WEEK NO: 01

DATE: 05 December, 2025

SUBJECT: Python Programming Lab

---

1. QUESTION

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

PROCEDURE

1. Illustrate the declaration and use of fundamental data types:

- Integer (**int**)
- Character (**char**)
- Floating-point number (**float**)
- String (**str**)

2. By utilizing the built-in **type()** function to determine the data type dynamically and display the data type associated with each assigned variable at runtime.

PROGRAM

```
a = 8523
b = 9999.99
c = '$'
d = "Hey! Whatcha doing?"
print("Integer data type:", a, "\t", type(a))
print("Float data type:", b, "\t", type(b))
print("Character  data type:", c, "\t", type(c))
print("String  data type:", d, "\t", type(d))
```

OUTPUT / OBSERVATION

```
Integer data type: 8523     <class 'int'>
Float data type: 9999.99   <class 'float'>
Character data type: $      <class 'str'>
String data type: Hey! Whatcha doing?   <class 'str'>
```

INFERENCE / CONCLUSION

- ✧ After ',' in print statement, you automatically prints a single space before displaying the actual content after comma.
- ✧ Every primary data type is a class in python. Hence when you check the data type , it gives <class 'data_type'>
- ✧ Print statement also automatically adds a next line unlike in C or C++.

2. QUESTION

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. Display a heading and evaluate arithmetic expressions using **print()** to observe how Python follows operator precedence.
2. Demonstrate the priority of operators such as `**`, `*`, and `+` by printing expressions without parentheses.
3. Use parentheses in an expression to show how they alter the default order of evaluation and change the result.

PROGRAM

```
print("Operator Precedence : ")
print(5 + 3 * 2)
print(2 * 3 ** 2)
print(2 ** 3 ** 2)
print((2 ** 3) ** 2)
```

OUTPUT / OBSERVATION

```
Operator Precedence :
11
18
512
64
```

INFERENCE / CONCLUSION

- ✧ The operator precedence follows () first , ** second, * and +/- at the end for the expression evaluation.
- ✧ If we get the two operators have the same priority then associativity rules are followed for the evaluation of the expression

3. QUESTION

Write a python program to illustrate type conversion functions

PROCEDURE

1. Initialize variables with integer, float, and string values to serve as inputs for type conversion.
2. Apply built-in type conversion functions such as **int(), float()**, and **str()** to convert data from one type to another.
3. Use the **print()** function to display the converted values and observe the results.

PROGRAM

```
x = 7
y = 3.8
print("Integer to float:", float(x))
print("Float to integer:", int(y))
num_str = "120"
print("String to integer:", int(num_str))
val = 25
print("Integer to string:", str(val))
z = "4.5"
print("String to float:", float(z))
```

OUTPUT / OBSERVATION

```
Integer to float: 7.0
Float to integer: 3
String to integer: 120
Integer to string: 25
String to float: 4.5
```

INFERENCE / CONCLUSION

- ✧ You cannot convert a character into integer data type. That would give ValueError. Like char_str = 'A' cannot be converted to integer using int(char_str).
- ✧ During type conversion type narrowing or type widening occurs. Type narrowing means 3.8 becoming 3.
- ✧ While Type Widening means 7 becoming 7.0

---

4. QUESTION

Write a python program to illustrate pi, sqrt, cos, sin functions of math module

PROCEDURE

1. Import the built-in **math** module using the **import** statement and assign it an alias for easier access.
2. Use **m.pi** to display the value of pi and apply **m.sqrt()** to compute the square root of a given number.
3. Apply the trigonometric functions **m.cos()** and **m.sin()** with radian values and display the results using **print()**.

PROGRAM

```
import math as m
print("Value of pi:", m.pi)
print("Square root of 16:", m.sqrt(16))
print("Value of cos 90 degrees (in radians):", m.cos(m.pi/2))
print("Value of sin 90 degrees (in radians):", m.sin(m.pi/2))
```

OUTPUT / OBSERVATION

```
Value of pi: 3.141592653589793
Square root of 16: 4.0
Value of cos 90 degrees (in radians): 6.123233995736766e-17
Value of sin 90 degrees (in radians): 1.0
```

INFERENCE / CONCLUSION

- ✧ As Cos 90° is very close to 0, but due to floating-point precision, Python shows a very small number.
- ✧ 'm' is a alias for math module , which helps us in reducing and improving code readability.
- ✧ Math module has a lot of functions and modules are built to store many such functions.

WEEK NO: 02

DATE: 12 December ,2025

SUBJECT: Python Programming Lab

---

1. QUESTION

Write a program to calculate simple interest

PROCEDURE

1. Accept the principal, time, and rate values from the user using the **input()** function and convert them using **float()**.
2. Calculate the simple interest using the formula **SI = (P × T × R) / 100**.
3. Display the calculated result using the **print()** function.

PROGRAM

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

OUTPUT / OBSERVATION

```
Enter principal amount: 2000
Enter time (in years): 2
Enter rate of interest: 5
Simple Interest is: 200.0
```

INFERENCE / CONCLUSION

- ✧ Input keyword is used to prompt the user for input and also store the input in the corresponding variable.
- ✧ Very large float values can cause precision issues.

---

2. QUESTION

Write a python program to calculate compound interest

PROCEDURE

1. Read the principal, rate, and time values from the user using the **input()** function and convert them using **float()**.
2. Apply the compound interest formula using the **exponent operator (**)**:
   CI = P × (1 + R/100)^{T} − P

3. Display the calculated compound interest using the **print()** function.

PROGRAM

```
P = float(input("Enter principal amount: "))
R = float(input("Enter rate of interest: "))
T = float(input("Enter time (in years): "))
CI = P * (1 + R / 100) ** T - P
print("Compound Interest is:", CI)
```

OUTPUT / OBSERVATION

```
Enter principal amount: 5000
Enter rate of interest: 8
Enter time (in years): 2
Compound Interest is: 832.0
```

INFERENCE / CONCLUSION

- ✧ If any string input is entered, the program will raise a **ValueError** during type conversion using **float()**.
- ✧ Incorrect values of principal, rate, or time affect the accuracy of the compound interest result.

---

3. QUESTION

Write a python program to print ASCII value of a character

PROCEDURE

1. Take a character and a numeric value from the user using the **input()** function.
2. Use the built-in **ord()** function to convert the given character into its ASCII value.
3. Use the built-in **chr()** function to convert the given numeric value into its corresponding ASCII character and display both results using **print()**.

PROGRAM

```
text = input("Enter any character or word: ")
num = int(input("Enter an ASCII value: "))
print("ASCII value of first character:", ord(text[0]))
print("Character for given ASCII value:", chr(num))
```

OUTPUT / OBSERVATION

Enter any character or word: Hello

Enter an ASCII value: 66

ASCII value of first character: 72

Character for given ASCII value: B

INFERENCE / CONCLUSION

- ✧ If an empty string is entered, accessing **text[0]** will cause an **IndexError.**
- ✧ If a non-integer or out-of-range value is given, **int()** or **chr()** may raise a **ValueError**.

---

4. QUESTION

Write a python program to find the area of a circle

PROCEDURE

1. Read the radius of the circle from the user using the **input()** function and convert it to a floating-point value. Also use pi value from math module
2. Store the value of pi as a constant and apply the area formula using the **exponent operator (**)**.
3. Display the calculated area of the circle using the **print()** function.

PROGRAM

```
import math as m
radius = float(input("Enter the radius of the circle: "))
area = m.pi * (radius ** 2)
print("The area of the circle is:", area)
```

OUTPUT / OBSERVATION

Enter the radius of the circle: 7

The area of the circle is: 153.9380400258998

INFERENCE / CONCLUSION

- ✧ A negative radius will still produce a numerical result , which can solved using conditional statements.
- ✧ We can also directly only use pi from the math module using **from math import pi**

---

5. QUESTION

Write a program whether the given number is prime or not.

PROCEDURE

1. Accept an integer from the user using the **input()** function and convert it using **int()**.
2. Check divisibility of the number from 2 up to half of the number (or √n) using a **for loop** and **if condition.**
3. Display whether the number is prime or not using the **print()** function based on the result.

PROGRAM

```
num = int(input("Enter a number: "))
if num <= 1:
    print("The number is not a prime number.")
else:
    is_prime = True
    for i in range(2, num):
        if num % i == 0:
            is_prime = False
            break
    if is_prime:
        print("The number is a prime number.")
    else:
        print("The number is not a prime number.")
```

OUTPUT / OBSERVATION

```
Enter a number: 13
The number is a prime number.
```

INFERENCE / CONCLUSION

- ✧ If the entered value is less than or equal to 1, it is automatically classified as not prime.
- ✧ You can improve the efficiency by using the square root limit which reduces unnecessary iterations.

---

6. QUESTION

Write a python program to find the area of a triangle

PROCEDURE

1. Accept the base and height of the triangle from the user using the **input()** function and convert them to floating-point values.
2. Apply the triangle area formula using arithmetic operators to compute the result.

3. Display the calculated area using the **print()** function.

PROGRAM

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

OUTPUT / OBSERVATION

Enter the base of the triangle: 10

Enter the height of the triangle: 6

The area of the triangle is: 30.0

INFERENCE / CONCLUSION

- ✧ Base and Height can be integer too but as in general we have decimal points in real life calculations , this is the best approach.
- ✧ Area can be concatenated as a string when type is converted to str.

---

7. QUESTION

Write a python program to find the area of a triangle

PROCEDURE

1. Read two separate text inputs from the user using the **input()** function.
2. Combine both strings into a single string using the + **(concatenation) operator**.
3. Display the concatenated result using the **print()** function.

PROGRAM

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

OUTPUT / OBSERVATION

Enter first string: Klein

Enter second string: Morreti

Concatenated string is: KleinMorreti

INFERENCE / CONCLUSION

- ✧ If both strings do not include spaces, the output will join them directly without separation.
- ✧ Only string data types can be concatenated, If we try to add a number without converting it to a string will raise a **TypeError**.

WEEK NO: 03

DATE: 19 December ,2025

SUBJECT: Python Programming Lab

---

1. QUESTION

Illustrate Numpy operations where you write a python program to read, process and display data

PROCEDURE

4. Accept the principal, time, and rate values from the user using the **input()** function and convert them using **float()**.
5. Calculate the simple interest using the formula **SI = (P × T × R) / 100**.
6. Display the calculated result using the **print()** function.

PROGRAM

```
import numpy as np
values = input("Enter numbers separated by space: ").split()
nums = []
for v in values:
   nums.append(int(v))
arr = np.array(nums)
print("Original array:", arr)
print("Squared values:", arr ** 2)
print("Sum of elements:", np.sum(arr))
print("Average of elements:", np.mean(arr))
```

OUTPUT / OBSERVATION

```
Enter numbers separated by space: 1 2 3 4 5
Original array: [1 2 3 4 5]
Squared values: [ 1  4  9 16 25]
Sum of elements: 15
Average of elements: 3.0
```

INFERENCE / CONCLUSION

- ✧ The **.split()** function breaks the entered string into separate values wherever there is a space and stores them in a list.
- ✧ The **append()** function adds each new value to the end of the list, helping to build the list dynamically.

- ✧ The **np.sum()** function calculates the total of all elements present in the NumPy array.
- ✧ The **np.mean()** function finds the average (sum divided by count) of the elements in the NumPy array.

---

2. QUESTION

Write a python program to access data using various numpy functions on 1D arrays.

PROCEDURE

1. Import the **NumPy** library and create a one-dimensional array using predefined values.
2. Access individual elements and a range of elements from the array using **indexing** and **slicing operations**.
3. Display the original array and the accessed data using the **print()** function.

PROGRAM

```
import numpy as np
arr = np.array([5, 15, 25, 35, 45, 55])
print("Original array:", arr)
print("Second element:", arr[1])
print("Last element:", arr[-1])
print("Elements from index 2 to 4:", arr[2:5])
print("First four elements:", arr[:4])
```

OUTPUT / OBSERVATION

```
Original array: [ 5 15 25 35 45 55]
Second element: 15
Last element: 55
Elements from index 2 to 4: [25 35 45]
First four elements: [ 5 15 25 35]
```

INFERENCE / CONCLUSION

- ✧ **Indexing** allows direct access to a single element in a NumPy array using its position.
- ✧ **Negative indexing** helps access elements from the end of the array.
- ✧ **Slicing** is used to extract a group of elements from a specified range.
- ✧ Accessing elements does not change the original array, it only retrieves the data.

---

3. QUESTION

Illustrate other built-In functions of Numpy on 2D arrays.

PROCEDURE

1. Import the **NumPy** library and create a two-dimensional array to represent a matrix.
2. Access specific elements, complete rows, complete columns, and a smaller sub-matrix using **indexing** and **slicing** techniques.
3. Display the matrix and the extracted data using the **print()** function.

PROGRAM

```
import numpy as np
mat = np.array([[2, 4, 6],
                [8, 10, 12],
                [14, 16, 18]])
print("Matrix:\n", mat)
print("Element at position (0, 1):", mat[0, 1])
print("First row:", mat[0, :])
print("Second column:", mat[:, 1])
print("Sub-matrix (rows 1-2, columns 0-1):\n", mat[1:3, 0:2])
```

OUTPUT / OBSERVATION

```
Matrix:
 [[ 2  4  6]
  [ 8 10 12]
  [14 16 18]]
Element at position (0, 1): 4
First row: [2 4 6]
Second column: [ 4 10 16]
Sub-matrix (rows 1-2, columns 0-1):
 [[ 8 10]
  [14 16]]
```

INFERENCE / CONCLUSION

- ✧ 2D indexing allows access to a single element using row and column positions.
- ✧ Row slicing helps retrieve all elements from a specific row of the matrix.
- ✧ Column slicing extracts all elements from a selected column.
- ✧ Sub-matrix slicing makes it possible to work with smaller sections of a larger matrix without changing the original data.

WEEK NO: 04

DATE: 26 December ,2025

SUBJECT: Python Programming Lab

---

1. QUESTION

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

PROCEDURE

1. Accept three numbers from the user using the **input()** function and convert them into integers.
2. Compare the three values using **conditional statements** to determine the minimum and maximum.
3. Display the smallest and largest values using the **print()** function.

PROGRAM

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
c = int(input("Enter third number: "))
minimum = a
maximum = a
if b < minimum:
    minimum = b
if c < minimum:
    minimum = c
if b > maximum:
    maximum = b
if c > maximum:
    maximum = c
print("Minimum value is:", minimum)
print("Maximum value is:", maximum)
```

OUTPUT / OBSERVATION

```
Enter first number: 12
Enter second number: 5
Enter third number: 20
Minimum value is: 5
Maximum value is: 20
```

INFERENCE / CONCLUSION

- ✧ The **if** statements are used to compare values step by step and update the current minimum and maximum.
- ✧ The **colon (:)** symbol is used after each **if** condition to indicate the beginning of an indented decision block.

---

2. QUESTION

Write a python program to count the number of even and odd numbers from a series of numbers.

PROCEDURE

1. Accept the total count of numbers from the user and repeatedly read each number using a **for loop**.
2. Use the **modulus operator** to check whether each number is even or odd with the help of an **if–else structure**.
3. Maintain separate counters for even and odd numbers and display the final counts using the **print()** function.

PROGRAM

```
count = int(input("How many numbers do you want to enter: "))
even_count = 0
odd_count = 0
for i in range(1, count + 1):
    value = int(input("Enter a number: "))
    if value % 2 == 0:
        even_count = even_count + 1
    else:
        odd_count = odd_count + 1
print("Total even numbers:", even_count)
print("Total odd numbers:", odd_count)
```

OUTPUT / OBSERVATION

```
How many numbers do you want to enter: 5
Enter a number: 12
Enter a number: 7
Enter a number: 4
Enter a number: 9
```

Enter a number: 6

Total even numbers: 3

Total odd numbers: 2

INFERENCE / CONCLUSION

- ✧ The **for loop** is used to repeatedly accept a fixed number of inputs based on the user's count.
- ✧ The **modulus operator (%)** helps determine evenness by checking the remainder when divided by 2.
- ✧ The **if–else statement** controls the flow by separating numbers into even and odd categories.

---

3. QUESTION

Write a Python program to find the power of a number without using built-in functions

PROCEDURE

1. The program accepts the base and exponent values from the user.
2. A variable is initialized to store the final result of the power calculation.
3. A loop is executed as many times as the exponent value.
4. In each iteration, the result is multiplied by the base value. After the loop ends, the final power value is displayed.

PROGRAM

```
base = int(input("Enter the base: "))
power = int(input("Enter the exponent: "))
ans = 1
for i in range(1, power + 1):
    ans = ans * base
print("Power of the number is:", ans)
```

OUTPUT / OBSERVATION

Enter the base: 3

Enter the exponent: 4

Power of the number is: 81

INFERENCE / CONCLUSION

- ✧ The program demonstrates how built-in power functions can be replaced using basic control structures.

✧ The variable storing the result changes in every iteration, showing step-by-step accumulation.

---

4. QUESTION

Write a Python program to print the multiplication table of a given number

PROCEDURE

✧ The program takes a number from the user.

✧ A **for loop** is used to repeat from 1 to 10.

✧ In each step, the number is multiplied by the loop value.

✧ The multiplication result is displayed in table format.

PROGRAM

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

OUTPUT / OBSERVATION

Enter a number: 7

Multiplication Table:

7 x 1 = 7

7 x 2 = 14

7 x 3 = 21

7 x 4 = 28

7 x 5 = 35

7 x 6 = 42

7 x 7 = 49

7 x 8 = 56

7 x 9 = 63

7 x 10 = 70

INFERENCE / CONCLUSION

✧ If the range is written wrong, the table may not print till 10.

✧ If indentation is not proper, Python will show an error.

✧ The **for loop** controls how many times the table is printed.

WEEK NO: 05

DATE: 02 January ,2025

SUBJECT: Python Programming Lab

---

1. QUESTION

Write a python program to find sum of elements in a list recursively

PROCEDURE

1. Accept a list of numbers from the user and store them in a list.
2. Define a **recursive function** that adds the first element to the sum of the remaining elements.
3. Display the final sum using the **print()** function.

PROGRAM

```
def list_sum(lst):
    if len(lst) == 0:
        return 0
    else:
        return lst[0] + list_sum(lst[1:])
values = []
n = int(input("Enter number of elements: "))
for i in range(n):
    values.append(int(input("Enter element: ")))
print("Sum of list elements:", list_sum(values))
```

OUTPUT / OBSERVATION

```
Enter number of elements: 4
Enter element: 5
Enter element: 10
Enter element: 3
Enter element: 2
Sum of list elements: 20
```

INFERENCE / CONCLUSION

✧ The recursive function processes the list by reducing its size in each call.

✧ The **base condition** stops recursion when the list becomes empty.

- ✧ The program shows how a list can be divided into **head and remaining part** for recursive processing.
- ✧ Slicing creates new sublists, which makes the logic simple but increases memory usage.

---

2. QUESTION

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

PROCEDURE

1. Read the number of terms from the user using the **input()** function.
2. Generate the Fibonacci sequence first using **iteration with a loop**, and then using a **recursive function**.
3. Display both Fibonacci series using the **print()** function.

PROGRAM

```
n = int(input("Enter number of terms: "))
print("Fibonacci series using iteration:")
a, b = 0, 1
for i in range(n):
    print(a, end=" ")
    a, b = b, a + b
print("\n\nFibonacci series using recursion:")
def fib(num):
    if num == 0:
        return 0
    elif num == 1:
        return 1
    else:
        return fib(num-1) + fib(num-2)
for i in range(n):
    print(fib(i), end=" ")
```

OUTPUT / OBSERVATION

```
Enter number of terms: 7
Fibonacci series using iteration:
0 1 1 2 3 5 8

Fibonacci series using recursion:
```

0 1 1 2 3 5 8

INFERENCE / CONCLUSION

- ✧ The **iterative approach** uses a **loop** and variable updates, making it faster and more memory-efficient.
- ✧ The **recursive approach** uses a **function calling itself**, clearly demonstrating the concept of recursion.
- ✧ The **base conditions** in the function prevent infinite recursion.
- ✧ Recursion is easier to understand conceptually, but iteration is more suitable for large values of n.

---

3. QUESTION

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

PROCEDURE

1. Accept a number from the user using the **input()** function.
2. Compute the factorial first using a **loop-based (iterative) approach** and then using a **recursive function**.
3. Display both results using the **print()** function.

PROGRAM

```
n = int(input("Enter a number: "))
# Iterative method
fact_iter = 1
for i in range(1, n + 1):
    fact_iter = fact_iter * i

print("Factorial using iteration:", fact_iter)
# Recursive method
def factorial(num):
    if num == 0 or num == 1:
        return 1
    else:
        return num * factorial(num - 1)
print("Factorial using recursion:", factorial(n))
```

OUTPUT / OBSERVATION

Enter a number: 5

Factorial using iteration: 120

Factorial using recursion: 120

INFERENCE / CONCLUSION

- ✧ The **call stack mechanism** stores each function call during recursion until the base condition is reached.
- ✧ Using both methods in one program highlights the **difference in program flow** between looping and function self-calling