Experiment No : 2

Dated : 28th Jan

Aim

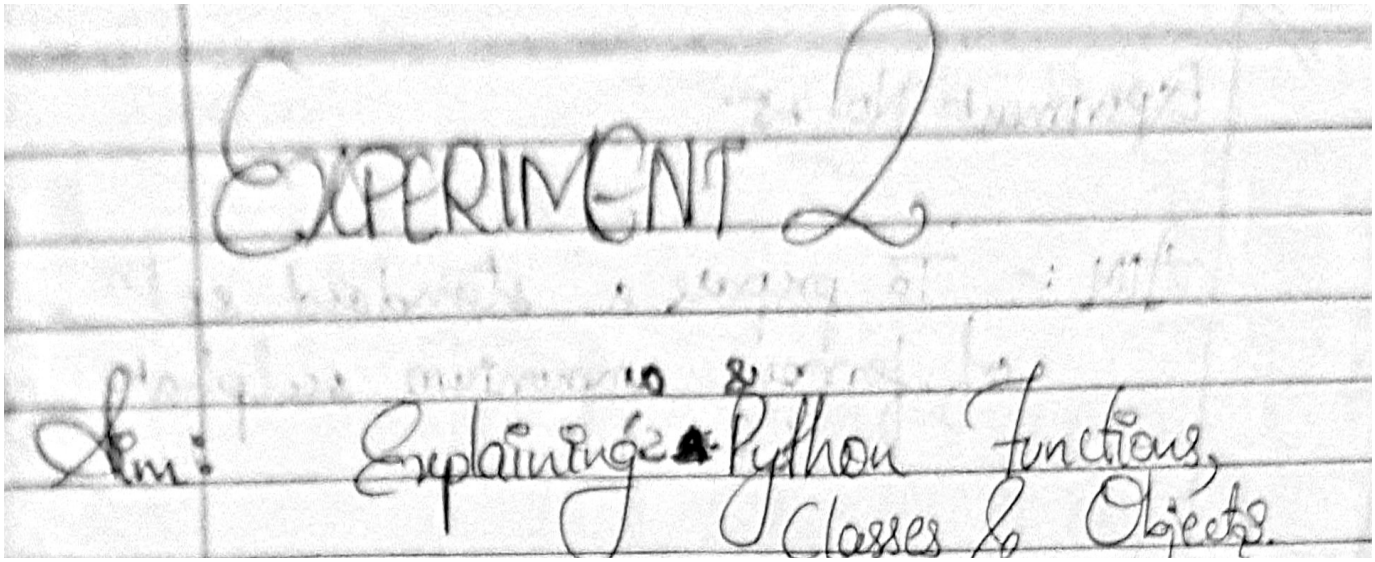
Exploring Python - Functions, Classes & Objects

Theory

- Functions
  - Syntax for defining Function
  - Advantages of function (atleast 5)
  - Types of arguments (enlist them)
  - Difference between Local & Global variables (at least 2)

- Classes & Objects
  - Syntax for defining Class
  - Creating an instance of a class.
  - What is the role of the self keyword? (explain in 1 line)
  - Types of Constructors (enlist them)

Handwritten Theory:



Theory: ① Functions -

Function is a group of related statements that perform a specific task.

ⓐ Syntax:

```
def function_name(parameters)
    statements
    return ...
```

ⓑ Advantages:

1. Reduces duplication of code
2. Reusing is easy
3. Improves clarity
4. Helps decomposing the code.
5. Helps in information hiding and readability.

ⓒ Types of arguments:

1. default arguments
2. keyword arguments
3. positional arguments
4. arbitrary positional arguments
5. arbitrary keyword arguments.

② Difference btw Local & Global

variables:

① Local variable is declared inside a function & Global function is declared outside or using the keyword 'global'.

② Parameter passing is required for local variables whereas it is not necessary for global variables

② Classes & Objects

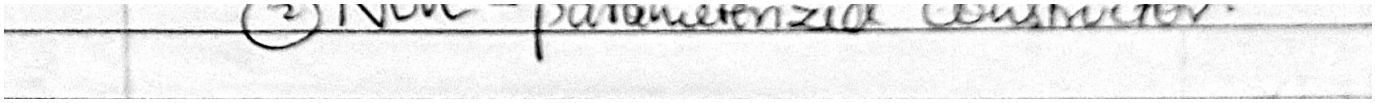
A class is an object constructor like a blueprint for creating an object

ⓐ Syntax: class (class_name):
statement

ⓑ call class name() to create a new instance of the class.

ⓒ 'self' represents the instance of the class, we can access the attributes & methods of the class by using 'self'.

ⓓ Types of constructors:
① Parameterized constructor



Programs to be performed :

Functions: (Attempt any 3)

1. Write a function to display Fibonacci Numbers using Recursion. Invoke the function to display the first 20 numbers from the sequence.

```
# 1. Write a function to display Fibonacci Numbers using Recursion. Invoke the functior

def Fibonacci(input_number):
    if input_number < 0:
        print("Incorrect input")
    elif input_number == 0:
        return 0
    elif input_number == 1 or input_number == 2:
        return 1
    else:
        return Fibonacci(input_number-1) + Fibonacci(input_number-2)

number = int(input("Enter the number of terms you want from the Fibonacci series: "))

for num in range(number):
  print(Fibonacci(num))
```

```
Enter the number of terms you want from the Fibonacci series: 20
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
```

2. Write a function to display the ASCII value of the character passed to it while invocation.

```
#2. Write a function to display the ASCII value of the character passed to it while inv
the_input = input("Enter the number of terms you want from the Fibonacci series: ")

print("The ASCII value of '", the_input , "' is ", ord(the_input) )
```

```
Enter the number of terms you want from the Fibonacci series: z
The ASCII value of ' z ' is  122
```

3. Write a menu driven program to implement a Simple Calculator.

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

```
#4. Write a Python function to check whether a number is perfect or not.
# A perfect number is a positive integer that is equal to the sum of its positive divis

import math

def Perfect_Number(to_check):
  any_array = []
  for num in range( 1 , int(to_check/2) + 1 ):
    if to_check%num == 0:
      any_array.append(num)

  if to_check == 1 or to_check == 0:
    return False

  else:
    sum = 0
    for element in any_array:
      sum = sum +  element

    if sum == to_check:
      print("The number given is a perfect number.")
      return True

    else:
      print("The number given is not a perfect number.")
      return False

to_check = int(input("Enter the number to check : "))
Perfect_Number(to_check)
```

```
Enter the number to check : 28
The number given is a perfect number.
```

```
True
```

5. Write a Python function that prints out the first n rows of Pascal's triangle

6. Write a Python function to check whether a string is a pangram or not.

Hint : Pangrams are words or sentences containing every letter of the alphabet at least once. For example : "The quick brown fox jumps over the lazy dog".

Classes: (Attempt any 2 out of 3)

1.

- Write a Python class named Student with 3 attributes student_id, student_name and student_class (class variable).
- Create a function to display the entire attribute and their values in Student class.
- Create two instances student1, student2 and assign given values to the said instances attributes.
- Print all the attributes of student1, student2 instances with their values

```
#Write a Python class named Student with 3 attributes student_id, student_name and stud
class Student:
  student_id = 0
  student_name = 'None'
  student_class = 'None'

  def enter_data( self , student_id , student_name , student_class):
    self.student_id = student_id
    self.student_name = student_name
    self.student_class = student_class

#Create a function to display the entire attribute and their values in Student class.
  def print_data(self):
    print("\nThe ID of the student is : ", self.student_id)
    print("The Name of the student is : ", self.student_name)
    print("The Class of the student is : ", self.student_class)


#Create two instances student1, student2 and assign given values to the said  instances

student1 = Student()

student1.print_data()
```

```
"""0, None , None for student 1 since they weren't initialized,
Hence default values were printed"""

student2 = Student()

student2.enter_data(69, "Yeah Boi", "NO-ice")
student2.print_data()
```

```
The ID of the student is :  0
The Name of the student is :  None
The Class of the student is :  None

The ID of the student is :  69
The Name of the student is :  Yeah Boi
The Class of the student is :  NO-ice
```

2.

- Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.
- Create an instance and calculate its area & perimeter by invoking the methods.

```
PI = 3.142

# Write a Python class named Circle constructed by a radius and two methods which will
class Circle:
  radius = perimeter = area = 0

  #Create an instance and calculate its area & perimeter by invoking the methods.
  def __init__(self, radius):
    self.radius = radius
    self.perimeter = 2*radius*PI
    self.area = PI*radius*radius

  def print_data(self):
    print("\nThe radius of the circle is : ", self.radius)
    print("The Perimeter of the student is : ", self.perimeter)
    print("The Area of the student is : ", self.area)


my_circle = Circle(7)
my_circle.print_data()
```

```
The radius of the circle is :  7
The Perimeter of the student is :  43.988
The Area of the student is :  153.958
```

3.

- Write a Python class to find the validity of a string of parentheses, '(', ')', '{', '}', '[' and ']'.
- These brackets must be close in the correct order, for example "()" and "()[]{}" are valid but "[)", "({[)]" and "{{{" are invalid.