Python Practical file

Submitted by:
Vijay Kumar
Roll no –25201322

Experiment Name	Date
1. Installation and Basic Programs (Variables and Data Types,Arithmetic, Comparison, Assignment, Logical Operators)	25/08/25
2. Operators based programs	01/09/25
3. String Manipulation, Number System and Conversions	08/09/25
4. Operator Precedence, Conditional Statements, Loops (For and While)	15/09/25
5. Nested Loops, Keyword Arguments	22/09/25
6. Program based on Functions	27/09/25
7. Modules and packages	29/09/25
8. String and its operations	13/10/25
9. File Handling	27/10/25
10. Data Structure: Lists,Tuples,Sets	03/11/25
11. Programming Exercises with classes and objects, Inheritance	10/11/25
12. Operator Overloading, Error Handling	17/11/25

In [ ]:

In [1]:
```python
# Operator overloading using + for Student class

class Student:
    def __init__(self, name, marks):
        self.name = name
        self.marks = marks

    def __add__(self, other):
        return self.marks + other.marks


# Creating student objects
s1 = Student("Alice", 85)
s2 = Student("Bob", 90)

total_marks = s1 + s2

print("Total Marks of both students:", total_marks)
```

```
Total Marks of both students: 175
```

In [2]:
```python
# Repeated integer input with exception handling

while True:
    try:
        num = int(input("Enter an integer: "))
        print("You entered:", num)
        break
    except ValueError:
        print("Invalid input, please enter an integer.")
```

```
You entered: 6
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [1]:
```python
# Employee class to track employee details and count

class Employee:
    emp_count = 0   # Class variable to track number of employees

    def __init__(self, name, designation, salary):
        self.name = name
        self.designation = designation
        self.salary = salary
        Employee.emp_count += 1

    def display_details(self):
        print("Name:", self.name)
        print("Designation:", self.designation)
        print("Salary:", self.salary)

    @classmethod
    def total_employees(cls):
        print("Total number of employees:", cls.emp_count)


# Creating employee objects
emp1 = Employee("Alice", "Manager", 60000)
emp2 = Employee("Bob", "Developer", 45000)

# Display employee details
emp1.display_details()
print()
emp2.display_details()
print()

# Display total employees
Employee.total_employees()
```

```
Name: Alice
Designation: Manager
Salary: 60000

Name: Bob
Designation: Developer
Salary: 45000

Total number of employees: 2
```

In [2]:
```python
# Department class
class Department:
    def __init__(self, name):
        self.name = name


# Course class
class Course:
    def __init__(self, name, year, department):
        self.name = name
```

```
        self.year = year
        self.department = department


# Student class
class Student:
    def __init__(self, name, rollno, course, year):
        self.name = name
        self.rollno = rollno
        self.course = course
        self.year = year

    def display_info(self):
        print("Student Name:", self.name)
        print("Roll No:", self.rollno)
        print("Course:", self.course.name)
        print("Course Year:", self.course.year)
        print("Department:", self.course.department.name)
        print("Student Year:", self.year)


# Creating department
dept = Department("Computer Science")

# Creating course
course = Course("Python Programming", 2, dept)

# Enrolling student
student = Student("Rahul", 101, course, 2)

# Display student details
student.display_info()
```

```
Student Name: Rahul
Roll No: 101
Course: Python Programming
Course Year: 2
Department: Computer Science
Student Year: 2
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [4]:
```python
# Remove duplicates from a list

n = int(input("Enter number of elements: "))
numbers = []

for i in range(n):
    num = int(input(f"Enter element {i+1}: "))
    numbers.append(num)

unique_numbers = []

for num in numbers:
    if num not in unique_numbers:
        unique_numbers.append(num)

print("Original List:", numbers)
print("List without duplicates:", unique_numbers)
```

```
Original List: [1, 3, 7, 9, 3]
List without duplicates: [1, 3, 7, 9]
```

In [2]:
```python
# Search an element in a tuple

nums = (10, 20, 30, 40, 50)

search = int(input("Enter a number to search: "))

if search in nums:
    print("Element found at index:", nums.index(search))
else:
    print("Element not present in the tuple")
```

```
Element found at index: 1
```

In [3]:
```python
# Set operations on two sequences

seq1 = list(map(int, input("Enter first sequence of integers: ").split()))
seq2 = list(map(int, input("Enter second sequence of integers: ").split()))

set1 = set(seq1)
set2 = set(seq2)

common_elements = sorted(set1 & set2)
symmetric_diff = sorted(set1 ^ set2)

print("Common elements:", common_elements)
print("Elements in exactly one sequence:", symmetric_diff)
```

```
Common elements: [1, 4, 8]
Elements in exactly one sequence: [3, 5, 6]
```

In [ ]:

In [ ]:

In [1]:
```python
# create_text_file.py
with open("sample.txt", "w") as f:
    f.write("Hello World\n")
    f.write("This\tis a\tfile\n")
    f.write("With spaces,        tabs,\n")
    f.write("and new lines.\n")

print("sample.txt created")

# count_characters.py
tabs = 0
spaces = 0
newlines = 0

with open("sample.txt", "r") as f:
    content = f.read()
    for ch in content:
        if ch == "\t":
            tabs += 1
        elif ch == " ":
            spaces += 1
        elif ch == "\n":
            newlines += 1

print("Tabs:", tabs)
print("Spaces:", spaces)
print("Newlines:", newlines)
```

```
sample.txt created
Tabs: 3
Spaces: 5
Newlines: 4
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [1]:
```
# Remove vowels from a string

text = input("Enter a string: ")
vowels = "aeiouAEIOU"
result = ""

for ch in text:
    if ch not in vowels:
        result += ch

print("String without vowels:", result)
```

```
String without vowels: hll wrld
```

In [2]:
```
# Check if a string is a palindrome

text = input("Enter a string: ")

cleaned = ""
for ch in text:
    if ch.isalnum():
        cleaned += ch.lower()

if cleaned == cleaned[::-1]:
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

```
The string is a palindrome.
```

In [ ]:
```
 Reverse words in a sentence

sentence = input("Enter a sentence: ")

words = sentence.split()
reversed_sentence = ""

for i in range(len(words) - 1, -1, -1):
    reversed_sentence += words[i] + " "

print("Sentence with reversed words:", reversed_sentence.strip())
```

In [ ]:

In [ ]:

In [1]:
```
# Importing random module with alias

import random as rnd

print("Five random integers between 1 and 100:")
for i in range(5):
    print(rnd.randint(1, 100))
```

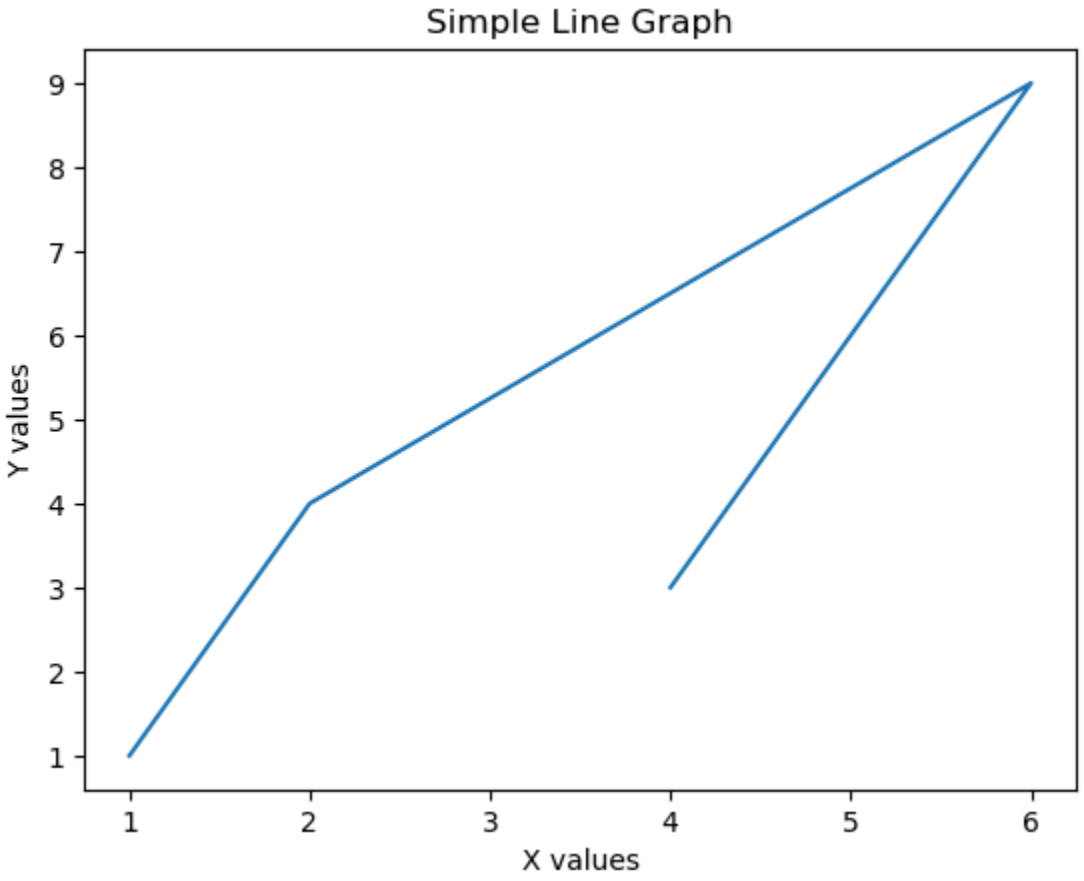
```
Five random integers between 1 and 100:
15
10
21
91
95
```

In [2]:
```
# Plotting a simple line graph

import matplotlib.pyplot as plt

x = [1, 2, 6, 4]
y = [1, 4, 9, 3]

plt.plot(x, y)
plt.xlabel("X values")
plt.ylabel("Y values")
plt.title("Simple Line Graph")
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [1]:
```
# Lambda function to square a number

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

square = lambda x: x * x

print("Square of the number:", square(num))
```

```
Square of the number: 16
```

In [2]:
```
# Recursive function to count digits

def count_digits(n):
    if n == 0:
        return 0
    return 1 + count_digits(n // 10)

num = int(input("Enter a positive integer: "))

if num == 0:
    print("Number of digits: 1")
else:
    print("Number of digits:", count_digits(num))
```

```
Number of digits: 1
```

In [3]:
```
# List input and reverse printing without built-in methods

n = int(input("Enter number of elements: "))
numbers = []

for i in range(n):
    value = int(input(f"Enter element {i+1}: "))
    numbers.append(value)

print("Original List:", numbers)

print("Reversed List:", end=" ")
for i in range(len(numbers) - 1, -1, -1):
    print(numbers[i], end=" ")
```

```
Original List: [2, 4, 5, 6, 7]
Reversed List: 7 6 5 4 2
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [1]:
```
# Printing number pattern using nested loops

for i in range(1, 6):
    for j in range(1, i + 1):
        print(j, end="")
    print()
```

```
1
12
123
1234
12345
```

In [2]:
```
# Multiplication table of a number

n = int(input("Enter a number: "))

print("Multiplication Table of", n)
for i in range(1, 11):
    print(n, "x", i, "=", n * i)
```

```
Multiplication Table of 3
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
```

In [4]:
```
# Function using keyword arguments

def student_info(name, age, course):
    print("Student Information")
    print("Name:", name)
    print("Age:", age)
    print("Course:", course)

# Function call using keyword arguments in any order
student_info(course="Python Programming", name="vijay", age=20)
```

```
Student Information
Name: vijay
Age: 20
Course: Python Programming
```

In [ ]:
```
# Absolute value, square root and cube of a number

import math
```

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

absolute_value = abs(num)
square_root = math.sqrt(abs(num))
cube = num ** 3

print("Absolute Value:", absolute_value)
print("Square Root:", square_root)
print("Cube:", cube)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [5]:
```
# Solving expression using operator precedence

step1 = 3 ** 4
print("Step 1: 3 ** 2 =", step1)

step2 = 2 * step1
print("Step 2: 2 *", step1, "=", step2)

step3 = 7 / 4
print("Step 3: 8 / 4 =", step3)

x = 10 + step2 - step3
print("Final Result: x =", x)
```

```
Step 1: 3 ** 2 = 81
Step 2: 2 * 81 = 162
Step 3: 8 / 4 = 1.75
Final Result: x = 170.25
```

In [6]:
```
# Dictionary storing student names and marks

students = {
    "vipin": 85,
    "pob": 90,
    "Chae": 78
}

# i. Print the dictionary
print("Student Marks Dictionary:", students)

# ii. Print marks of a student entered by the user
name = input("Enter student name: ")

if name in students:
    print("Marks of", name, ":", students[name])
else:
    print("Student not found")
```

```
Student Marks Dictionary: {'Alice': 85, 'Bob': 90, 'Charlie': 78}
Student not found
```

In [7]:
```
# Reverse of a number using while loop

num = int(input("Enter a number: "))
reverse = 0

while num > 0:
    digit = num % 10
    reverse = reverse * 10 + digit
    num //= 10

print("Reversed number:", reverse)
```

```
Reversed number: 3421
```

In [8]:
```python
# Multiplication table using for loop

n = int(input("Enter a number: "))

print("Multiplication Table of", n)
for i in range(1, 11):
    print(n, "x", i, "=", n * i)
```

```
Multiplication Table of 8
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
8 x 10 = 80
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [1]:
```
# Decimal to Binary, Octal and Hexadecimal Conversion

decimal_number = int(input("Enter a decimal number: "))

binary = bin(decimal_number)
octal = oct(decimal_number)
hexadecimal = hex(decimal_number)

print("Binary equivalent:", binary)
print("Octal equivalent:", octal)
print("Hexadecimal equivalent:", hexadecimal)
```

```
Binary equivalent: 0b100010
Octal equivalent: 0o42
Hexadecimal equivalent: 0x22
```

In [2]:
```
# String to float conversion and addition

num_str = input("Enter a floating-point number as a string: ")

num_float = float(num_str)
result = num_float + 10.5

print("Result after adding 10.5:", result)
```

```
Result after adding 10.5: 34.0
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```python
# Calculate area of a triangle using Heron's Formula

a = float(input("Enter first side: "))
b = float(input("Enter second side: "))
c = float(input("Enter third side: "))

# Semi-perimeter
s = (a + b + c) / 2

# Area calculation
area = (s * (s - a) * (s - b) * (s - c)) ** 0.5

print("Area of the triangle is:", area)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

NAME: VIJAY KUMAR

ROLL NO: 25201322

In [3]:
```python
# Displaying different data types using variables and literals

integer_var = 25
float_var = 12.5
string_var = "Python Programming"
boolean_var = True

print("Integer value:", integer_var)
print("Float value:", float_var)
print("String value:", string_var)
print("Boolean value:", boolean_var)

# Literal constants
print("Integer literal:", 100)
print("Float literal:", 3.14)
print("String literal:", "Hello World")
print("Boolean literal:", False)
```

```
Integer value: 25
Float value: 12.5
String value: Python Programming
Boolean value: True
Integer literal: 100
Float literal: 3.14
String literal: Hello World
Boolean literal: False
```

In [4]:
```python
# Using += operator on strings

str1 = "Hello"
str1 += " World"

print("Updated string:", str1)
```

```
Updated string: Hello World
```

In [5]:
```python
# String addition and multiplication

str1 = "Python"
str2 = " Lab"

# Addition (Concatenation)
add_result = str1 + str2

# Multiplication (Repetition)
mul_result = str1 * 3
```

```
print("String Addition Result:", add_result)
print("String Multiplication Result:", mul_result)
```

```
String Addition Result: Python Lab
String Multiplication Result: PythonPythonPython
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: