Module I: Basics of Python, Conditional & Control Statements [10 Hours]

---

1. Introduction to Python

History of Python:

- **Python** was created by **Guido van Rossum** during 1985-1990 at Centrum Wiskunde & Informatica (CWI) in the Netherlands.
- It was officially released in **1991**.
- The name "Python" was inspired by the British comedy series "Monty Python's Flying Circus," not the snake.

Key Features of Python:

- **Easy to Learn and Use:** Python has a simple syntax similar to English.
- **Interpreted Language:** Code is executed line-by-line.
- **Dynamically Typed:** No need to declare variable types explicitly.
- **Extensive Standard Library:** Comes with many built-in modules.
- **Portable:** Works on different platforms.
- **Supports Multiple Programming Paradigms:** Object-oriented, procedural, functional.

Setting Up Python:

- Download Python from python.org.
- Install Python and verify installation using `python --version` or `python3 --version` in terminal or command prompt.
- Use text editors or IDEs like **IDLE, Visual Studio Code, PyCharm, Jupyter Notebook** for coding.

---

2. Basic Syntax, Variables, and Data Types

Python Syntax:

- **Indentation:** Python uses whitespace to define blocks of code.
- **Case Sensitivity:** `Variable` and `variable` are different.
- **Comments:**

```
# This is a single-line comment
'''
This is a multi-line comment
'''
```

Variables:

- Python automatically identifies the type of variable based on value assigned.

```
x = 5        # integer
name = "Suva"  # string
```

Data Types:

Type	Example
int	10
float	3.14
str	"Python"
bool	True or False
list	[1, 2, 3]
tuple	(1, 2, 3)
dict	{"key": "value"}
set	{1, 2, 3}

---

3. Operators in Python

Arithmetic Operators:

`+ - * / % ** //`

Comparison Operators:

`== != > < >= <=`

Logical Operators:

`and`, `or`, `not`

Assignment Operators:

`= += -= *= /=`

Membership Operators:

`in`, `not in`

---

4. Conditional Statements

The if Statement:

```
if temperature > 30:
    print("It's hot!")
```

if-else Statement:

```
if number % 2 == 0:
    print("Even number")
else:
    print("Odd number")
```

Nested if-else:

```
if marks >= 90:
    print("Grade A")
elif marks >= 80:
    print("Grade B")
else:
    print("Grade C")
```

---

5. Looping in Python

For Loop:

Iterates over a sequence (list, tuple, string):

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

While Loop:

```
count = 0
while count < 5:
    print(count)
    count += 1
```

Nested Loops:

```
for i in range(3):
    for j in range(2):
        print(i, j)
```

---

6. Loop Control Statements

break:

- Terminates the loop prematurely.

```
for i in range(5):
    if i == 3:
        break
    print(i)
```

continue:

- Skips the current iteration and proceeds to the next.

```
for i in range(5):
    if i == 2:
        continue
    print(i)
```

pass:

- Acts as a placeholder where syntactically some code is required but no action is needed.

```
for i in range(3):
    pass
```

Summary:

- Python provides a clean and readable syntax.
- Variables, data types, and operators form the foundation.
- Conditional statements and loops allow control over the program flow.
- Practice small coding examples regularly to strengthen understanding.

End of Module I