WEEK-01 - 05 December, 2025

1.Q - Write a python script to illustrate data types (int, char, float, string).

PROCEDURE:

Illustrate the declaration and use of fundamental data types:

- Integer (int)
- Floating-point number (float)
- String (str)

Utilize the built-in type() function to dynamically determine and display the data type (class) associated with each assigned variable at runtime

CODE:

```
age = 20
print("Age:", age)
print("Data type of age:", type(age))
height = 5.7
print("\nHeight:", height)
print("Data type of height:", type(height))
grade = 'A'
print("\nGrade:", grade)
print("Data type of grade:", type(grade))
name = "Ramya"
print("\nName:", name)
print("Data type of name:", type(name))
```

OUTPUT:

```
Age: 20
Data type of age: <class 'int'>
Height: 5.7
Data type of height: <class 'float'>
Grade: A
Data type of grade: <class 'str'>
Name: Ramya
```

Data type of name: <class 'str'>

2.Q- Write a python program to perform the following expressions using operator precedence

(1) 5+3*2

(2) 2*3**2

(3) 2**3**2

(4) (2**3)**2

PROCEDURE:

1. Open a Python IDE or text editor.
2. Assign expressions to variables:

   expr1 = 5 + 3 * 2

   expr2 = 2 * 3 ** 2

   expr3 = 2 ** 3 ** 2

   expr4 = (2 ** 3) ** 2
3. Print the value and type of each expression:

   print(expr1, type(expr1))

   print(expr2, type(expr2))

   print(expr3, type(expr3))

   print(expr4, type(expr4))
4. Save and run the program.

CODE:

```
result1 = 5 + 3 * 2
print("5 + 3 * 2 =", result1)
result2 = 2 * 3 ** 2
print("2 * 3 ** 2 =", result2)
result3 = 2 ** 3 ** 2
print("2 ** 3 ** 2 =", result3)

result4 = (2 ** 3) ** 2
print("(2 ** 3) ** 2 =", result4)
```

OUTPUT:

5 + 3 * 2 = 11

2 * 3 ** 2 = 18

2 ** 3 ** 2 = 512

(2 ** 3) ** 2 = 64

3.Q- Write a python program to illustrate type conversion functions.

PROCEDURE:

1. Open Python IDE or editor.
2. Convert integer to float and print.
3. Convert float to integer and print.
4. Convert integer to string and print.
5. Convert string to integer and print.
6. Convert string to float and print.
7. Save and run the program.

CODE:

```
a = 10
b = float(a)
print("Integer to Float:", b)
print("Type:", type(b))
x = 12.8
y = int(x)
print("\nFloat to Integer:", y)
print("Type:", type(y))
num = 25
s = str(num)
print("\nInteger to String:", s)
print("Type:", type(s))
n = "50"
```

```
m = int(n)
print("\nString to Integer:", m)
print("Type:", type(m))
f = "3.14"
g = float(f)
print("\nString to Float:", g)
print("Type:", type(g))
```

OUTPUT:

Integer to Float: 10.0

Type: <class 'float'>

Float to Integer: 12

Type: <class 'int'>

Integer to String: 25

Type: <class 'str'>

String to Integer: 50

Type: <class 'int'>

String to Float: 3.14

Type: <class 'float'>

4.Q- Write a python program to illustrate pi, sqrt, cos, sin functions of math module.

PROCEDURE:

1. Open Python IDE or editor.
2. Import the math module.
3. Print the value of pi.
4. Calculate and print the square root of a number.
5. Calculate and print the cosine of an angle.
6. Calculate and print the sine of an angle.
7. Save and run the program.

CODE:

```
import math
print("Value of pi:", math.pi)
num = 25
print("Square root of", num, "is:", math.sqrt(num))
angle = 0
print("Cos of", angle, "is:", math.cos(angle))
print("Sin of", angle, "is:", math.sin(angle))
```

OUTPUT:

Value of pi: 3.141592653589793

Square root of 25 is: 5.0

Cos of 0 is: 1.0

Sin of 0 is: 0.0

INFERENCES/CONCLUSION:

1. Data Types:
   - Python has int, float, char (as string), and string types.
   - Use type() to check the data type of a variable.
2. Operator Precedence:
   - ** has highest priority, then *, /, then +, -.
   - Parentheses () can change the order of evaluation.
3. Type Conversion:
   - int(), float(), str() can convert values between types.
   - Helps in performing operations with compatible types.
4. Math Module:
   - math.pi gives value of π.
   - math.sqrt() calculates square root.
   - math.cos() and math.sin() give trigonometric values in radians.

WEEK-02 - 12 December, 2025

1.Q- Write a program to calculate simple interest.

PROCEDURE:

1. Open Python IDE or editor.
2. Input principal, rate, and time from user.
3. Calculate simple interest using:
   (principal * rate * time) / 100
4. Print the simple interest.
5. Save and run the program.

CODE:

```
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the rate of interest: "))
time = float(input("Enter the time (in years): "))
simple_interest = (principal * rate * time) / 100
print("Simple Interest is:", simple_interest)
```

OUTPUT:

```
Enter the principal amount: 1000
Enter the rate of interest: 5
Enter the time (in years): 2
Simple Interest is: 100.0
```

2.Q - Write a python program to calculate compound interest.

PROCEDURE:

1. Open Python IDE or editor.
2. Input principal, rate, and time from user.
3. Calculate total amount using formula: principal * (1 + rate/100) ** time
4. Calculate compound interest: amount - principal
5. Print compound interest and total amount.
6. Save and run the program.

CODE:

```
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the rate of interest: "))
time = float(input("Enter the time (in years): "))
amount = principal * (1 + rate / 100) ** time
compound_interest = amount - principal
print("Compound Interest is:", compound_interest)
print("Total Amount is:", amount)
```

OUTPUT:

```
Enter the principal amount: 1000
Enter the rate of interest: 10
Enter the time (in years): 2
Compound Interest is: 210.0
Total Amount is: 1210.0
```

3.Q - Write a python program to print ASCII value of a character and vice versa.

PROCEDURE:

1. Open Python IDE or editor.
2. Input a character from the user and print its ASCII value using ord().
3. Input an ASCII value from the user and print its corresponding character using chr().
4. Save and run the program.

CODE:

```
ch = input("Enter a character: ")
print("ASCII value of", ch, "is:", ord(ch))

num = int(input("Enter an ASCII value: "))
print("Character for ASCII value", num, "is:", chr(num))
```

OUTPUT:

Enter a character: A

ASCII value of A is: 65

Enter an ASCII value: 97

Character for ASCII value 97 is: a

4.Q - Write a python program to find the area of a circle.

PROCEDURE:

1. Open Python IDE or editor.
2. Import the math module.
3. Input the radius of the circle from the user.
4. Calculate area using formula: math.pi * radius * radius
5. Print the area of the circle.
6. Save and run the program.

CODE:

```
import math

radius = float(input("Enter the radius of the circle: "))

area = math.pi * radius * radius

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

OUTPUT:

Enter the radius of the circle: 7

Area of the circle is: 153.93804002589985

5.Q- Write a python program to find the area of a triangle.

PROCEDURE:

1. Open Python IDE or editor.

2. Input the base and height of the triangle from the user.

3. Calculate area using formula: 0.5 * base * height

4. Print the area of the triangle.

5. Save and run the program.

CODE:

```
base = float(input("Enter the base of the triangle: "))
height = float(input("Enter the height of the triangle: "))
area = 0.5 * base * height
print("Area of the triangle is:", area)
```

OUTPUT:

Enter the base of the triangle: 10

Enter the height of the triangle: 5

Area of the triangle is: 25.0

6.Q - Write a program to perform string concatenation.

PROCEDURE:

1. Open Python IDE or editor.

2. Input the first and second strings from the user.

3. Concatenate the strings using + operator.

4. Print the concatenated string.

5. Save and run the program.

CODE:

```
str1 = input("Enter first string: ")
str2 = input("Enter second string: ")
result = str1 + str2
print("Concatenated string:", result)
```

OUTPUT:

Enter first string: Hello

Enter second string: World

Concatenated string: HelloWorld

INFERENCES/CONCLUSION:

1. Simple Interest:
   - Calculates interest using formula (P * R * T) / 100.
   - Helps in understanding basic arithmetic operations and user input.
2. Compound Interest:
   - Calculates interest using formula: A = P * (1 + R/100)^T.
   - Demonstrates use of exponentiation and arithmetic operations.
3. ASCII Value:
   - ord() gives ASCII value of a character.
   - chr() converts ASCII value to character.
   - Useful for understanding character encoding.
4. Area of a Circle:
   - Uses formula: π * r^2.
   - Demonstrates use of math module and floating-point calculations.
5. Area of a Triangle:
   - Uses formula: 0.5 * base * height.
   - Helps practice arithmetic operations and user input.
6. String Concatenation:
   - Combines two strings using + operator.
   - Demonstrates string operations and handling user input.

WEEK-03 - 19 December, 2025

1.Q - Write a program to work with 1D array operations including indexing and slicing.

PROCEDURE:

1. Open Python IDE or editor.
2. Import the numpy module as np.
3. Create arrays using np.array(), np.linspace(), and np.arange().
4. Create arrays of zeros and ones using np.zeros() and np.ones().
5. Check array properties like shape and size.
6. Perform sorting and create identity matrix using np.sort() and np.eye().
7. Create arrays with different data types using dtype.
8. Perform arithmetic operations on arrays: addition, subtraction, multiplication, division.
9. Access array elements using indexing and slicing.
10. Print all results.
11. Save and run the program.

CODE:

```
import numpy as np
a = np.array([10, 20, 30, 40, 50])
print(a)
print(type(a))
b = np.linspace(1, 10, 6)
print(b)
a1 = np.arange(10)
a2 = np.arange(1, 5)
a3 = np.arange(10, 0, -1)
print(a1)
print(a2)
print(a3)
l = np.zeros(5)
p = np.ones(5)
```

```
print(l)
print(p)
k = np.array([1, 2, 3, 4])
print(k.shape)
print(k.size)
k = np.array([63, 2, 56, 100, 8])
print(np.sort(k))
print(np.eye(3))
print(np.array([1, 2, 3], dtype=float))
print(np.array([1, 2, 3], dtype=complex))
print(np.array([1, 2, 3], dtype=str))
print(np.array([1, 2, 3], dtype=bool))
print(np.array([1, 2, 3], dtype=object))
g = np.array([1, 2, 3, 4, 5])
g = g + 5
print("addition:", g)
g = g - 1
print("subtraction:", g)
g = g * 2
print("multiplication:", g)
g = g / 2
print("division:", g)
arr = np.array([10, 20, 30, 63, 21])
print(arr[0])
print(arr[-1])
print(arr[::2])
print(arr[1:3])
```

OUTPUT:

```
[10 20 30 40 50]
<class 'numpy.ndarray'>
```

[1. 2.8 4.6 6.4 8.2 10.]

[0 1 2 3 4 5 6 7 8 9]

[1 2 3 4]

[10 9 8 7 6 5 4 3 2 1]

[0. 0. 0. 0. 0.]

[1. 1. 1. 1. 1.]

(4,)

4

[2   8 56 63 100]

[[1. 0. 0.]

 [0. 1. 0.]

 [0. 0. 1.]]

[1. 2. 3.]

[1.+0.j 2.+0.j 3.+0.j]

['1' '2' '3']

[True True True]

[1 2 3]

addition: [6 7 8 9 10]

subtraction: [5 6 7 8 9]

multiplication: [10 12 14 16 18]

division: [5. 6. 7. 8. 9.]

10

21

[10 30 21]

[20 30]

2.Q - Write a program to work with 2D array operations.

PROCEDURE:

1. Open Python IDE or editor.

2. Import the numpy module as np.

3. Create 2D arrays using np.array().

4. Print the arrays.

5. Check array properties: shape, size, and dimensions.

6. Access elements and slices of arrays using indexing and slicing.

7. Perform element-wise operations: addition, subtraction, multiplication, division.

8. Perform matrix multiplication using @ or np.dot().

9. Find transpose of an array using .T.

10. Find max, min, and mean using np.max(), np.min(), np.mean().

11. Create arrays of zeros and ones using np.zeros() and np.ones().

12. Create a range of numbers using np.arange() and reshape using .reshape().

13. Print all results.

14. Save and run the program.

CODE:

```
import numpy as np
a = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])
b = np.array([[9, 8, 7],
              [6, 5, 4],
              [3, 2, 1]])
print(a)
print(b)
print("shape of a:",a.shape)
print("size of a:",a.size)
print("dimensions of a:",a.ndim)
print("array accessing:")
print(a[0, 0])
print(a[1])
print(a[:, 1])
print(a[0:2, 1:3])
```

```
print(“addition:”,a + b)
print(“subtraction:”,a - b)
print(“multiplication:”,a * b)
print(“division:”,a / b)
print(“matrix multiplication:”,a@b)
print(np.dot(a, b))
print(a.T)
print(“max of a:”,np.max(a))
print(“min of a:”,np.min(a))
print(“mean of a:”,np.mean(a))
print(np.zeros((2, 3)))
print(np.ones((3, 2)))
c = np.arange(1, 13)
c = c.reshape(3, 4)
print(c)
```

OUTPUT:

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[9 8 7]
 [6 5 4]
 [3 2 1]]
shape of a: (3, 3)
size of a: 9
dimensions of a: 2
array accessing:
1
[4 5 6]
[2 5 8]
[[2 3]
```

```
 [5 6]]
addition: [[10 10 10]
 [10 10 10]
 [10 10 10]]
subtraction: [[-8 -6 -4]
 [-2  0  2]
 [ 4  6  8]]
multiplication: [[ 9 16 21]
 [24 25 24]
 [21 16  9]]
division: [[0.11111111 0.25       0.42857143]
 [0.66666667 1.         1.5        ]
 [2.33333333 4.         9.        ]]
matrix multiplication: [[ 30  24  18]
 [ 84  69  54]
 [138 114  90]]
[[ 30  24  18]
 [ 84  69  54]
 [138 114  90]]
[[1 4 7]
 [2 5 8]
 [3 6 9]]
max of a: 9
min of a: 1
mean of a: 5.0
[[0. 0. 0.]
 [0. 0. 0.]]
[[1. 1.]
 [1. 1.]
 [1. 1.]]
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

INFERENCES/CONCLUSION:

1. 1D Array Operations:
   - Arrays store multiple values in a single variable.
   - Indexing allows access to individual elements.
   - Slicing allows access to a range of elements.
   - Useful for performing arithmetic operations on arrays.
2. 2D Array Operations:
   - 2D arrays (matrices) store data in rows and columns.
   - Supports element access, slicing, and arithmetic operations.
   - Matrix multiplication can be done using @ or np.dot().
   - Useful for mathematical and scientific computations.

WEEK-04 - 26 December, 2025

1.Q - Write a python program find the power of a number without built in functions.

PROCEDURE:

1. Open Python IDE or editor.
2. Input the base and exponent from the user.
3. Initialize result as 1.
4. Multiply result by base in a loop running exponent times.
5. Print the final power value.
6. Save and run the program.

CODE:

```
base = int(input("Enter the base: "))
exponent = int(input("Enter the exponent: "))
result = 1
for i in range(exponent):
    result = result * base
print("Power =", result)
```

OUTPUT:

```
Enter the base: 2
Enter the exponent: 5
Power = 32
```

2.Q- Write a python program to count the number of even and odd numbers upto the given range.

PROCEDURE:

1. Open Python IDE or editor.
2. Input the range (n) from the user.
3. Initialize counters for even and odd numbers.
4. Use a loop from 1 to n:
   - If the number is divisible by 2, increment even counter.
   - Otherwise, increment odd counter.
5. Print the count of even and odd numbers.

6. Save and run the program.

CODE:

```
n = int(input("Enter the range: "))
even = 0
odd = 0
for i in range(1, n + 1):
    if i % 2 == 0:
        even = even + 1
    else:
        odd = odd + 1
print("Even numbers count:", even)
print("Odd numbers count:", odd)
```

OUTPUT:

Enter the range: 10

Even numbers count: 5

Odd numbers count: 5

3.Q- Write a python program to print the multiplication table for a given number.

PROCEDURE:

1. Open Python IDE or editor.
2. Input a number from the user.
3. Use a loop from 1 to 10 to multiply the number.
4. Print each product in table format.
5. Save and run the program.

CODE:

```
n = int(input("Enter a number: ")
for i in range(1, 11):
    print(n, "x", i, "=", n * i)
```

OUTPUT:

Enter a number: 5

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

5 x 4 = 20

5 x 5 = 25

5 x 6 = 30

5 x 7 = 35

5 x 8 = 40

5 x 9 = 45

5 x 10 = 50

4.Q- Write a python program to display minimum and maximum among three numbers.

PROCEDURE:

1. Open Python IDE or editor.
2. Input three numbers from the user.
3. Initialize maximum and minimum with the first number.
4. Compare other numbers to update maximum and minimum.
5. Print the maximum and minimum numbers.
6. Save and run the program.

CODE:

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
c = int(input("Enter third number: "))

maximum = a
minimum = a
if b > maximum:
    maximum = b
if c > maximum:
```

```
    maximum = c
if b < minimum:
    minimum = b
if c < minimum:
    minimum = c
print("Maximum number:", maximum)
print("Minimum number:", minimum)
```

OUTPUT:

Enter first number: 12

Enter second number: 5

Enter third number: 20

Maximum number: 20

Minimum number: 5

INFERENCES/CONCLUSION:

1. Power of a Number:
   - Calculates power using a loop without built-in functions.
   - Demonstrates use of loops and repeated multiplication.
2. Count Even and Odd Numbers:
   - Uses a loop and modulo (%) operator to count even and odd numbers.
   - Helps understand conditionals and iteration.
3. Multiplication Table:
   - Prints the multiplication table of a given number using a loop.
   - Useful for practicing loops and formatted output.
4. Minimum and Maximum of Three Numbers:
   - Uses comparison operators to find the largest and smallest numbers.
   - Demonstrates conditional logic and variable updating.

WEEK-05 - 2 JANUARY, 2025

1.Q - Write a python program to find if a number is prime or not with and without recursion.

PROCEDURE:

1. Open Python IDE or editor.
2. Input a number from the user.
3. Without Recursion:
   - Check if the number is less than or equal to 1 → not prime.
   - Use a loop from 2 to √number to check divisibility.
   - If divisible, the number is not prime.
   - Otherwise, it is prime.
4. With Recursion:
   - Define a recursive function that checks divisibility from 2 up to √number.
   - If divisible, return False.
   - If all checks pass, return True.
5. Print whether the number is prime or not.
6. Save and run the program.

CODE:

Without Recursion

```
num = int(input("Enter a number: "))
is_prime = True
if num <= 1:
    is_prime = False
else:
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            is_prime = False
            break

if is_prime:
```

```
    print(num, "is a prime number")
else:
    print(num, "is not a prime number")
```

With Recursion

```
def is_prime_recursive(n, i=2):
    if n <= 2:
        return True if n == 2 else False
    if n % i == 0:
        return False
    if i * i > n:
        return True
    return is_prime_recursive(n, i + 1)
num = int(input("Enter a number: "))
if is_prime_recursive(num):
    print(num, "is a prime number")
else:
    print(num, "is not a prime number")
```

OUTPUT:

Enter a number: 7

7 is a prime number

Enter a number: 10

10 is not a prime number

2.Q- Write a python program to display Fibonacci series using iteration and recursion.

PROCEDURE:

1. Open Python IDE or editor.
2. Input the number of terms from the user.
3. Using Iteration:
   - Initialize first two numbers as 0 and 1.

- Use a loop to generate and print Fibonacci numbers.

4. Using Recursion:

- Define a recursive function to return Fibonacci numbers.
- Call the function in a loop to print the series.

5. Display the Fibonacci series.

6. Save and run the program.

CODE:

Fibonacci Series using Iteration

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

Fibonacci Series using Recursion

```
def fibonacci(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)
n = int(input("Enter the number of terms: "))
print("Fibonacci series:")
for i in range(n):
    print(fibonacci(i), end=" ")
```

OUTPUT:

```
Enter the number of terms: 7
Fibonacci series:
```

0 1 1 2 3 5 8

3.Q - Write a python program to find the factorial of a number with and without recursion.

PROCEDURE:

1. Open Python IDE or editor.
2. Input a number from the user.
3. Without Recursion:
   - Initialize factorial as 1.
   - Use a loop from 1 to the number.
   - Multiply each value to get factorial.
4. With Recursion:
   - Define a recursive function.
   - If number is 0 or 1, return 1.
   - Otherwise, return number multiplied by factorial of (number - 1).
5. Print the factorial result.
6. Save and run the program.

CODE:

Without Recursion (Using Loop)

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

With Recursion

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
num = int(input("Enter a number: "))
```

```
print("Factorial =", factorial(num))
```

OUTPUT:

Enter a number: 5

Factorial = 120

INFERENCES/CONCLUSION:

1. Prime Number:
   - A prime number is divisible only by 1 and itself.
   - It can be checked using loops or using recursion.
   - Both methods give the same result.
2. Fibonacci Series:
   - Fibonacci series is generated by adding the previous two numbers.
   - Iteration method uses loops and is faster.
   - Recursion method uses function calls.
   - Both methods produce the same Fibonacci series.
3. Factorial of a Number:
   - Factorial is the product of all positive integers up to a given number.
   - It can be calculated using loop or recursion.
   - Both methods give the same factorial value.