from the iterable to the list individually.

4. **Write a Python function to get the largest number, smallest num and sum of all from a list.**

```
numList = []

num = int(input("Enter Total Number of List : "))

for i in range(1, num + 1):

    value = int(input("Enter the Value of %d : " %i))

    numList.append(value)

    list_sum = sum(numList)


print("Smallest Number in List is : ", min(numList))

print("Largest Number in List is : ", max(numList))

print("Sum of List is : ", sum(numList))
```

5. **How will you compare two lists?**

The two lists to be compared with each other. We converted those lists into the set and compared each element with the help of == operator. All elements are equal in both lists, then if block executed and printed the result.

```
list1 = [11, 12, 13, 14, 15]

list2 = [12, 13, 11, 15, 14]

a = set(list1)

b = set(list2)

if a == b:

    print("Both List are equal")

else:
```

```
    print("List are not equal")
```

**6. Write a Python program to count the number of strings where the string length is 2 or more and the first and last character are same from a given list of strings.**

```
def count_strings(strings):
   count = 0
   for string in strings:
      if len(string) >= 2 and string[0] == string[-1]:
         count += 1
   return count
strings = ['dmk','mdp','dmd','1998','2023']
result = count_strings(strings)
print("The number of strings are the same:", result)
```

**7. Write a Python program to remove duplicates from a list.**

```
numList = []
num = int(input("Enter Total Number of List : "))
for i in range(1, num + 1):
    value = int(input("Enter the Value of %d : " %i))
    numList.append(value)
print ("The list is : " + str(numList))
numList = list(set(numList))
print ("The list after removing duplicates : " + str(numList))
```

**8. Write a Python program to check a list is empty or not.**

```
numList = []
if len(numList) == 0:
   print('List is Empty ')
else:
   print('List is Not Empty')
```

9. **Write a Python function that takes two lists and returns true if they have at least one common member.**

```
numList1 = []
numList2 = []
num = int(input("Enter Total Number of List 1 : "))
for i in range(1, num + 1):
    value = int(input("Enter the Value of %d : " %i))
    numList1.append(value)
num = int(input("Enter Total Number of List 2: "))
for i in range(1, num + 1):
    value = int(input("Enter the Value of %d : " %i))
    numList2.append(value)

print ("The list is : " + str(numList1))
print ("The list is : " + str(numList2))

for list1 in numList1:
for list2 in numList2:
    if list1 == list2:
        result = True
if result:
    print(result)
    print("Lists have atleast one common member")
else:
    result = False
    print(result)
    print("Lists not have any common member")
```

**10. Write a Python program to generate and print a list of first and last 5 elements where the values are square of numbers between 1 and 30.**

```
def Values():
    l = list()
    for i in range(1,30):
        l.append(i**2)
    print(l[:5])
    print(l[-5:])
Values()
```

**11. Write a Python function that takes a list and returns a new list with unique elements of the first list.**

```
def uniquelist(lst):
  numlist = []
  for num in lst:
    if num not in numlist:
      numlist.append(num)
  return numlist
print("Unique list is :",uniquelist([25,35,42,52,25,36,42,75,88,9,1,52]))
```

12. **Write a Python program to convert a list of characters into a string.**

```
char = ['M','A','I','T','R','I']
str1 = ''.join(char)
print("String is :",str1)
```

**13. Write a Python program to select an item randomly from a list.**

```
import random
numList = []
num = int(input("Enter Total Number of List : "))
```

```
for i in range(1, num + 1):
    value = int(input("Enter the Value of %d : " %i))
    numList.append(value)
print ("The list is : " + str(numList))
numList = list(set(numList))
random_element = random.choice(numList)
print("Random Number is :",random_element)
```

14. **Write a Python program to find the second smallest number in a list.**

```
numList = []
num = int(input("Enter Total Number of List : "))
for i in range(1, num + 1):
    value = int(input("Enter the Value of %d : " %i))
    numList.append(value)
print ("The list is : " + str(numList))
numList = list(set(numList))
sorted_list = sorted(numList)
print("second smallest number in a list : ",sorted_list[1])
```

**15. Write a Python program to get unique values from a list**

```
numList = []
num = int(input("Enter Total Number of List : "))
for i in range(1, num + 1):
    value = int(input("Enter the Value of %d : " %i))
    numList.append(value)
print ("The list is : " + str(numList))
numList = list(set(numList))
print("New List",set(numList))
```

**16. Write a Python program to check whether a list contains a sub list**

```
numList = []
subList = []
num = int(input("Enter Total Number of List : "))
for i in range(1, num + 1):
    value = int(input("Enter the Value of %d : " %i))
    numList.append(value)
num = int(input("Enter Total Number of Sub List : "))
for i in range(1, num + 1):
    value = int(input("Enter the Value of %d : " %i))
    subList.append(value)
print ("The list is : " + str(numList))
print ("The Sub list is : " + str(subList))
res = False
for idx in range(len(numList) - len(subList) + 1):
    if numList[idx: idx + len(subList)] == subList:
        res = True
        break
print("sublist is in list ? : " + str(res))
```

**17. Write a Python program to split a list into different variables.**

```
list = ['Maitri', 'Desai', 'Pandya']
var1, var2, var3 = list
print(var1)
print(var2)
print(var3)
```

**18. What is tuple? Difference between list and tuple.**

Tuples are used to store multiple items in a single variable.

Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.

A tuple is a collection which is ordered and unchangeable.

| Sno | LIST | TUPLE |
|---|---|---|
| 1 | Lists are mutable | Tuples are immutable |
| 2 | The implication of iterations is Time-consuming | The implication of iterations is comparatively Faster |
| 3 | The list is better for performing operations, such as insertion and deletion. | A Tuple data type is appropriate for accessing the elements |
| 4 | Lists consume more memory | Tuple consumes less memory as compared to the list |
| 5 | Lists have several built-in methods | Tuple does not have many built-in methods. |
| 6 | Unexpected changes and errors are more likely to occur | In a tuple, it is hard to take place. |

**19. Write a Python program to create a tuple with different data types.**

```
tup = ('Maitri', 1998, 4.56, True)
for i in tup:
    print(i, 'type is', type(i))
```

**20. Write a Python program to create a tuple with numbers.**

```
tup = 25,36,98,12,74
print(tup)
```

**21. Write a Python program to convert a tuple to a string.**

```
tup = ('M','a','i','t','r','i')
string = ''.join(tup)
print(string)
```

**22. Write a Python program to check whether an element exists within a tuple.**

```
tup = (12,18,66,33,74,20,2)

if 2 in tup:
    print('2 is in tuple')
else:
    print('2 is NOT in tuple')
if 3 in tup:
    print('3 is in tuple')
else:
    print('3 is not in tuple')
```

**23. Write a Python program to find the length of a tuple.**

```
tup = ('Maitri', '24', 2.45,'Desai')
print('Length of tuple is :',len(tup))
```

**24. Write a Python program to convert a list to a tuple.**

```
lst = ['Maitri', 'Desai', 24]
tup = tuple(lst)
print(tup)
```

**25. Write a Python program to reverse a tuple.**

```
tup = ('Maitri', '24', 2.45,'Desai')
reverse_tuple = tup[::-1]
```

```
print(reverse_tuple)
```

**26. Write a Python program to replace last value of tuples in a list.**

```
lst = [('Maitri', '24', 2.45,'Desai'), ('Tops','Technologies')]
new_list = [i[:len(i)-1]+('Pandya',) for i in lst]
print('New List is :',new_list)
```

**27. Write a Python program to find the repeated items of a tuple.**

```
tup = (2, 4, 5, 6, 2, 3, 4, 4, 7 ,5)
repeated_items = []
for i in tup:
   if tup.count(i) > 1:
      repeated_items.append(i)
print('Repeted Item in tuplr is :',set(repeated_items))
```

**28. Write a Python program to remove an empty tuple(s) from a list of tuples.**

29. 
```
lst = [('Maitri', '24', 2.45,''),(), ('Tops','Technologies')]
new_list = [i for i in lst if i]
print('New List is :',new_list
```

30. **Write a Python program to convert a list of tuples into a dictionary.**

The program converts two tuples into a dictionary using the map() function and prints the resulting dictionary. The map() function applies the lambda function to pair up corresponding elements from the two tuples and return the resulting key-value pairs, which are then used to construct the dictionary.

**31. How will you create a dictionary using tuples in python?**

```
import operator
dicts = dic = {'A' : "maitri", 'B' : "Pandya"}
print('Dictionary : ',dicts)
```

```
sorted_dicts = sorted(dicts.items(), key=operator.itemgetter(1))
print('Dictionary in ascending order by value : ',sorted_dicts)
sorted_dicts = dict( sorted(dicts.items(), key=operator.itemgetter(1),reverse=True))
print('Dictionary in descending order by value : ',sorted_dicts)
```

**32. Write a Python script to sort (ascending and descending) a dictionary by value.**

```
import operator
dicts = dic = {'A' : "maitri", 'B' : "Pandya"}
print('Dictionary : ',dicts)
sorted_dicts = sorted(dicts.items(), key=operator.itemgetter(1))
print('Dictionary in ascending order by value : ',sorted_dicts)
sorted_dicts = dict( sorted(dicts.items(), key=operator.itemgetter(1),reverse=True))
print('Dictionary in descending order by value : ',sorted_dicts)
```

**33. Write a Python script to concatenate following dictionaries to create a new one.**

```
dicts1={"Name":"Maitr" , "Age":24}
dicts2={"DOB": "10 Dec", "Gender": "Female"}
dicts3={"Town":"Vadodara"}
dicts4 = {}
for dicts in (dicts1, dicts2, dicts3): dicts4.update(dicts)
print(dicts4)
```

**34. Write a Python script to check if a given key already exists in a dictionary.**

```
Adict = {"Name":"Maitr", "Surname":"Desai" , "Age":24}
print("dictionary : ",Adict)
check_key = "Desai"
print("Key is :",check_key)
if check_key in Adict:
```

```
  print("Key is Present.")
else:
  print("Key is not Present.")
```

35. **How Do You Traverse Through A Dictionary Object In Python?**

When we use a for loop to traverse any iterable object, internally it uses the iter() method to get an iterator object, which further uses the next() method to iterate over. This method raises a StopIteration to signal the end of the iteration.

To iterate through the dictionary's keys, utilise the keys() method that is supplied by the dictionary. An iterable of the keys available in the dictionary is returned. Then, as seen below, you can cycle through the keys using a for loop.

**36. How Do You Check The Presence Of A Key In A Dictionary?**

The get() method is a dictionary method that returns the value of the associated key. If the key is not present it returns either a default value (if passed) or it returns None. Using this method we can pass a key and check if a key exists in the python dictionary.

37. **Write a Python script to print a dictionary where the keys are numbers between 1 and 15.**

```
num=int(input("Enter the Limit : "))
dists = dict()
for x in range(1,num+1):
   dists[x]=x**2
print(dists)
```

**38. Write a Python program to check multiple keys exists in a dictionary**

```
Adict = {"Name":"Maitr", "Surname":"Desai" , "Age":24, "Vadodara":"City"}
print(Adict.keys() >= {'Surname', 'Name'})
print(Adict.keys() >= {'Age', 'City'})
print(Adict.keys() >= {'Pandya', 'Desai'})
```

39. **Write a Python script to merge two Python dictionaries**

```
dict1 = {"Name":"Maitr", "Surname":"Desai"}
dict2 = {"Age":24, "Vadodara":"City"}
for key in dict2:
   dict1[key] = dict2[key]
print(dict1)
```

**40. Write a Python program to map two lists into a dictionary**

```
dict_key = ['Name', 'Surname', 'Age']
dict_values = ['Maitri','Surname', 24]
dict1 = dict(zip(dict_key, dict_values))
print(dict1)
```

**41. Write a Python program to combine two dictionary adding values for common keys. d1 = {'a': 100, 'b': 200, 'c':300} o d2 = {'a': 300, 'b': 200,'d':400} Sample output: Counter ({'a': 400, 'b': 400,'d': 400, 'c': 300}).**

```
from collections import Counter
d1 = {'a': 100, 'b': 200, 'c':300}
d2 = {'a': 300, 'b': 200, 'd':400}
d = Counter(d1) + Counter(d2)
print(d)
```

42. **Write a Python program to print all unique values in a dictionary.**

```
Lst = [{"Name":"Maitri"}, {"Surname":"Desai"}, {"Age":24}, {"Vadodara":"City"}]
print("Original List : ",Lst)
unique_value = set( val for dic in Lst for val in dic.values())
```

```
print("Unique Values : ",unique_value)
```

**43. Why Do You Use the Zip () Method in Python?**

Python's zip() function creates an iterator that will aggregate elements from two or more iterables. You can use the resulting iterator to quickly and consistently solve common programming problems, like creating dictionaries.

Zip is an in-built function in Python used to iterate over multiple iterables.

**44. Write a Python program to create and display all combinations of letters, selecting each letter from a different key in a dictionary. Sample data: {'1': ['a','b'], '2': ['c','d']}**

**Expected Output: ac ad bc bd**

```
import itertools
dicts ={'1':['a','b'], '2':['c','d']}
for combo in itertools.product(*[dicts[k] for k in sorted(dicts.keys())]):
    print(''.join(combo))
```

**45. Write a Python program to find the highest 3 values in a dictionary**

```
dicts = {'a': 75, 'b': 22, 'c': 42, 'd': 85, 'e': 12, 'f': 96, 'x':0 }
sorted_list = sorted(set(dicts.values()))
print("the highest 3 values are", sorted_list[-3:])
keys_highest_values = []
for key in dicts:
    if dicts[key] in sorted_list[-3:]:
        keys_highest_values.append(key)
print("their keys are (not in order)", keys_highest_values)
```

**46. Write a Python program to combine values in python list of dictionaries. Sample data: [{'item': 'item1', 'amount': 400}, {'item': 'item2', 'amount': 300}, o {'item': 'item1', 'amount': 750}]**

**Expected Output: Counter ({'item1': 1150, 'item2': 300})**

```
from collections import Counter

lst = [{'item': 'item1', 'amount': 400}, {'item': 'item2', 'amount': 300}, {'item': 'item1', 'amount': 750}]

result = Counter()

for d in lst:

    result[d['item']] += d['amount']

print(result)
```

**47. Write a Python program to create a dictionary from a string. Note: Track the count of the letters from the string. Sample string: 'w3resource'**

**Expected output: {'3': 1,’s’: 1, 'r': 2, 'u': 1, 'w': 1, 'c': 1, 'e': 2, 'o': 1}**

```
from collections import defaultdict, Counter

str1 = '3sruwceo'

my_dict = {}

for letter in str1:

    my_dict[letter] = my_dict.get(letter, 0) + 1

print(my_dict)
```

**48. Write a Python function to calculate the factorial of a number (a nonnegative integer)**

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

factorial = 1

if num < 0:

   print(" Factorial does not exist")

elif num == 0:

   print("The factorial of 0 is 1")

else:

   for i in range(1,num + 1):

       factorial = factorial*i
```

```
print(f'The factorial of :{num} is {factorial}')
```

49. **Write a Python function to check whether a number is in a given range**

```
def rng(n):
    if n in range(3,9):
        print( " %s is in the range"%str(n))
    else :
        print(" not in range.")
rng(5)
```

50. **Write a Python function to check whether a number is perfect or not.**

```
num=int(input("Enter  number: "))
sum1=0
for i in range(1,num):
    if (num%i==0):
        sum1=sum1+i
if(sum1==num):
    print("number is a perfect")
else:
    print("number is not a perfect")
```

**51. Write a Python function that checks whether a passed string is palindrome or not**

```
def str_palindrome(s):
    if len(s) < 1:
        return True
    else:
        if s[0] == s[-1]:
            return str_palindrome(s[1:-1])
        else:
```

```
        return False
string = str(input("Enter string:"))
if(str_palindrome(string)==True):
   print("String is palindrome")
else:
   print("String is not palindrome")
```

52. **How Many Basic Types Of Functions Are Available In Python?**

There are two types of functions in python:

- **User-Defined Functions** - these types of functions are defined by the user to perform any specific task. These functions are defined by a programmer to perform any specific task or to reduce the complexity of big problems and use that function according to their need.
- **Built-in Functions** - These are pre-defined functions in python. Built-in functions are already defined in python. A user has to remember the name and parameters of a particular function. Since these functions are pre-defined, there is no need to define them again.

53. **How can you pick a random item from a list or tuple?**

Generate a random item from the tuple using random. choice() method(This function returns a random element from the specified sequence i.e tuple here) by passing the input tuple as an argument to the choice() function. Print the generated random tuple item.

**54. How can you pick a random item from a range?**

Use the import keyword, to import the random module. the random.randrange() function(Returns a random number within the specified range) to generate a random number within the given range by passing minimum, and maximum numbers as arguments.

**55. How can you get a random number in python?**

Python random. seed() function is used to save the state of a random function so that it can generate some random numbers in Python on multiple executions of the code on the same machine or on different machines (for a specific seed value).

**56. How will you set the starting value in generating random numbers?**

The seed() method is used to initialize the random number generator. The random number generator needs a number to start with (a seed value), to be able to generate a random number. By default the random number generator uses the current system time.

**57. How will you randomizes the items of a list in place?**

The shuffle() method randomizes the items of a list in place.

58. **Write a Python program to read a random line from a file.**

```
import random
def random_file(fname):
    lines = open(fname).read().splitlines()
    return random.choice(lines)
print(random_file('hello.txt'))
```

**59. Write a Python program to convert degree to radian**

```
pi=3.14159265359
degree = float(input("Input degree: "))
radian = degree*(pi/180)
print(radian)
```

**60. Write a Python program to calculate the area of a trapezoid**

```
Base1 = float(input('Enter the First Base of a Trapezoid : '))
Base2 = float(input('Enter the Second Base of a Trapezoid : '))
Height_of_Trapezoid= float(input('Enter the Height of a Trapezoid : '))
area = 0.5 * (Base1 + Base2) * Height_of_Trapezoid
print("The Area of a trapezoid : ",area)
```

# Module – 4 (Advance python programming)

1. **What is File function in python? What is keywords to create and write file.**

file object provides methods and attributes to access and manipulate files. Using file objects, we can read or write any files. Whenever we open a file to perform any operations on it, Python returns a file object. To create a file object in Python use the built-in functions, such as open() and os.popen().

IOError exception is raised when a file object is misused, or file operation fails for an I/O-related reason. For example, when you try to write to a file when a file is opened in read-only mode.

To create a new file in Python, use the open() method, with one of the following parameters: "x" - Create - will create a file, returns an error if the file exist. "a" - Append - will create a file if the specified file does not exist. "w" - Write - will create a file if the specified file does not exist.

2. **Write a Python program to read an entire text file.**

```
with open('hello.txt') as f:
    read_data = f.read()
    print(read_data)
print(f.closed)
```

3. **Write a Python program to append text to a file and display the text.**

```
f = open('hello.txt', 'a')
f.write('i am doing assignment of python ')
f.close()
f = open('hello.txt', 'r')
print(f.read())
f.close()
```

4. **Write a Python program to read first n lines of a file.**

```
n = 2
f = open('hello.txt')
for i in range(n):
```

```
    print(f.readline())
f.close()
```

5. **Write a Python program to read last n lines of a file.**

```
n = 4
f = open('hello.txt')
lines = list(f)
for line in lines[-n:]:
    print(line)
f.close()
```

6. **Write a Python program to read a file line by line and store it into a list Write a Python program to read a file line by line store it into a variable.**

```
with open("hello.txt") as f:
    content_list = [x.strip() for x in content_list]
print(content_list)
```

7. **Write a python program to find the longest words.**

```
with open('hello.txt') as f:
    lines = list(f)
    longest = ''
    for line in lines:
        for word in line.split():
            if len(longest) < len(word):
                longest = word
    print(f'Longest word in file is : {longest}')
```

8. **Write a Python program to count the number of lines in a text file.**

```
with open('hello.txt') as f:
    print("Total lines iin file is :",len(list(f)))
```

9. **Write a Python program to count the frequency of words in a file.**

```
from collections import Counter
def total_count(fname):
    with open(fname) as f:
        return Counter(f.read().split())
print("Number of words in file :",total_count("hello.txt"))
```

**10. Write a Python program to write a list to a file.**

```
lst = [1, 2, 3, 4]
with open('hello_1.txt', 'w+') as f:
  f.write(' '.join(map(str, lst)))
```

11. **Write a Python program to copy the contents of a file to another file.**

```
f = open('hello.txt')
f1 = open('hello_1.txt', 'a')
f1.write(f.read())
f.close()
f1.close()
```

**12. Explain Exception handling? What is an Error in Python?**

An exception in Python is an incident that happens while executing a program that causes the regular course of the program's commands to be disrupted. When a Python code comes across a condition it can't handle, it raises an exception. An object in Python that describes an error is called an exception.

The most common reason of an error in a Python program is when a certain statement is not in accordance with the prescribed usage. Such an error is called a syntax error. The Python interpreter immediately reports it, usually along with the reason.

13. **How many except statements can a try-except block have? Name Some built-in exception classes:**

More than zero except statements can a try-except block can havebuilt-in exceptions are SyntaxError, ValueError, IOError, KeyboardInterrupt, ImportError, EOFError, ZeroDivisionError, IndexError, NameError, IndentationError, TypeError,and OverFlowerror.

14. **When will the else part of try-except-else be executed?**

The else part is executed when no exception occurs.

15. **Can one block of except statements handle multiple exception?**

try-except blocks can be used to catch and respond to one or multiple exceptions. In cases where a process raises more than one possible exception, they can all be handled using a single except clause.

16. **When is the finally block executed?**

The finally block always executes when the try block exits. This ensures that the finally block is executed even if an unexpected exception occurs.

17. **What happens when „1“== 1 is executed?**

evaluates to False and does not raise any exception.

18. **How Do You Handle Exceptions With Try/Except/Finally In Python? Explain with coding snippets.**

- Try: This block will test the excepted error to occur.
- Except: Here you can handle the error.
- Else: If there is no exception then this block will be executed.
- Finally: Finally block always gets executed either exception is generated or not.

```
try:
    # Some Code....

except:
    # optional block
    # Handling of exception (if required)

else:
    # execute if no exception

finally:
   # Some code .....(always executed)
```

19. **Write python program that user to enter only odd numbers, else will raise an exception.**

```
def odd_number():
    while True:
        try:
            number = int(input("Enter an odd number: "))
            if number % 2 != 0:
                return number
            else:
                raise ValueError("Even numbers are not allowed.")
        except ValueError as e:
            print(e)
odd_number = odd_number()
print("You entered:", odd_number)
```

20. **What are oops concepts? Is multiple inheritance supported in java**

Major OOP (object-oriented programming) concepts in Python include Class, Object, Method, Inheritance, Polymorphism, Data Abstraction, and Encapsulation. That was all about the differences, moving ahead let's get an idea of classes and objects.

Java doesn't support multiple inheritance.

21. **How to Define a Class in Python? What Is Self? Give An Example Of A Python Class**

A Class is like an object constructor, or a "blueprint" for creating objects. Create a Class. To create a class, use the keyword class : (ex : ‘class MyClass:’)

self represents the instance of the class. By using the “self” we can access the attributes and methods of the class in python. It binds the attributes with the given arguments. The reason you need to use self. is because Python does not use the @ syntax to refer to instance attributes.

```
class Welcome:
    def detailsData(self, name, age):
        print(f'Welcome {name} your age is {age}')
obj = Welcome()
```

**22. Write a Python class named Rectangle constructed by a length and width and a method which will compute the area of a rectangle**

```
class Rectangle():
    def __init__(self, l, w):
        self.length = l
        self.width  = w
    def rectangle_area(self):
        return self.length*self.width
newRectangle = Rectangle(12, 10)
print("New Rectangle :",newRectangle.rectangle_area())
```

**23. Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle**

```
class Circle():
    def __init__(self, r):
        self.radius = r
    def area(self):
        return self.radius**2*3.14
    def perimeter(self):
        return 2*self.radius*3.14
NewCircle = Circle(8)
print("New Circle Area : ",NewCircle.area())
print("New Circle Parameter : ",NewCircle.perimeter())
```

**24. Explain Inheritance in Python with an example? What is init? Or What Is A Constructor In Python?**

Inheritance allows us to define a class that inherits all the methods and properties from another class. Parent class is the class being inherited from, also called base class. Child class is the class that inherits from another class, also called derived class.

```
class Animal:
    def __init__(self,name,age,salary):
        self.name = name
        self.age = age
        self.salary = salary
    def display(self):
        print("Welcome to parent class")
        print(f'Welcome {self.name} your age is {self.age} and salary is {self.salary}')
class Dog(Animal):
    def show(self):
        print("Dog is playing")
obj = Dog("Rocky",5,8000)
```

The __init__ method is the Python equivalent of the C++ constructor in an object-oriented approach. The __init__ function is called every time an object is created from a class. The __init__ method lets the class initialize the object's attributes and serves no other purpose. It is only used within classes.

The constructor is a method that is called when an object is created. This method is defined in the class and can be used to initialize basic variables. If you create four objects, the class constructor is called four times. Every class has a constructor, but its not required to explicitly define it.

**25. What is Instantiation in terms of OOP terminology?**

Instantiate (a verb) and instantiation (the noun) in computer science refer to the creation of an object (or an “instance” of a given class) in an object-oriented programming (OOP) language. Referencing a class declaration, an instantiated object is named and created, in memory or on disk.

26. **What is used to check whether an object o is an instance of class A?**

The Python's isinstance() function checks whether the object or variable is an instance of the specified class type or data type. For example, isinstance(name, str) checks if name is an instance of a class str .

27. **What relationship is appropriate for Course and Faculty?**

"Course" typically refers to a specific subject or topic of study that students enroll in to gain knowledge and skills. The "Faculty" refers to the group of teachers or instructors who are responsible for delivering the course content and providing guidance to the students.

**28. What relationship is appropriate for Student and Person?**

The relationship between a student and a person can be described as one of a learner and an individual. "person" can refer to anyone who is not specifically identified by their role or occupation, such as a teacher, parent, or mentor.

A student is an individual who is actively engaged in learning, typically within an institution of education .