



DEPARTMENT OF COMPUTER ENGINEERING

Subject: PWP	Subject Code: 22616
Semester: VI	Course: Computer Engineering
Laboratory No: L001	Name of Subject Teacher: Prof. Sangeeta Wankhede
Name of Student: Darshan Bhangale	Roll Id: 22203A0039

Experiment No:	1
Title of Experiment	Install and configure Python IDE

X. Practical Related Question:

1. Write steps for installing Python on window?

○ Download Python Installer:

- Go to the official Python website:
<https://www.python.org/downloads/windows/>. ○ Download the latest version of Python for Windows.

○ Run the Installer:

- Locate the downloaded .exe file and double-click to open it. ○ Check the box "Add Python to PATH" (important). ○ Click "Install Now" to start the installation.

○ Verify Installation:

- Open Command Prompt (Win + R → type cmd → press Enter). ○ Type python -version and press Enter. If installed correctly, it will display the Python version.

○ Install Additional Packages (Optional):

- To install Python packages, use pip, e.g., pip installs numpy.

2. What is IDLE in Python?

➤ **IDLE (Integrated Development and Learning Environment)** is Python's built-in development environment. It provides:

- A simple GUI for writing and executing Python code.
- A Python Shell for interactive execution.
- A script editor with syntax highlighting and debugging tools.

3. Key Features of Python?

- **Easy to Learn & Use:** Simple syntax similar to English.
- **Interpreted Language:** No need for compilation; executed line by line.
- **Cross-Platform:** Runs on Windows, macOS, and Linux.
- **Dynamically Typed:** No need to declare variable types.
- **Object-Oriented & Functional:** Supports multiple programming paradigms.
- **Large Standard Library:** Provides modules for handling files, databases, web development, etc.
- **Extensive Community Support:** Active community with many open-source packages.

4. What is Python Path?

- **PYTHONPATH** is an environment variable that tells Python where to look for modules and packages. It helps in:
 - Locating user-defined modules when importing.
 - Managing different Python versions.
 - Customizing Python's module search directories.

To check or modify PYTHONPATH:

- Use `echo %PYTHONPATH%` (Windows) or `echo $PYTHONPATH` (Mac/Linux) in the terminal.

5. Use of PEP and PIP?

- **PEP (Python Enhancement Proposal):**
 - A document that provides guidelines, best practices, and improvements for Python.
 - Example: **PEP 8** defines Python's coding style.

○ PIP (Python Package Installer):

- A tool to install and manage Python packages from PyPI (Python Package Index).
- Example: `pip install requests` install the requests module.

XI. Exercise

1. Print the Version of Python

- To check the installed Python version, open **Command Prompt (cmd)** and type:

`python --`

`version` or

`python -V`

2. Steps to Load Python Interpreter in Windows

- Follow these steps to open the Python interpreter:

1. **Open Command Prompt:** ○ Press Win + R, type `cmd`, and press Enter.

2. **Start Python Interpreter:**

- Type `python` and press Enter.
- You should see the Python interactive shell (`>>>` prompt).

3. **Verify Interpreter is Working:**

- Type `print("Hello, Python!")` and press Enter.
- If Python is installed correctly, it will print:

Hello, Python!

4. **Exit the Interpreter:**

- Type `exit()` or press Ctrl + Z and hit Enter.

Alternative:

- You can also use **IDLE** (Python's Integrated Development Environment):
 - Search for "IDLE" in the Windows Start menu and open it.

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	



DEPARTMENT OF COMPUTER ENGINEERING

Subject: PWP	Subject Code: 22616
Semester: VI	Course: Computer Engineering
Laboratory No: L001	Name of Subject Teacher: Prof. Sangeeta Wankhede
Name of Student: Darshan Bhangale	Roll Id: 22203A0039

Experiment No:	2
Title of Experiment	Write simple Python program to display message on screen

X. Practical Related Question:

1. Different Modes of Programming in Python

➤ Python supports two primary programming modes:

1. Interactive Mode:

- Allows execution of Python commands one at a time.
- Uses Python Shell (>>> prompt).
- Best for quick testing and debugging.

2. Script Mode:

- Used to write and execute Python programs stored in files (.py).
- Runs the entire script at once.
- Suitable for larger programs.

2. Procedure to Execute a Program Using Interactive Mode

➤ Follow these steps:

1. **Open Python Interpreter:**
 - Open Command Prompt (cmd).
 - Type python and press Enter.

2. Execute Python Code:

- You will see the >>> prompt. ○ Type a Python command, e.g.:

```
print("Hello, World!") ○
```

Press Enter, and it will output:

```
Hello, World!
```

3. Exit the Interactive Mode:

- Type `exit()` or press Ctrl + Z and Enter.

3. Steps Involved in Executing a Program Using Script Mode

1. Create a Python Script:

- Open a text editor (e.g., Notepad, VS Code, or IDLE). ○ Write a Python program, e.g.:

```
print("Hello, Python Script!") ○ Save
```

the file with a .py extension (e.g., script.py).

2. Execute the Script:

- Open **Command Prompt (cmd)**. ○ Navigate to the file location using `cd` command. ○ Run the script using:

```
python script.py ○ The output will be
```

displayed in the terminal.

4. Procedure to Make a Python File Executable

➤ To make a Python script executable, follow these steps:

1. Save the script as script.py.
2. Run the script in the command prompt: `python`

```
script.py
```

3. If you want to run the script directly:

o Rename script.py to script.cmd or script.bat. o Add python script.py inside the batch file. o Run it by double-clicking or from the command prompt.

XI. Exercise:

1. Python Program to Display Your Name Using Interactive Mode

➤ Follow these steps:

1. Open **Command Prompt (cmd)**.
2. Type python and press Enter to enter interactive mode (>>>).
3. Enter the following command:

```
print("Your Name")
```

4. Press **Enter**, and it will display:

```
PS D:\darshan\22203A0039> python
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Darshan")
Darshan
>>> exit()
PS D:\darshan\22203A0039> █
```

5. Type exit() to quit interactive mode.

2. Python Program to Display “MSBTE” Using Script Mode

➤ Follow these steps:

1. Open **Notepad, IDLE**, or any text editor.
2. Type the following Python script:

```
print("MSBTE")
```

3. Save the file as msbte.py.
4. Open **Command Prompt (cmd)** and navigate to the file location using the cd command.
5. Run the script using:

python msbte.py

Output:

```
PS D:\darshan\22203A0039> python msbte.py
MSBTE
PS D:\darshan\22203A0039> █
```

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Subject: Programming with python	Subject Code: 22616
Semester: 6 th Semester	Course: Computer Engineering
Laboratory No: L001B	Name of Subject Teacher: Sangeeta Wankhede
Name of Student: Darshan Bhangale	Roll Id: 22203A0039

Experiment No:	3
Title of Experiment	Write simple Python program using operators: Arithmetic Operators, Logical Operators, Bitwise Operators

- **Practical related Questions:**

1. Mention the use of //, **, % operator in Python

→

1.// (Floor Division Operator):

- This operator performs integer division, meaning it divides two numbers and returns the largest integer less than or equal to the result (i.e., the quotient is rounded down).
- Example:\

```
result = 10 // 3 # result will be 3
```

2. ** (Exponentiation Operator):

- This operator is used to raise a number to the power of another number.
- Example:

```
result = 2 ** 3 # result will be 8 (2 raised to the power of 3)
```

3. % (Modulo Operator):

- This operator returns the remainder of the division between two numbers.
- Example:

```
result = 10 % 3 # result will be 1 (the remainder when 10 is divided by 3)
```

2. Describe ternary operator in Python

→ In Python, the ternary operator is a shorthand way of writing an if-else statement. It allows you to evaluate a condition and choose between two values based on that condition, all in one line.

value_if_true if condition else value_if_false

- condition: The expression that gets evaluated (it returns either True or False).
- value_if_true: The value that will be chosen if the condition is True.
- value_if_false: The value that will be chosen if the condition is False.

Example:

```
age = 18
```

```
status = "Adult" if age >= 18 else "Minor"
```

```
print(status)
```

Output: Adult

3. Describe about different Logical operators in Python with appropriate examples.

→ In Python, logical operators are used to combine conditional statements or boolean values. There are three main logical operators:

1. and (Logical AND)
2. or (Logical OR)
3. not (Logical NOT)

1. and (Logical AND)

- The and operator returns True only if both operands are True. If either operand is False, the result will be False.

Example:

```
a = True
```

```
b = False
```

```
result = a and b
```

```
print(result)
```

Output: False (because b is False)

- If both a and b were True, the result would be True.

2. or (Logical OR)

- The or operator returns True if at least one of the operands is True. It only returns False when both operands are False.

Example:

```
a = True
```

```
b = False
```

```
result = a or b
```

```
print(result)
```

Output: True (because a is True)

- If both a and b were False, the result would be False.

3. not (Logical NOT)

- The not operator is a unary operator (works on a single operand). It inverts the boolean value of its operand:
 - If the operand is True, it returns False.
 - If the operand is False, it returns True.

Example:

```
a = True
```

```
result = not a
```

```
print(result) # Output: False (because not True is False)
```

- and: Returns True only if both operands are True.
- or: Returns True if at least one operand is True.
- not: Inverts the boolean value of the operand.

4. Describe about different Arithmetic operators in Python with appropriate examples.

→ In Python, arithmetic operators are used to perform mathematical operations between numeric values. Here are the main arithmetic operators in Python:

1. + (Addition)

- This operator adds two numbers or concatenates strings.

2. - (Subtraction)

- This operator subtracts one number from another.

3. * (Multiplication)

- This operator multiplies two numbers or repeats a string.

4. / (Division)

- This operator divides the first operand by the second operand and always returns a float, even if both operands are integers.

5. // (Floor Division)

- This operator divides the first operand by the second operand and returns the largest integer less than or equal to the result (i.e., the quotient rounded down).

6. % (Modulo)

- This operator returns the remainder when one number is divided by another.

7. ** (Exponentiation)

- This operator raises the first operand to the power of the second operand (i.e., it performs exponentiation).

Example:

```
a = 5
b = 7
result = a + b
print("Addition: ",result)

result = a - b
print("Subtraction: ",result)

result = a * b
print("Multiplication: ",result)

result = a / b
print("Division:",result)

result = a // b
print("Floor Division:",result)
```

```
result = a % b
print("Modular Division: ",result)

result = a ** b
print("Exponential: ",result)
```

OUTPUT:

```
PS D:\darshan\22203A0039> & C:/Users/bhang/AppDat
Addition: 12
Subtraction: -2
Multiplication: 35
Division: 0.7142857142857143
Floor Divsion: 0
Modular Division: 5
Exponential: 78125
PS D:\darshan\22203A0039> █
```

5. Describe about different Bitwise operators in Python with appropriate examples.

→ Bitwise operators in Python are used to manipulate data at the bit level. They operate on integers and perform bit-by-bit operations. Here's an overview of the common bitwise

- &: Bitwise AND – Returns 1 if both bits are 1.
- |: Bitwise OR – Returns 1 if at least one of the bits is 1.
- ^: Bitwise XOR – Returns 1 if the bits are different.
- ~: Bitwise NOT – Inverts all bits.
- <<: Left shift – Shifts bits to the left (multiplies by 2).
- >>: Right shift – Shifts bits to the right (divides by 2).

Example:

```
a = 3
b = 9
result = a & b
print("Bitwise AND : ",result)
result = a | b
print("Bitwise OR: ",result)
result = a ^ b
print("Bitwise XOR: ",result)
result = ~a
print("Bitwise NOT: ",result)
```

```
result = a << 1
print("left Shift: ",result)
a = 5
result = a >> 1
print("Right Shift: ",result)
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bhang.
Bitwise AND :  1
Bitwise OR:  11
Bitwise XOR:  10
Bitwise NOT:  -4
left Shift:  6
Right Shift:  2
PS D:\darshan\22203A0039> 
```

- **Exercise**

1. **Write a program to convert U.S. dollars to Indian rupees**

```
dollars = float(input("Enter The Amount:"))
rupees = dollars *87.42
print("Dollars: ",dollars)
print("Rupees: ",rupees)
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bhang/Ap
Enter The Amount:100
Dollars:  100.0
Rupees:   8742.0
PS D:\darshan\22203A0039> █
```

2. Write a program to convert bits to Megabytes, Gigabytes and Terabytes

```
3. bits = int(input("Enter the bits : "))
4. print("Bits: ", bits)
5. bytes = bits / 8
6. print("Bytes:", bytes)
7. megabytes = bytes / 10**6
8. gigabytes = bytes / 10**9
9. terabytes = bytes / 10**12
10. print("MegaBytes:", megabytes)
11. print("GigaBytes:", gigabytes)
12. print("TeraBytes:", terabytes)
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bhang/AppDat
Enter the bits : 10000000000
Bits: 10000000000
Bytes: 1250000000.0
MegaBytes: 1250.0
GigaBytes: 1.25
TeraBytes: 0.00125
PS D:\darshan\22203A0039> █
```

3. Write a program to find the square root of a number

```
num = int(input('Enter a number: '))
num_sqrt = num ** 0.5
print("The square root of ",num," is ",num_sqrt)
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bhang/AppDat
Enter a number: 25
The square root of 25 is 5.0
PS D:\darshan\22203A0039> █
```


4. Write a program to find the area of Rectangle

```
length = float(input("Enter the length of the rectangle: "))
width = float(input("Enter the width of the rectangle: "))
area = length * width
print("The area of the rectangle is: ",area)
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bhang/AppData
Enter the length of the rectangle: 10
Enter the width of the rectangle: 20
The area of the rectangle is: 200.0
PS D:\darshan\22203A0039> █
```

5. Write a program to calculate area and perimeter of the square

```
length = float(input("Enter the length: "))
area = length ** 2
perimeter = 4 * length
print("The area of the square is: ",area)
print("The perimeter of the square is: ",perimeter)
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bhang/AppData
Enter the length: 14
The area of the square is: 196.0
The perimeter of the square is: 56.0
PS D:\darshan\22203A0039> █
```

6. Write a program to calculate surface volume and area of a cylinder.

```
import math
radius = float(input("Enter the radius : "))
height = float(input("Enter the height : "))
volume = math.pi * radius**2 * height
surface_area = 2 * math.pi * radius * (radius + height)
print("The volume of the cylinder is: ",volume)
print("The surface area of the cylinder is: ",surface_area)
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bhang/AppData/Local/Program
Enter the radius : 5
Enter the height : 12.3
The volume of the cylinder is: 966.0397409788615
The surface area of the cylinder is: 543.4955290710342
PS D:\darshan\22203A0039> █
```

7. Write a program to swap the value of two variables

```
a = 15
b = 20
print(f"Before swapping: a = ",a," b = ",b)
temp = a
a = b
b = temp
print(f"After swapping: a = ",a," b = ",b)
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bhang/AppD
Before swapping: a = 15 b = 20
After swapping: a = 20 b = 15
PS D:\darshan\22203A0039> █
```

Grade and Dated Signature of Teacher	Process Related (35)	Product Related (15)	Dated Sign

Subject: Programming With Python	Subject Code:22616
Semester:6 th Semester	Course: Computer Engineering
Laboratory No: 1	Name of Subject Teacher: Sangeeta Wankhede
Name of Student: Darshan Bhangale	Roll Id: 22203A0039

Experiment No:	4
Title of Experiment	Write simple Python program to demonstrate use of conditional statements: if statement, 'if ... else' statement, Nested 'if' statement

Practical related Questions

1. List operators used in if conditional statement

→

1. Comparison Operators

- == (Equal to)
- != (Not equal to)
- > (Greater than)
- < (Less than)
- >= (Greater than or equal to)
- <= (Less than or equal to)

2. Logical Operators

- and (Logical AND)
- or (Logical OR)
- not (Logical NOT)

3. Membership Operators

- in (Checks if a value is present in a sequence)
- not in (Checks if a value is not present in a sequence)

4. Identity Operators

1. is (Checks if two variables refer to the same object)
2. is not (Checks if two variables refer to different objects)

2. Differentiate between if-else and nested-if statement

→

Feature	if-else Statement	nested-if Statement
Definition	Executes one block of code if the condition is True, otherwise executes the else block.	An if statement inside another if statement to check multiple conditions.
Structure	Uses a single if and else block.	Uses multiple if statements, one inside another.
Complexity	Simple and easy to understand.	More complex due to multiple conditions.
Use Case	Used when there are only two possible outcomes.	Used when one condition depends on another condition.
Example	<pre>python if x > 10: print("Greater") else: print("Smaller or Equal")</pre>	<pre>python if x > 10: if x % 2 == 0: print("Even & Greater than 10") else: print("Odd & Greater than 10")</pre>

Exercise

1. Write a program to check whether a number is even or odd

→

Code

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

if num % 2 == 0:
    print(num, " is an Even number.")
else:
    print(num, " is an Odd number.")
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bhang/AppDa
Enter a number: 14
14 is an Even number.
PS D:\darshan\22203A0039> █
```

2. Write a program to find out absolute value of an input number

→

Code :

```
num = float(input("Enter a number: "))
if num < 0:
    result = -num
else:
    result = num

print("The absolute value of", num, "is", result)
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bhang/AppDa
Enter a number: 13
The absolute value of 13.0 is 13.0
PS D:\darshan\22203A0039> █
```

3. Write a program to check the largest number among the three numbers

→

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

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

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

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bhang/App
Enter first number: 12
Enter second number: 11
Enter third number: 18
The largest number is: 18.0
PS D:\darshan\22203A0039> █
```

4. Write a program to check if the input year is a leap year or not

→

```
year = int(input("Enter a year: "))

if year % 4 == 0:
    print(year," is a Leap Year")
else:
    print(year," is Not a Leap Year")
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bha
Enter a year: 2016
2016 is a Leap Year
PS D:\darshan\22203A0039> 
```

5. Write a program to check if a Number is Positive, Negative or Zero

→

Code

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

if num >= 0:
    print(num, " is a Positive Number")
elif num < 0:
    print(num, " is a Negative Number")
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bha
Enter a number: 39
39.0 is a Positive Number
PS D:\darshan\22203A0039> 
```


6. Write a program that takes the marks of 5 subjects and displays the grade.

→

Code

Program to calculate grade based on marks of 5 subjects

```
marks = []
for i in range(5):
    mark = float(input("Enter marks for subjects : "))
    marks.append(mark)

total_marks = sum(marks)
percentage = (total_marks / 500) * 100

if percentage >= 90:
    grade = 'A'
elif percentage >= 80:
    grade = 'B'
elif percentage >= 70:
    grade = 'C'
elif percentage >= 60:
    grade = 'D'
else:
    grade = 'F'

print("Total Marks: ",total_marks)
print("Percentage: ",percentage,"%")
print("Grade: ",grade)
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bhang/A
Enter marks for subjects : 78
Enter marks for subjects : 92
Enter marks for subjects : 89
Enter marks for subjects : 96
Enter marks for subjects : 84
Total Marks: 439.0
Percentage: 87.8 %
Grade: B
PS D:\darshan\22203A0039> █
```

Grade and Dated Signature of Teacher	Process Related (35)	Product Related (15)	Dated Sign

Subject: Programming with python	Subject Code: 22616
Semester: 6 th Semester	Course: Computer Engineering
Laboratory No: L001B	Name of Subject Teacher: Prof. Sangeeta Wankhede
Name of Student: Darshan Bhangale	Roll Id: 22203A0039

Experiment No:	5
Title of Experiment	Write Python program to demonstrate use of looping statements: a) 'while' loop b) 'for' loop c) Nested loops

- **Practical related Questions:**

1. What would be the output from the following Python code segment?

```
x = 10 while
```

```
x > 5:
```

```
    print(x)
```

```
    x -= 1
```

Output:

10

9

8

7

6

2. Change the following Python code from using a while loop to for loop:

→ while loop :

```
x=1
```

```
while x<10:
```

```
    print x,
```

```
    x+=1
```

for loop:

```
for x in range(1, 10):
```

```
    print(x, end=" ")
```

- **Exercise**

1. Print the following patterns using loop:

a.

```
*
```

```
**
```

```
***
```

```
****
```

Code:

```
for i in range(1, 5):  
    print("*" * i)
```

```
PS D:\darshan\22203A0039> & C:/Users/bha
*
**
***
****
PS D:\darshan\22203A0039> 
```

b.

b.

*

*

Code:

```
for i in range(1, 6, 2):
    print('*' * i)

for i in range(3, 0, -2):
    print('*' * i)
```

```
PS D:\darshan\22203A0039> & C:/Use
*
***
*****
***
*
PS D:\darshan\22203A0039> 
```

c.

1010101

10101

101

1

Code:

```
for i in range(7, 0, -2):
    pattern = ""
    for j in range(i):
        if j % 2 == 0:
            pattern += "1"
        else:
            pattern += "0"
    print(pattern)
```

```
PS D:\darshan\22203A0039> & C:/Users/bhang/AppData/Local/Programs/Python/Python312/python.exe d:/darshan/22203A0039/msbte.py
1010101
10101
101
1
PS D:\darshan\22203A0039> █
```

2. Write a Python program to print all even numbers between 1 to 100 using while loop.

Code:

```
num = 2
while num <= 100:
    print(num, end=" ")
    num += 2
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bhang/AppData/Local/Programs/Python/Python312/python.exe d:/darshan/22203A0039/msbte.py
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100
PS D:\darshan\22203A0039> █
```

3. Write a Python program to find the sum of first 10 natural numbers using for loop.

Code:

```
4.sum = 0
5.for num in range(1, 11):
6.     sum += num
7.print("Sum of first 10 natural numbers:", sum)
8.
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bha
Sum of first 10 natural numbers: 55
PS D:\darshan\22203A0039> █
```

4. Write a Python program to print Fibonacci series.

Code:

```
n = int(input("Enter the number of terms: "))
a, b = 0, 1
for _ in range(n):
    print(a, end=" ")
    a, b = b, a + b
```

Output: Enter the number of terms: 10

0 1 1 2 3 5 8 13 21 34

5. Write a Python program to calculate factorial of a number

Code:

```
num = int(input("Enter a number: "))
factorial = 1
for i in range(1, num + 1):
    factorial *= i
print("Factorial of", num, "is", factorial)
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users/bhang
Enter a number: 5
Factorial of 5 is 120
PS D:\darshan\22203A0039> & C:/Users/bhang
█
```

6. Write a Python Program to Reverse a Given Number

Code:

```
num = int(input("Enter a number: "))
rev = 0
while num > 0:
    d = num % 10
    rev = rev * 10 + d
    num //= 10
print("Reversed number:", rev)
```

Output:

```
PS D:\darshan\22203A0039> & C:/User
Enter a number: 12
Reversed number: 21
PS D:\darshan\22203A0039> █
```

7. Write a Python program takes in a number and finds the sum of digits in a number.

Code:

```
num = int(input("Enter a number: "))
sum = 0
while num > 0:
    d = num % 10
    sum += d
    num //= 10
print("Sum of digits:", sum)
```

Output:

```
PS D:\darshan\22203A0039> & C:/Us
Enter a number: 92332
Sum of digits: 19
PS D:\darshan\22203A0039> █
```


8. Write a Python program that takes a number and checks whether it is a palindrome or not.

Code:

```
num = int(input("Enter a number: "))
temp = num
rev = 0
while temp > 0:
    d = temp % 10
    rev = rev * 10 + d
    temp //= 10
if num == rev:
    print(num, "is a palindrome")
else:
    print(num, "is not a palindrome")
```

Output:

```
PS D:\darshan\22203A0039> & C:/Users
Enter a number: 232
232 is a palindrome
PS D:\darshan\22203A0039> █
```

Grade and Dated Signature of Teacher	Process Related (35)	Product Related (15)	Dated Sign

Subject: Programming with python	Subject Code: 22616
Semester: 06	Course: CO6I-A
Laboratory No: L001-B	Name of Subject Teacher: Prof. Sangeeta Shirsat
Name of Student: Darshan Bhangale	Roll ID: 22203A0039

Experiment No:	6
Title of Experiment	Write Python program to perform following operations on Lists: Create list, Access list, Update list (Add item, Remove item), Delete list

- **Practical related questions**

1. When to use list

Solution:

A list in Python is used when you need to store multiple values in a single variable while keeping them in order. Lists allow you to add, remove, or change elements as needed.

They are useful when you need to access elements using their position or when you want to store different types of data together, like numbers and strings.

Lists can grow or shrink in size, making them flexible for many situations. If you need an ordered, changeable collection of items, a list is a good choice.

2. Describe various list functions

Solution:

Function	Syntax	Description
<code>append()</code>	<code>list.append(item)</code>	Adds an item to the end of the list.
<code>extend()</code>	<code>list.extend(iterable)</code>	Adds multiple elements from another list or iterable.
<code>insert()</code>	<code>list.insert(index, item)</code>	Inserts an item at a specific position.
<code>remove()</code>	<code>list.remove(item)</code>	Removes the first occurrence of a specific item.
<code>pop()</code>	<code>list.pop(index)</code>	Removes and returns the item at the given index (default is last).
<code>clear()</code>	<code>list.clear()</code>	Removes all elements from the list.
<code>index()</code>	<code>list.index(item)</code>	Returns the index of the first occurrence of an item.
<code>count()</code>	<code>list.count(item)</code>	Returns the number of times an item appears in the list.
<code>sort()</code>	<code>list.sort()</code>	Sorts the list in ascending order (modifies the list).
<code>reverse()</code>	<code>list.reverse()</code>	Reverses the order of elements in the list.
<code>copy()</code>	<code>list.copy()</code>	Returns a copy of the list.

3. Write syntax for a method to sort a list

Solution:

The syntax to sort a list in Python is:

`list.sort()`

4. Write syntax for a method to count occurrences of a list item in Python

Solution:

The syntax to count occurrences of an item in a list is:
list.count()

5. How to concatenate list

Solution:

You can concatenate (combine) two or more lists in Python using the following methods:

1. Using + Operator

```
list1 = [1, 2, 3]
list2 = [4, 5, 6]
result = list1 + list2
print(result)
```

2. Using extend

```
list1 = [1, 2, 3]
list2 = [4, 5, 6]
list1.extend(list2)
print(list1)
```

6. Justify the statement “Lists are mutable”

Solution:

The statement "Lists are mutable" means that lists in Python can be modified after creation. You can change, add, or remove elements without creating a new list.

```
numbers = [1, 2, 3]
numbers[1] = 5
numbers.append(4)
numbers.remove(1)

print(numbers)
```

7. Describe the use pop operator in list

Solution:

The pop() method in Python is used to remove and return an element from a list based on its index. If no index is specified, it removes and returns the last element

- **Excercise**

1. Write a Python program to sum all the items in a list

Solution:

```
list1 = [10, 20, 30, 40, 50]
total = sum(list1)
print("The List is: ",list1)
print("The sum of all the items in the list is:", total)
```

```
PS C:\Users\student1\Desktop\darshan> & C:/Users/s
first.py
The List is: [10, 20, 30, 40, 50]
The sum of all the items in the list is: 150
PS C:\Users\student1\Desktop\darshan> □
```

2. Write a Python program to multiplies all the items in a list

```
list1 = [5, 2, 2, 2, 2]
result = 1
print("The list is: ",list1)
for n in list1:
    result = result * n
print("The product of all the items in the list is:", result)
```

```
PS C:\Users\student1\Desktop\darshan> & C:/Users/stud
second.py
The list is: [5, 2, 2, 2, 2]
The product of all the items in the list is: 80
PS C:\Users\student1\Desktop\darshan> █
```

3. Write a Python program to get the largest number from a list

```
list1 = [1, 30, 512, 72, 211,32,1900]
num = max(list1)
print("The List is: ", list1)
print("The largest number in the list is:", num)
```

```
PS C:\Users\student1\Desktop\darshan> & C:/Users
third.py
The List is: [1, 30, 512, 72, 211, 32, 1900]
The largest number in the list is: 1900
PS C:\Users\student1\Desktop\darshan> █
```

4. **Write a Python program to get the smallest number from a list.**

```
list1 = [1, 30, 512, 72, 211, 32, 1900]
num = min(list1)
print("The List is: ", list1)
print("The Smallest number in the list is:", num)
```

```
PS C:\Users\student1\Desktop\darshan> & C:/Users
fourth.py
The List is:  [1, 30, 512, 72, 211, 32, 1900]
The Smallest number in the list is: 1
PS C:\Users\student1\Desktop\darshan> █
```

5. **Write a Python program to reverse a list**

```
list1 = [10, 32, 12, 50, 20]
print("The List is: ", list1)
list1.reverse()
print("Reversed List: ", list1)
```

```
PS C:\Users\student1\Desktop\darshan> & C:/Use
fifth.py
The List is:  [10, 32, 12, 50, 20]
Reversed List:  [20, 50, 12, 32, 10]
PS C:\Users\student1\Desktop\darshan> █
```

6. Write a Python program to find common items from two lists

```
list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]
list3 = list(set(list1) & set(list2))
print("List1 is: ",list1)
print("List2 is: ",list2)
print("The common items between the two lists are:", list3)
```

```
PS C:\Users\student1\Desktop\darshan> & C:/Users/student1/
sixth.py
List1 is:  [1, 2, 3, 4, 5]
List2 is:  [4, 5, 6, 7, 8]
The common items between the two lists are: [4, 5]
PS C:\Users\student1\Desktop\darshan> █
```

7. Write a Python program to select the even items of a list

```
list1 = [2,3,5,6,10]
list2 = []
for n in list1:
    if n % 2 == 0:
        list2.append(n)
print("The List is: ",list1)
print("The Even Numbers in the list are: ",list2)
```

```
PS C:\Users\student1\Desktop\darshan> & C:/Users/stu
seventh.py
The List is:  [2, 3, 5, 6, 10]
The Even Numbers in the list are:  [2, 6, 10]
PS C:\Users\student1\Desktop\darshan> █
```

Grade and Dated Signature of Teacher	Process Related (35)	Product Related (15)	Dated Sign

Subject: Programming with python	Subject Code: 22616
Semester: 6 th Semester	Course: Computer Engineering
Laboratory No: L001B	Name of Subject Teacher: Prof. Sangeeta Wankhede
Name of Student: Darshan Bhangale	Roll Id: 22203A0039

Experiment No:	7
Title of Experiment	Write Python program to perform following operations on Tuples: Create Tuple, Access Tuple, Update Tuple, Delete Tuple

- **Practical related Questions:**

1. Define empty tuple. Write syntax to create empty tuple. $x = 10$

Ans: An empty tuple is a tuple that contains no elements.

Syntax: empty_tuple = ()

Code: empty_tuple = ()

```
print("Empty Tuple:", empty_tuple)
```

```
print("Type:", type(empty_tuple))
```

Output: Empty Tuple: ()

Type: <class 'tuple'>

2. Write syntax to copy specific elements existing tuple into new tuple.

Ans: Syntax to Copy Specific Elements from an Existing Tuple into a New Tuple:

```
old_tuple = (1, 2, 3, 4, 5)
```

```
new_tuple = (old_tuple[1], old_tuple[3])
```

```
print("New Tuple:", new_tuple)
```

Output: New Tuple: (2, 4)

3. Compare tuple with list

Ans:

Tuple	List
Tuples are immutable	Lists are mutable
The implication of iterations is comparatively faster	The implication of iterations is time-consuming
A tuple data type is appropriate for accessing the elements	The list is better for performing operations, such as insertion and deletion
Tuple consumes less memory as compared to the list	Lists consume more memory
Tuple does not have many built-in methods	Lists have several built-in methods
In a tuple, it is hard to take place	Unexpected changes and errors are more likely to occur

• Exercise

1. Create a tuple and find the minimum and maximum number from it.

Code:

```
tuple1 = (3, 15, 7, 1, 9, 24, 45)
```

```
min = min(tuple1)
```

```
max = max(tuple1)
```

```
print("The Tuple is: ", tuple1)
```

```
print("Minimum number:", min)
```

```
print("Maximum number:", max)
```

Output:

```
PS C:\Users\student1\Desktop\darshan> & C:/Users/student1/A
eight.py
The Tuple is: (3, 15, 7, 1, 9, 24, 45)
Minimum number: 1
Maximum number: 45
PS C:\Users\student1\Desktop\darshan> █
```

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

Code:

```
tuple1 = (1, 2, 3, 4, 5)
tuple2 = (4, 5, 6, 7, 3)
tuple3 = tuple(set(tuple1) & set(tuple2))
print("Tuple1 is: ",tuple1)
print("Tuple2 is: ",tuple2)
print("The common items between the two Tuples are:", tuple3)
```

Output:

```
PS C:\Users\student1\Desktop\darshan> & C:/Users/student1/A
ninth.py
Tuple1 is: (1, 2, 3, 4, 5)
Tuple2 is: (4, 5, 6, 7, 3)
The common items between the two Tuples are: (3, 4, 5)
PS C:\Users\student1\Desktop\darshan> █
```

Print the number in words for Example: 1234 => One Two Three Four Code:

```
digits=("zero","one","two","three","four","five","six","seven","eight","nine")

num=5312

print("Number is=",num)

for x in str(num):

    print(digits[int(x)],end=" ")
```

Output:

```
PS D:\darshan\22203A0039> & C:/
Number is= 5312
five three one two
PS D:\darshan\22203A0039> █
```

Grade and Dated Signature of Teacher	Process Related (35)	Product Related (15)	Dated Sign

Subject: Python	Subject Code: 22412
Semester: 06	Course: CO6IA
Laboratory No: L001B	Name of Subject Teacher: Sangeeta Shirsat
Name of Student: Darshan Bhangale	Roll Id: 22203A0039

Experiment No:	08
Title of Experiment	Write Python program to perform following operations on Set: Create Set, Access Set elements, Update Set, Delete Set

XI. Exercise

- 1. Write a Python program to create a set, add member(s) in a set and remove one item from set.**

Code :

```
my_set = {1,2,3,4,5}
print("Orginial set : ",my_set)

my_set.add(6)
print("Set after adding element : ",my_set)
```

Output:

```
Orginial set : {1, 2, 3, 4, 5}
Set after adding element : {1, 2, 3, 4, 5, 6}
```

- 2. Write a Python program to perform following operations on set: intersection of sets, union of sets, set difference, symmetric difference, clear a set.**

Code :

```
my_set = {1,2,3,4,5}
my_set1 = {3,4,5,6,7,8}
print("Orginial set : ",my_set)
print("Orginial set2 : ",my_set1)
```

```

print("")
my_set.intersection(my_set1)
print("Set after intersection : ",my_set)

my_set.union(my_set1)
print("Set after union : ",my_set)

my_set.difference(my_set1)
print("Set after difference : ",my_set)

my_set.symmetric_difference(my_set1)
print("Set after symmetric_difference : ",my_set)

```

Output:

```

Orginial set : {1, 2, 3, 4, 5}
Orginial set2 : {3, 4, 5, 6, 7, 8}

Set after intersection : {1, 2, 3, 4, 5}
Set after union : {1, 2, 3, 4, 5}
Set after difference : {1, 2, 3, 4, 5}
Set after symmetric_difference : {1, 2, 3, 4, 5}

```

3. Write a Python program to find maximum and the minivalue in a set.

Code :

```

my_set = {1,2,3,4,5}
my_set1 = {3,4,5,6,7,8}
print("Orginial set : ",my_set)
print("Orginial set2 : ",my_set1)
print("")
print("Minimum value of set1 : ",min(my_set))
print("Maximum value of set1 : ",max(my_set))
print("")
print("Minimum value of set2 : ",min(my_set1))
print("Maximum value of set2 : ",max(my_set1))

```

Output:

```
Orginial set : {1, 2, 3, 4, 5}
Orginial set2 : {3, 4, 5, 6, 7, 8}

Minimum value of set1 : 1
Maximum value of set1 : 5

Minimum value of set2 : 3
Maximum value of set2 : 8
```

4. Write a Python program to find the length of a set.

Code :

```
my_set = {1,2,3,4,5}
my_set1 = {3,4,5,6,7,8}
print("Orginial set : ",my_set)
print("Orginial set2 : ",my_set1)
print("")
print("Length of set1 : ",len(my_set))
print("Length of set2 : ",len(my_set1))
```

Output:

```
Orginial set : {1, 2, 3, 4, 5}
Orginial set2 : {3, 4, 5, 6, 7, 8}

Length of set1 : 5
Length of set2 : 6
```

X. Practical related Questions

1. Describe the various set operations

1. Union:

- Union operation performed on two sets returns all the elements from both the sets.
- The union of A and B is defined as the set that consists of all elements belonging to either set A or set B (or both).It is performed by using | operator.

Example: For union operation in sets.

```
>>> A={1,2,4,6,8}
```

```
>>> B={1,2,3,4,5}
```

```
>>> C=A | B
```

```
>>> C
```

```
{1, 2, 3, 4, 5, 6, 8}
```

2. Intersection:

- Intersection operation performed on two sets returns all the elements which are common or in both the sets.
- The intersection of A and B is defined as the set composed of all elements that belong to both A and B.
- It is performed by using & operator.

Example: For intersection operation in sets.

```
>>> A={1,2,4,6,8}
```

```
>>> B={1,2,3,4,5}
```

```
>>> C=A & B
```

```
>>> C
```

```
{1, 2, 4}
```

3. Difference:

- Difference operation on two sets set1 and set2 returns all the elements which are present on set1 but not in set2.
- It is performed by using – operator.

Example: For difference operation in sets.

```
>>> A={1,2,4,6,8}
```

```
>>> B={1,2,3,4,5}
```

```
>>> C=A -B
```

4. Symmetric Difference:

- Symmetric Difference operation performed by using ^ operation.

Example: For symmetric difference operation in sets.

```
>>> A={1,2,4,6,8}
```

```
>>> B={1,2,3,4,5}
```

```
>>> C=A ^ B
```

```
>>> C
```

```
{3, 5, 6, 8}
```

2. Describe the various methods of set

The sets in Python are typically used for mathematical operations like union, intersection, difference and complement etc.

a) Creating a set:

A set is created by using the set() function or placing all the elements within a pair of curly braces.

Example:

```
>>> a={1,3,5,4,2}
```

```
>>> print("a=",a)
```

```
a= {1, 2, 3, 4, 5}
```

```
>>> print(type(a))
```

```
<class 'set'>
```

b) Accessing values in a set: We cannot access individual values in a set. We can only access all the elements together. But we can also get a list of individual elements by looping through the set.

Example:

```
Num=set([10,20,30,40,50])
```

```
for n in Num:
```

```
print(n)
```

Output:

```
10
```

```
20
```

```
30
```

```
40
```

```
50
```

c) Updating items in a set: We can add elements to a set by using add() method. There is no specific index attached to the newly added element.

Example:

```
Num=set([10,20,30,40,50])
```

```
Num.add(60)
print(Num)
Output:
{ 10,20,30,40,50,60}
```

d) Removing items in set: We can remove elements from a set by using discard() method. There is no specific index attached to the newly added element.

Example:

```
Num=set([10,20,30,40,50])
Num.discard(50)
Print(Num)
Output:
{ 10,20,30,40}
```

Subject: Python	Subject Code: 22412
Semester: 06	Course: CO6IA
Laboratory No: L001B	Name of Subject Teacher: Sangeeta Shirsat
Name of Student: Darshan Bhangale	Roll Id: 2220A0039

Experiment No:	09
Title of Experiment	Write Python program to perform following operations on Dictionaries: Create Dictionary, Access Dictionary elements, Update Dictionary, Delete Dictionary, Looping through Dictionary

XI. Exercise

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

Code :

```
my_set = {'b':1,'a':2,'e':3,'d':4,'c':5}

print("Original : ",str(my_set))

print("")

sorted = dict(sorted(list(my_set.items())))

print("Sorted by value in ascending order : ",sorted)

rev = dict(reversed(list(sorted.items())))

print("Sorted by value in descending order : ",rev)
```

Output:

```
Original : {'b': 1, 'a': 2, 'e': 3, 'd': 4, 'c': 5}
Sorted by value in ascending order : {'a': 2, 'b': 1, 'c': 5, 'd': 4, 'e': 3}
Sorted by value in descending order : {'e': 3, 'd': 4, 'c': 5, 'b': 1, 'a': 2}
```

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

a. Sample Dictionary:

b. dic1 = {1:10, 2:20}

c. dic2 = {3:30, 4:40}

d. dic3 = {5:50,6:60}

Code :

```
dict1 = {1: 10, 2: 20}
dict2 = {3: 30, 4: 40}
dict3 = {5: 50, 6: 60}
concatenated_dict = {}
concatenated_dict.update(dict1)
concatenated_dict.update(dict2)
concatenated_dict.update(dict3)
print(concatenated_dict)
```

Output:

```
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

3. Write a Python program to combine two dictionary adding values for common keys.

a. d1 = {'a': 100, 'b': 200, 'c':300}

b. d2 = {'a': 300, 'b': 200, 'd':400}

Code :

```
d1 = {'a': 100, 'b': 200, 'c': 300}
d2 = {'a': 300, 'b': 200, 'd': 400}
combined_dict = {}
```

```
for key in set(d1) | set(d2): # Union of keys from both dictionaries
    combined_dict[key] = d1.get(key, 0) + d2.get(key, 0) # Add values if key exists
print(combined_dict)
```

Output:

```
{'b': 400, 'c': 300, 'd': 400, 'a': 400}
```

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

a. Sample Data: [{"V": "S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII": "S005"}, {"V": "S009"}, {"VIII": "S007"}]

Code :

```
sample_data = [{"V": "S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII": "S005"}, {"V": "S009"}, {"VIII": "S007"}]

unique_values = set()

for dic in sample_data:
    for value in dic.values():
        unique_values.add(value)

print(unique_values)
```

Output:

```
{'S001', 'S005', 'S007', 'S002', 'S009'}
```

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

Code :

```
my_dict = eval(input("Enter the dictionary: "))

highest_values = []

highest_keys = []

for key, value in my_dict.items():
    if not highest_values or value > highest_values[-1]:
        highest_values.append(value)
```

```
highest_keys.append(key)

if len(highest_values) > 3:

    highest_values.pop(0)

    highest_keys.pop(0)

print("Three highest values in the dictionary:")

for i in range(len(highest_keys)):

    print(highest_keys[i], ":" ,highest_values[i])
```

Output:

```
Enter the dictionary: dict(Germany="Berlin", Canada="Ottawa", England="London")
Three highest values in the dictionary:
Germany : Berlin
Canada : Ottawa
```

Subject: Programming with python	Subject Code:22516
Semester: 6 th Semester	Course: Computer Engineering
Laboratory No: L001B	Name of Subject Teacher: Prof. Sangeeta Shirsat
Name of Student: Darshan Bhangale	Roll Id: 22203A0039

Experiment No:	10
Title of Experiment	Write Python program to demonstrate math built-in functions (Any 2 programs)

• **Practical related questions :**

1. Describe about string formatting operator with example Solution:

Symbol	Description
%s	Inserts a string
%d	Inserts an integer
%f	Inserts a floating-point number
%x	Inserts a lowercase hexadecimal
%X	Inserts an uppercase hexadecimal

Example:

```
name = "Darshan"
age = 18
height = 5.9
number = 900
```

```
print("Name is: %s" % name)
print("Age is: %d" % age)
print("Height is: %f" % height)
```

```
print("Hex (lowercase): %x" % number)
print("Hex (uppercase): %X" % number)
```

Output:

```
Output
Name is: Darshan
Age is: 18
Height is: 5.900000
Hex (lowercase): 384
Hex (uppercase): 384

=== Code Execution Successful ===
```

2. Give the syntax and example of title() and capitalize() methods

Solution:

Syntax of title() and capitalize()

Method	Syntax
capitalize()	string.capitalize()
title()	string.title()

Example:

```
name = "darshan bhangale"
capital_name = name.capitalize()
print("capitalize name:", capital_name)
title_name = name.title()
print("titlecase name:", title_name)
```


Output:

```
Output
capitalize name: Darshan bhangale
titlecase name: Darshan Bhangale

=== Code Execution Successful ===
```

3. Give the syntax and significance of string functions: title() and strip().

Solution:

Function	Syntax	Significance
title()	string.title()	Capitalizes the first letter of each word in the string while converting the rest to lowercase.
strip()	string.strip()	Removes leading and trailing whitespace (or specified characters) from the string.

Example:

```
name = "Darshan bhangale"
first_name = " Darshan"
print("using title():", name.title())
print("using strip():", first_name.strip())
```

Output:

```
Output
using title(): Darshan Bhangale
using strip(): Darshan

=== Code Execution Successful ===
```

- **Exercise :**

1. **Write a Python function that accepts a string and calculate the number of upper case letters and lower case letters**

Solution:

```
str = "Darshan Bhangale"
u_count = 0
l_count = 0
for i in str:
    if i.isupper():
        u_count += 1
    elif i.islower():
        l_count += 1
print("The String is: ",str)
print("Uppercase Letters:", u_count)
print("Lowercase Letters:", l_count)
```

Output:

```
Output
The String is:  Darshan Bhangale
Uppercase Letters: 2
Lowercase Letters: 13

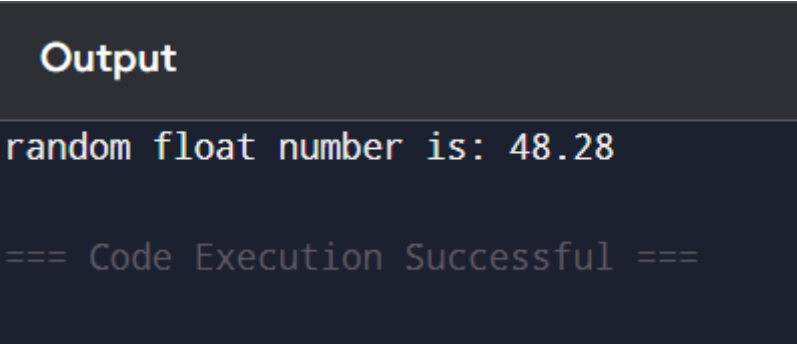
=== Code Execution Successful ===
```

2. Write a Python program to generate a random float where the value is between 5 and 50 using Python math module

Solution:

```
import math
import random
random_float_number = random.uniform(5, 50)
res = round(random_float_number, 2)
print("random float number is:", res)
```

Output:



```
Output
random float number is: 48.28
=== Code Execution Successful ===
```

Subject: Programming with python	Subject Code:22516
Semester: 6 th Semester	Course: Computer Engineering
Laboratory No: L001B	Name of Subject Teacher: Prof. Sangeeta Shirsat
Name of Student: Darshan Bhangale	Roll Id: 22203A0039

Experiment No:	11
Title of Experiment	Develop user defined Python function for given problem: a. Function with minimum 2 arguments b. Function returning value

• **Practical related questions :**

1. What is the output of the following program?

```
def myfunc(text, num):  
    while num > 0:  
        print(text)  
        num = num - 1  
    myfunc('Hello', 4)
```

Solution:

Output

```
Hello  
Hello  
Hello  
Hello
```

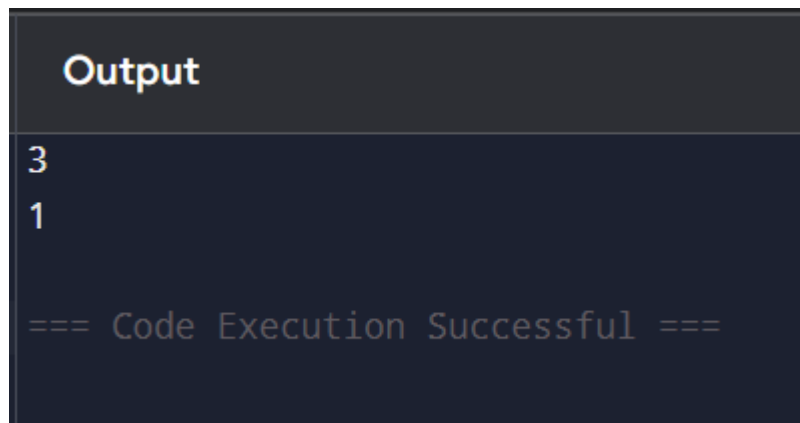
```
=== Code Execution Successful ===
```

2. What is the output of the following program?

```
num = 1
def func():
    num = 3

    print(num)
func()
print(num)
```

Solution:

A screenshot of a code execution environment with a dark background. At the top, the word "Output" is written in a light blue font. Below it, the numbers "3" and "1" are printed on separate lines in a light blue font. At the bottom, the text "=== Code Execution Successful ===" is displayed in a light blue font.

```
Output
3
1

=== Code Execution Successful ===
```

• Exercise :

1. Write a Python function that takes a number as a parameter and check the number is prime or not

Solution:

```
def prime(no):
    f = 0
    stop = no
    for i in range(2, stop):
        if no % i == 0:
            f = 1
            break

    if f == 0:
        print(f"{no} is a Prime number")
    else:
        print(f"{no} is not a Prime number")

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

Output:

```
Output
Enter a number: 39
39 is not a Prime number

=== Code Execution Successful ===
```

2. Write a Python function to calculate the factorial of a number (a non-negative integer). The function accepts the number as an argument

Solution:

```
def find_factorial(num):
    if num < 0:
        return "Factorial is not defined for negative numbers"

    result = 1
    for val in range(1, num + 1):
        result *= val

    return result

n = int(input("Enter a number: "))
print(f"Factorial of {n} is:", find_factorial(n))
```

Output:

```
Output
Enter a number: 5
Factorial of 5 is: 120

=== Code Execution Successful ===
```

3. Write a Python function that accepts a string and calculate the number of upper case letters and lower case letters

Solution:

```
str = "Darshan Bhangale"
u_count = 0
l_count = 0

for i in str:
    if i.isupper():
        u_count += 1
    elif i.islower():
        l_count += 1
print("The String is; ",str)
print("Uppercase Letters:", u_count)
print("Lowercase Letters:", l_count)
```

Output:

Output

```
The String is;  Darshan Bhangale
Uppercase Letters: 2
Lowercase Letters: 13
```

```
=== Code Execution Successful ===
```



DEPARTMENT OF COMPUTER ENGINEERING

Subject: Programming With Python	Subject Code:22616
Semester: 6 th Semester	Course: Computer Engineering
Laboratory No: L001B	Name of Subject Teacher: Prof. Sangeeta Wankhede
Name of Student: Darshan Bhangale	Roll Id: 22203A0039

Experiment No:	12
Title of Experiment	<ul style="list-style-type: none">• Write Python program to demonstrate math built-in functions (Any 2 programs)• Write Python program to demonstrate string built-in functions (Any 2 programs)

IX. Practical related Questions

1. Describe about string formatting operator with example.

Ans:

The string formatting operator (%) is used to format strings by inserting values into placeholders(%s,%d,etc).

Syntax: "string % format_values"

Example:

name="Darshan"

age=18

print("My name is %s and I am %d years old"%(name,age))

Output:

Output

My name is Darshan and I am 18 years old

=== Code Execution Successful ===

2. Give the syntax and example of title() and capitalize() methods.

Ans:

Syntax: string.title()
 string.capitalize()

CODE:

```
text = "vidyalankar polytechnic"  
print(text.title())  
print(text.capitalize())
```

OUTPUT:

Output

```
Vidyalankar Polytechnic  
Vidyalankar polytechnic
```

```
=== Code Execution Successful ===
```

**3. Give the syntax and significance of string functions:
title() and strip().**

Ans:

Syntax: string.title()

string.strip()

Code:

```
text1 = "VIT"  
print(text1.strip())
```

OUTPUT:

Output

VIT

=== Code Execution Successful ===

X. Exercise

1. . Write a Python function that accepts a string and calculate the number of upper case letters and lower case letters.

Code:

```
def count_case(str):
    upper_count = 0
    lower_count = 0

    for char in str:
        if char.isupper():
            upper_count += 1
        elif char.islower():
            lower_count += 1
    print("The String is:", str)
    print("Uppercase letters:", upper_count)
    print("Lowercase letters:", lower_count)

count_case("Darshan Bhangale")
```

OUTPUT:

Output

```
The String is: Darshan Bhangale
Uppercase letters: 2
Lowercase letters: 13
```

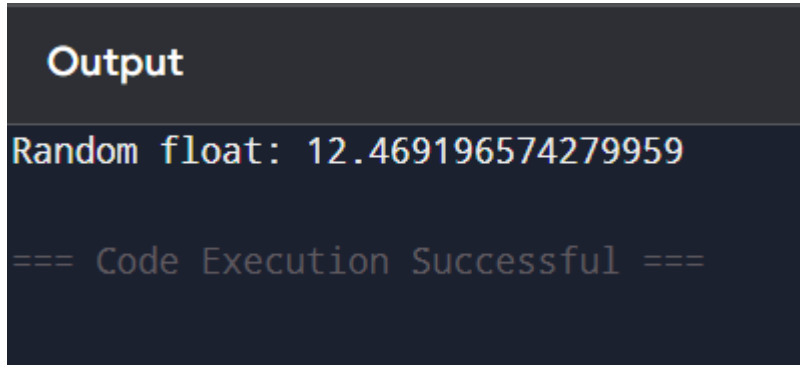
```
=== Code Execution Successful ===
```

2. Write a Python program to generate a random float where the value is between 5 and 50 using Python math module.

CODE:

```
import random
random_float = random.uniform(5, 50)
print("Random float:", random_float)
```

OUTPUT:



Output

Random float: 12.469196574279959

=== Code Execution Successful ===

Subject: PWP	Subject Code: 22616
Semester: 6 th Semester	Course: Computer Engineering
Laboratory No: L001B	Name of Subject Teacher: Prof. Sangeeta Shirsat
Name of Student: Darshan Bhangale	Roll Id: 22203A0039

Experiment No:	13
Title of Experiment	Write Python program to demonstrate use of: 1. built-in packages (e.g. NumPy, Pandas) 2. user defined packages

• **Practical related questions :**

1. Describe numpy array :

Ans :

A NumPy array is a powerful data structure provided by the NumPy library in Python used for storing and handling large, multi-dimensional numerical data efficiently.

2. Why should we use Numpy rather than Matlab, Octave or Yorick?

Ans :

- **Open-Source & Free**

1. NumPy is completely free and open-source, whereas MATLAB requires a paid license.
2. Octave and Yorick are also free, but they don't have the same ecosystem support as NumPy.

- **Integration with Python Ecosystem**

1. NumPy works seamlessly with Python, which is widely used in data science, AI, ML, web development, and automation.
2. You can integrate NumPy with Pandas, TensorFlow, PyTorch, SciPy, Matplotlib, etc., making it a better choice for real-world applications.

- **Exercise :**

1. **Write a python program to create two matrices and perform addition, subtraction, multiplication and division on matrix.**

Ans :

Code :

Import numpy as np

```
A = np.array([[4, 8],  
              [6, 2]])
```

```
B = np.array([[2, 4],  
              [3, 1]])
```

```
Add = A + B
```

```
Print("Addition:\n",  
add)
```

```
Sub = A - B
```

```
Print("Subtraction:\n",  
sub)
```

```
Mul = A * B
```

```
Print("Multiplication:\n", mul)
```

```
Div = A / B
```

```
Print("Division:\n",  
div)
```

Output :

```
Addition:  
[[ 6 12]  
[ 9  3]]  
Subtraction:  
[[2 4]  
[3 1]]  
Multiplication:  
[[ 8 32]  
[18  2]]  
Division:  
[[2. 2.]  
[2. 2.]]
```

2. Write a python program to concatenate two strings.

Ans :

Code :

```
import numpy as np

str1 = np.array(['Darshan'])
str2 = np.array(['Bhangale'])

result = np.char.add(str1, ' ')
result = np.char.add(result, str2)

print(result)
```

Output

```
['Darshan Bhangale']
```

```
=== Code Execution Successful ===
```


3. Write a numpy program to generate six random integers between 10 and 30.

Ans :

Code :

```
import numpy as np
```

```
random_numbers = np.random.randint(10, 31, size=6)  
print(random_numbers)
```

Output :

Output

```
[18 30 21 26 24 14]
```

```
=== Code Execution Successful ===
```



DEPARTMENT OF COMPUTER ENGINEERING

Subject: Programming with Python	Subject Code: 22616
Semester: 6 th Semester	Course: Computer Engineering
Laboratory No: L001	Name of Subject Teacher: Sangeeta Wankhede
Name of Student : Darshan Bhangale	Roll Id: 22203A0039

Experiment No:	14
Title of Experiment	Write a program in Python to demonstrate the following operations: · (a) Method overloading · (b) Method overriding

- **Practical Related Questions**

1. State the difference between method overriding and overloading

→

The method overriding in Python means creating two methods with the same name but differ in the programming logic. The concept of Method overriding allows us to change or override the Parent Class function in the Child Class.

Method overloading, on the other hand, involves defining multiple methods with the same name within a class, but with different parameter lists. This allows the methods to perform different tasks based on the number or types of parameters passed to them.

2. What is the output of the following program?

→

Code

```
# parent class
class Animal:
    # properties
    multicellular = True
    # Eukaryotic means Cells with Nucleus
    eukaryotic = True
    # function breath
    def breathe(self):
        print("I breathe oxygen.")
    # function feed
    def feed(self):
        print("I eat food.")
# child class
class Herbivorous(Animal):
    # function feed
    def feed(self):
        print("I eat only plants. I am vegetarian.")
herbi = Herbivorous()
herbi.feed()
# calling some other function
herbi.breathe()
```

Output:

```
Output
I eat only plants. I am vegetarian.
I breathe oxygen.

=== Code Execution Successful ===
```

- **Exercise**

1. Write a Python program to create a class to print an integer and a character with two methods having the same name but different sequence of the integer and the character parameters. For example, if the parameters of the first method are of the form (int n, char c), then that of the second method will be of the form (char c, int n)

→

Code:

```
class Printer:
    def show(self, n, c):
        print("Integer:", n, "Character:", c)

    def show_reverse(self, c, n):
        print("Character:", c, "Integer:", n)
```

```
obj = Printer()
```

```
obj.show(39, 'D')
```

```
obj.show_reverse('B', 39)
```

Output:

```
Output
Integer: 39 Character: D
Character: B Integer: 39

=== Code Execution Successful ===
```

2. Write a Python program to create a class to print the area of a square and a rectangle. The class has two methods with the same name but different number of parameters. The method for printing area of rectangle has two parameters which are length and breadth respectively while the other method for printing area of square has one parameter which is side of square.

→

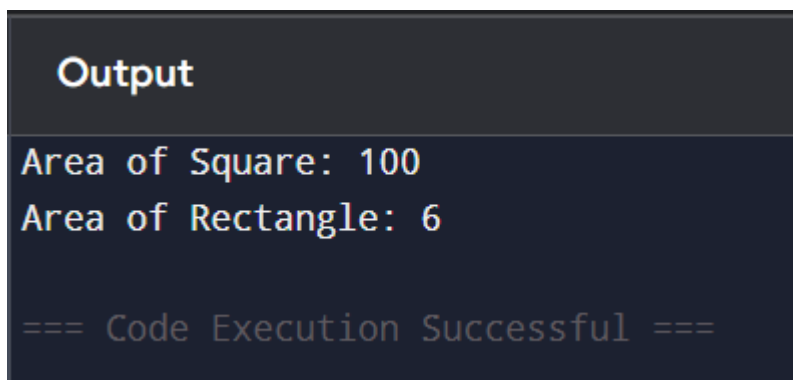
Code:

```
class Shape:
    def area(self, *args):
        if len(args) == 1:
            side = args[0]
            print("Area of Square:", side * side)
        elif len(args) == 2:
            length, breadth = args
            print("Area of Rectangle:", length * breadth)
        else:
            print("Invalid number of arguments!")

obj = Shape()

obj.area(10)
obj.area(2,3)
```

Output:



The screenshot shows a dark-themed window titled "Output". It contains the following text: "Area of Square: 100", "Area of Rectangle: 6", and "=== Code Execution Successful ===".

3. Write a Python program to create a class 'Degree' having a method 'getDegree' that prints 'I got a degree'. It has two subclasses namely 'Undergraduate' and 'Postgraduate' each having a method with the same name that prints 'I am an Undergraduate' and 'I am a Postgraduate' respectively. Call the method by creating an object of each of the three classes.

→

Code:

```
class Degree:
    def getDegree(self):
        print("I got a degree")

class Undergraduate(Degree):
    def getDegree(self):
        print("I am an Undergraduate")

class Postgraduate(Degree):
    def getDegree(self):
        print("I am a Postgraduate")

d = Degree()
d.getDegree()

ug = Undergraduate()
ug.getDegree()

pg = Postgraduate()
pg.getDegree()
```

Output:

Output

```
I got a degree
I am an Undergraduate
I am a Postgraduate
```

```
=== Code Execution Successful ===
```

Subject: Programming with python	Subject Code: 22616
Semester: 6 th Semester	Course: Computer Engineering
Laboratory No: L001B	Name of Subject Teacher: Sangeeta Wankhede
Name of Student: Darshan Bhangale	Roll Id: 22203A0039

Experiment No:	15
Title of Experiment	Write a program in Python to demonstrate following operations: Simple inheritance, Multiple inheritance

- **Practical related Questions**

1.State the use of inheritance



- Code Reusability – Avoids code duplication by allowing child classes to inherit methods and attributes from the parent class.
- Extensibility – Child classes can override or extend the functionalities of the parent class.
- Maintainability – Makes code easier to maintain and update by organizing related classes in a hierarchy.
- Encapsulation & Abstraction – Helps in designing a well-structured object-oriented program.

2. List different types of inheritance



Types of Inheritance in Python

Python supports five types of inheritance:

Single Inheritance – A child class inherits from a single parent class.

1. Multiple Inheritance – A child class inherits from more than one parent class.
2. Multilevel Inheritance – A class is derived from another derived class, forming a chain.
3. Hierarchical Inheritance – Multiple child classes inherit from the same parent class.

- **EXERCISE**

1. **Create a class Employee with data members: name, department and salary. Create suitable methods for reading and printing employee information**



```
class Employee:
```

```
    def __init__(self, name, department, salary):
```

```
        self.name = name
```

```
        self.department = department
```

```
        self.salary = salary
```

```
    def display_info(self):
```

```
        print("Employee Name:", self.name)
```

```
        print("Department:", self.department)
```

```
        print("Salary:", self.salary)
```

```
emp1 = Employee("Darshan Bhangale", "Testing", 50000)
```

```
emp1.display_info()
```


OUTPUT:

```
Output
Employee Name: Darshan Bhangale
Department: Testing
Salary: 50000

=== Code Execution Successful ===
```

2. Python program to read and print students information using two classes using simple inheritance.



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

    def display_info(self):
        print("Student Name:", self.name)
        print("Age:", self.age)

class CollegeStudent(Student):
    def __init__(self, name, age, course):
        super().__init__(name, age)
        self.course = course

    def display_info(self):
        super().display_info()
        print("Course:", self.course)

student1 = CollegeStudent("Darshan", 18, "Computer Engineering")
student1.display_info()
```

OUTPUT:

```
Output
Student Name: Darshan
Age: 18
Course: Computer Engineering

=== Code Execution Successful ===
```

2. Write a Python program to implement multiple inheritance

→

```
class Parent1:
    def feature1(self):
        print("Feature 1 from Parent1")

class Parent2:
    def feature2(self):
        print("Feature 2 from Parent2")

class Child(Parent1, Parent2):
    def feature3(self):
        print("Feature 3 from Child")
obj = Child()

obj.feature1()
obj.feature2()
obj.feature3()
```

OUTPUT:

Output

Feature 1 from Parent1

Feature 2 from Parent2

Feature 3 from Child

=== Code Execution Successful ===

Subject: Programming with python	Subject Code: 22616
Semester: 6 th Semester	Course: Computer Engineering
Laboratory No: L001B	Name of Subject Teacher: Sangeeta Wankhede
Name of Student: Darshan Bhangale	Roll Id: 22203A0039

Experiment No:	16
Title of Experiment	Write a program in Python to handle user defined exception for given problem

• Practical related Questions

1. State Exception

→ An **exception** is an **unexpected event or error** that occurs during program execution, disrupting the normal flow of the program. Exceptions occur due to issues like invalid input, division by zero, or accessing an undefined variable.

2. How to handle exception in Python?

→ In Python, exceptions are handled using the **try-except** block. This prevents the program from crashing when an error occurs.

Syntax:

try:

Code that may cause an exception

except ExceptionType:

Code to handle the exception

Example:

try:

```
a = int(input("Enter numerator: "))
```

```
b = int(input("Enter denominator: "))
```

```
result = a / b
```

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

except ZeroDivisionError:

```
print("Error: Division by zero is not allowed!")
```

except ValueError:

```
print("Error: Please enter only numeric values!")
```

except Exception as e:

```
print(f" An unexpected error occurred: {e}")
```

- **EXERCISE**

1. Write a Python program to Check for ZeroDivisionError Exception

→

try:

```
num1 = float(input("Enter numerator: "))
```

```
num2 = float(input("Enter denominator: "))
```

```
result = num1 / num2
```

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

except ZeroDivisionError:

```
print("Error: Division by zero is not allowed!")
```

except ValueError:

```
print("Error: Invalid input! Please enter numeric values.")
```

OUTPUT:

```
Output
Enter numerator: 50
Enter denominator: 0
ERROR!
Error: Division by zero is not allowed!

=== Code Execution Successful ===
```

2. Write a Python program to create user defined exception that will check whether the password is correct or not?

→

```
class InvalidPasswordException(Exception):
    def __init__(self, message="Invalid Password!"):
        self.message = message
        super().__init__(self.message)

def check_password(password):
    correct_password = "darshan@123"
    if password != correct_password:
        raise InvalidPasswordException("Incorrect password! Please try again.")
    else:
        print("Login successful!")

try:
    user_password = input("Enter password: ")
    check_password(user_password)
except InvalidPasswordException as e:
    print(e)
```

OUTPUT:

Output

```
Enter password: DArshan  
Incorrect password! Please try again.
```

```
=== Code Execution Successful ===
```

Output

```
Enter password: darshan@123  
Login successful!
```

```
=== Code Execution Successful ===
```