Experiment #1: Implementation of Basic Python Programs

Aim/Objective:

Implementation of Basic Python Programs.

Description:

Students will learn and understand the basic concepts in Python programming language.

Pre-Requisites:

- Basic Computer Skills
- Basic Mathematics
- Logical Thinking
- Text Editor or Integrated Development Environment (IDE)

Pre-Lab:

1. What is the purpose of implementing basic python programs in this lab?

- To understand core programming concepts like variables, functions, conditionals, and user input.

2. What is python, and how do you comment in Python?

- Python is a simple, high-level programming language. Comments are written with # (e.g., # This is a comment).

3. What are variables in Python? How do you print output in Python?

- Variables store data. You print output using print() (e.g., print("Hello, World!")).

4. How do you take user input in Python? What are data types in Python?

- Use input() to take input. Common data types: int, float, str, bool.

5. How do you create a function in Python? How do you define a conditional statement in Python?

- A function is created with def (e.g., def greet():). Conditionals use if, elif, and else (e.g., if x > 10:).

In-Lab:

Implementation of Basic Python Programs.

1. Write a python Program to Find Largest of 3 Numbers using nested if-else

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

if num1 >= num2:
   if num1 >= num3:
      largest = num1
   else:
      largest = num3
else:
   if num2 >= num3:
      largest = num2
   else:
      largest = num3

print("The largest number is:", largest)
```

OUTPUT:

```
Enter the first number: 1
Enter the second number: 2
Enter the third number: 3
The largest number is: 3.0
```

2. Write a python Program to Swap Two Variables using Third Variable

```
a = float(input("Enter the first variable (a): "))
b = float(input("Enter the second variable (b): "))
temp = a
a = b
b = temp
print("After swapping:")
print("a =", a)
print("b =", b)
```

OUTPUT:

```
Enter the first variable (a): 1
Enter the second variable (b): 2
After swapping:
a = 2.0
b = 1.0
```

3. Write a python Program to find FACTORIAL of a given number.

```
n = int(input("Enter a number to find its factorial: "))
factorial = 1
for i in range(1, n + 1):
   factorial *= i
print("The factorial of", n, "is:", factorial)
```

OUTPUT:

```
Enter a number to find its factorial: 5
The factorial of 5 is: 120
```

4. Write a python Program to find the PRIME NUMBERS in the given range

```
start = int(input("Enter the start of the range: "))
end = int(input("Enter the end of the range: "))
print("Prime numbers between", start, "and", end, "are:")

for num in range(start, end + 1):
   if num > 1:
      is_prime = True
      for i in range(2, int(num**0.5) + 1):
         if num % i == 0:
            is_prime = False
            break
      if is_prime:
         print(num)
```

OUTPUT:

```
Enter the start of the range: 2
Enter the end of the range: 5
Prime numbers between 2 and 5 are:
2
3
5
```

5. Write a python Program to Print Fibonacci Series up to N Term

```
n = int(input("Enter the number of terms: "))
a, b = 0, 1
print("Fibonacci Series up to", n, "terms:")

for _ in range(n):
    print(a, end=" ")
    a, b = b, a + b
```

OUTPUT:

```
Enter the number of terms: 2
Fibonacci Series up to 2 terms:
0 1
```

6. Create a list of integers and *append*, *insert*, and *remove* elements from the list. Access elements using *indexing* and *slicing*

```
my_list = [10, 20, 30, 40, 50]
my_list.append(60)
my_list.insert(2, 25)
my_list.remove(40)
print("Element at index 3:", my_list[3])
print("Slice from index 1 to 4:", my_list[1:5])
```

OUTPUT:

```
Element at index 3: 30
Slice from index 1 to 4: [20, 25, 30, 50]
```

7. Perform the list methods of *append(), extend(), insert(), remove(), pop(), clear(), index(), count(), sort(), reverse(), copy()* on a sample list and observe the changes.

```
my_list = [10, 20, 30, 40, 50]
my_list.append(60)
print("After append(60):", my_list)
my_list.extend([70, 80])
print("After extend([70, 80]):", my_list)
my_list.insert(2, 25)
print("After insert(2, 25):", my_list)
my_list.remove(40)
print("After remove(40):", my_list)
last_element = my_list.pop()
print("After pop():", my_list, "| Popped element:", last_element)
specific_element = my_list.pop(2)
print("After pop(2):", my_list, "| Popped element:", specific_element)
my_list.clear()
print("After clear():", my_list)
my_list = [10, 20, 30, 20, 50, 20]
index_of_20 = my_list.index(20)
print("Index of first 20:", index_of_20)
count_of_20 = my_list.count(20)
print("Count of 20:", count_of_20)
my_list.sort()
print("After sort():", my_list)
my_list.reverse()
print("After reverse():", my_list)
copy_list = my_list.copy()
print("Copy of the list:", copy_list)
```

OUTPUT:

```
After append(60): [10, 20, 30, 40, 50, 60]
After extend([70, 80]): [10, 20, 30, 40, 50, 60, 70, 80]
After insert(2, 25): [10, 20, 25, 30, 40, 50, 60, 70, 80]
After remove(40): [10, 20, 25, 30, 50, 60, 70, 80]
After pop(): [10, 20, 25, 30, 50, 60, 70] | Popped element: 80
```

After pop(2): [10, 20, 30, 50, 60, 70] | Popped element: 25
After clear(): []
Index of first 20: 1
Count of 20: 3
After sort(): [10, 20, 20, 20, 30, 50]
After reverse(): [50, 30, 20, 20, 20, 10]
Copy of the list: [50, 30, 20, 20, 20, 10]

8. Create a tuple of integers and Access elements using indexing and slicing.

```
my_tuple = (10, 20, 30, 40, 50)
print("Element at index 2:", my_tuple[2])
print("Slice from index 1 to 3:", my_tuple[1:4])
```

OUTPUT:

Element at index 2: 30
Slice from index 1 to 3: (20, 30, 40)

9. Create a dictionary with tuples as keys and access and modify the dictionary using these keys.

```
my_dict = {(1, 2): "A", (3, 4): "B", (5, 6): "C"}
print("Value for key (1, 2):", my_dict[(1, 2)])
my_dict[(3, 4)] = "Modified B"
print("After modifying key (3, 4):", my_dict)
my_dict[(7, 8)] = "D"
print("After adding key (7, 8):", my_dict)
```

OUTPUT:

Value for key (1, 2): A
After modifying key (3, 4): {(1, 2): 'A', (3, 4): 'Modified B', (5, 6): 'C'}
After adding key (7, 8): {(1, 2): 'A', (3, 4): 'Modified B', (5, 6): 'C', (7, 8): 'D'}

10. Define a function that takes your name as input and returns a greeting message

```
name = input("Enter your name: ")
greeting = "Hello, " + name + "! Nice to meet you."
print(greeting)
```

OUTPUT:

```
Enter your name: THANOS
Hello, THANOS! Nice to meet
```

Sample VIVA-VOCE Questions (In-Lab):

1. What is the difference between a list and a tuple in Python?

- **List:** Mutable, defined with `[]`.
- **Tuple:** Immutable, defined with `()`.

2. How does Python handle memory management?

- Python uses **reference counting** and a **garbage collector** to manage memory, automatically deallocating objects when no longer in use.

3. Explain the purpose of the self-parameter in Python classes.

- `self` refers to the current instance of the class, allowing access to its attributes and methods.

4. Describe how you can handle exceptions in Python. Provide an example.

- Use `try`, `except`, `else`, and `finally` blocks to handle exceptions. Example:

python Copy code

```python
try:
    # code
except Exception:
    # handle
```

5. What are decorators in Python and how are they used? Provide an example.

- A decorator is a function that modifies another function's behavior. Example:

python Copy code

```python
def decorator(func):
    def wrapper():
        print("Before")
        func()
        print("After")
    return wrapper
```

Post-Lab:

Implementation of Advanced Python Programs.

1. Write a function that takes two numbers and returns their sum, difference, product, and quotient.

```
num1, num2 = float(input("Enter first number: ")), float(input("Enter second number: "))

print("Sum:", num1 + num2)
print("Difference:", num1 - num2)
print("Product:", num1 * num2)

quotient = num1 / num2 if num2 != 0 else "undefined"
print("Quotient:", quotient)
```

OUTPUT:

```
Enter first number: 1
Enter second number: 2
Sum: 3.0
Difference: -1.0
Product: 2.0
Quotient: 0.5
```

2. Implement a *recursive function* to calculate the factorial of a number.

```
n = int(input("Enter a number: "))
result = 1
temp = n

while temp > 1:
    result *= temp
    temp -= 1

print("Factorial:", result)
```

OUTPUT:

```
Enter a number: 5
Factorial: 120
```

3. Define a simple class **Person** with attributes *name* and *age*. Create **instances** of the Person class and print their attributes. Add a method *greet* to the Person class that prints a greeting message. Call the *greet* method on different instances and observe the output.

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print("Hello, my name is " + self.name + " and I am " + str(self.age) + " years old.")

person1 = Person("Alice", 30)
person2 = Person("Bob", 25)

print("Person 1:", person1.name, person1.age)
person1.greet()

print("Person 2:", person2.name, person2.age)
person2.greet()
```

OUTPUT:

```
Person 1: Alice 30
Hello, my name is Alice and I am 30 years old.
Person 2: Bob 25
Hello, my name is Bob and I am 25 years old.
```

4. Create a subclass **Student** that inherits from the **Person** class. Add a new attribute *student_id* and a method *study* to the Student class. Create instances of **Student** and call methods from both **Person** and **Student.**

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print("Hello, my name is " + self.name + " and I am " + str(self.age) + " years
old.")

class Student(Person):
    def __init__(self, name, age, student_id):
        super().__init__(name, age)
        self.student_id = student_id

    def study(self):
        print("Studying...")

student1 = Student("Alice", 30, "S12345")
student2 = Student("Bob", 25, "S67890")

print("Student 1:", student1.name, student1.age, student1.student_id)
student1.greet()
student1.study()

print("Student 2:", student2.name, student2.age, student2.student_id)
student2.greet()
student2.study()
```

OUTPUT:

Student 1: Alice 30 S12345
Hello, my name is Alice and I am 30 years old.
Studying...
Student 2: Bob 25 S67890
Hello, my name is Bob and I am 25 years old.
Studying...

Evaluator Remark (if Any):	**Marks Secured _____ out of 50**
	Signature of the Evaluator with Date