**Session 1: Python Introduction**

**🔹 What is Python?**

Python is a high-level, interpreted programming language known for its simplicity and readability. It supports multiple programming paradigms including procedural, object-oriented, and functional programming.

**🔹 Key Features of Python:**

* Easy to learn and use
* Interpreted language
* Dynamically typed
* Extensive standard libraries
* Cross-platform compatibility
* Supports GUI programming
* Strong community support

**🔹 Difference Between List and Tuple:**

| **Feature** | **List** | **Tuple** |
| --- | --- | --- |
| Mutability | Mutable (can be changed) | Immutable (cannot be changed) |
| Syntax | [] | () |
| Performance | Slower | Faster |
| Use Case | When data may change | When data should remain constant |

**🔹 Difference Between Set and Dictionary:**

| **Feature** | **Set** | **Dictionary** |
| --- | --- | --- |
| Structure | Unordered collection of unique elements | Key-value pairs |
| Syntax | {1, 2, 3} | {'a': 1, 'b': 2} |
| Access | No indexing | Access via keys |

**🔹 Python Code Examples:**

**Check if a number is even or odd:**

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

if num % 2 == 0:

print("Even")

else:

print("Odd")

**Concatenate two strings:**

str1 = "Hello"

str2 = "World"

result = str1 + " " + str2

print(result)

**🔹 Applications of Python:**

* Web development (Django, Flask)
* Data science and machine learning
* Automation and scripting
* Game development
* Desktop applications
* IoT and embedded systems

**🔹 What is a Dynamically Typed Language?**

In Python, you don’t need to declare the type of a variable. The interpreter infers the type at runtime.

x = 10 # Integer

x = "Hello" # Now it's a string

**🔹 Multithreading in Python:**

Python supports multithreading using the threading module. However, due to the Global Interpreter Lock (GIL), true parallelism is limited.

import threading

def print\_numbers():

for i in range(5):

print(i)

t1 = threading.Thread(target=print\_numbers)

t1.start()

**Session 2: Conditional Operators**

**🔹 Conditional Statements:**

Used to execute code based on conditions.

if condition:

# code

elif another\_condition:

# code

else:

# code

**🔹 Difference Between if, elif, and else:**

* if: First condition check
* elif: Additional condition checks
* else: Executes if none of the above conditions are true

**🔹 Purpose of Indentation:**

Python uses indentation to define blocks of code. Incorrect indentation leads to errors.

if True:

print("Indented block")

**🔹 Nested if Statements:**

An if inside another if.

x = 10

if x &gt; 5:

if x &lt; 15:

print("x is between 5 and 15")

**🔹 Ternary Conditional Expression:**

Short form of if-else.

x = 5

result = "Even" if x % 2 == 0 else "Odd"

print(result)

**🔹 if \_\_name\_\_ == "\_\_main\_\_":**

Used to ensure that code runs only when the script is executed directly, not when imported.

def main():

print("Running script")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**🔹 Difference Between == and is:**

* ==: Compares values
* is: Compares object identity

a = [1, 2]

b = [1, 2]

print(a == b) # True

print(a is b) # False

**Session 3: Loops**

**🔹 Concept of Loops:**

Loops are used to repeat a block of code.

**For loop:**

for i in range(5):

print(i)

**While loop:**

i = 0

while i &lt; 5:

print(i)

i += 1

**🔹 Reverse from 10 to 1:**

for i in range(10, 0, -1):

print(i)

**🔹 Factorial using while loop:**

num = 5

fact = 1

while num &gt; 0:

fact \*= num

num -= 1

print("Factorial:", fact)

**🔹 Print until negative number:**

numbers = [1, 2, 3, -1, 4]

i = 0

while i &lt; len(numbers) and numbers[i] &gt;= 0:

print(numbers[i])

i += 1

**🔹 Skip even numbers using for and else:**

for i in range(1, 11):

if i % 2 == 0:

continue

print(i)

else:

print("Loop completed")

**🔹 Count digits and letters in a string:**

s = "Hello123"

letters = digits = 0

for char in s:

if char.isalpha():

letters += 1

elif char.isdigit():

digits += 1

print("Letters:", letters)

print("Digits:", digits)

**🔹 Convert month name to number of days:**

month = input("Enter month name: ").lower()

days = {

"january": 31, "february": 28, "march": 31,

"april": 30, "may": 31, "june": 30,

"july": 31, "august": 31, "september": 30,

"october": 31, "november": 30, "december": 31

}

print("Days:", days.get(month, "Invalid month"))