**Module 6) Python Fundamentals**

**Introduction to Python:**

Theory:

1.Introduction to Python and its Features (simple, high-level, interpreted language.

**Python** is a **simple, high-level, interpreted** programming language. It was created by **Guido van Rossum** and released in **1991**. Python is widely used for web development, data analysis, artificial intelligence, machine learning, automation, and more.

**Key Features of Python**

1. **Simple and Easy to Learn**
   * Python’s syntax is clean and close to English.
   * Beginners find it easy to understand and write code.
2. **High-Level Language**
   * You don’t need to manage memory or understand system-level details.
   * Focus is on solving the problem, not on complex technical parts.
3. **Interpreted Language**
   * Python code is executed line by line.
   * You don’t need to compile your code before running it.

2. **History and evolution of Python**.

**Python's** journey from a niche scripting language to one of the most popular programming languages in the world is a fascinating story of continuous evolution, driven by a focus on readability, simplicity, and community involvement. Below is an overview of its history and evolution:

3. **Advantages of using Python over other programming languages**.

### ****Simple and Readable Syntax****

* Python's code looks like English.
* It's easy to read, write, and understand — even for beginners.

**Example:**

python

Copyedit

print ("Hello, World!")

1. **Writing and executing your first Python program.**

**Create a new file named** hello.py.

Code:

python

Copyedit

print ("Hello, World!")

**2. Programming Style**

**1.** Understanding Python’s PEP 8 guidelines

**PEP 8** stands for **Python Enhancement Proposal 8**.  
It is the **official style guide** for writing clean and readable Python code.

PEP 8 tells us:

* **How to write Python code properly**
* What naming style, indentation, spacing, etc., to follow
* Helps all developers write code in a **consistent** way.

**2. Indentation, comments, and naming conventions in Python**

### What is indentation?

Indentation means adding **spaces** at the beginning of a line to define **blocks of code** (like under loops, functions, if-else, etc.)

In Python, **indentation is mandatory** — it shows the structure of your code.

### Standard Rule:

* Use **4 spaces** for each level of indentation.
* Do **not mix tabs and spaces**.

**3. Writing readable and maintainable code.**

* **Readable code** = Easy to **understand** by humans
* **# Bad**
* **x = 10**
* **y = 20**
* **z = x + y**
* **# Good**
* **price = 10**
* **tax = 20**
* **total\_price = price + taxMaintainable code** = Easy to **fix**, **update**, or **extend** in the future.

**3. Core Python Concepts:**

**1.** Understanding data types: integers, floats, strings, lists, tuples, dictionaries, sets

1. **Integer (**int**)**

* Whole numbers (positive or negative)
* No decimal point

python

CopyEdit

x = 10

y = -3

**Type**: int  
Example: 5, 100, -25

2. **Float (**float**)**

* Numbers with a **decimal point**

python

pi = 3.14

temperature = -12.5

**Type**: float

Example: 2.5, 0.0, -9.81

3. **String (**str**)**

* A sequence of **characters** (letters, digits, symbols)
* Written in **quotes** (' ' or " ")

python

name = "Ruchi"

message = 'Hello, Python!'

**Type**: str

Example: "hello", '123'

4. **List (**list**)**

* A **collection** of multiple items
* **Mutable**: You can change values
* Items are in square brackets []

python

fruits = ["apple", "banana", "cherry"]

numbers = [1, 2, 3, 4]

**Type**: list

Example: [10, 20], ["a", "b"]

5. **Tuple (**tuple**)**

* Like a list, but **immutable** (can’t change values)
* Written in **round brackets** ()

python

colors = ("red", "green", "blue")

**Type**: tuple

Example: (1, 2), ("x", "y")

6. **Dictionary (**dict**)**

* Stores **key-value pairs**
* Written using **curly braces** {}

python

student = {

"name": "Ruchi",

"age": 20,

"marks": 88

}

**Type**: dict  
 Example: {"key": "value"}

7. **Set (**set**)**

* A collection of **unique** values
* Unordered (no indexing)
* Written in curly braces {}

python

unique numbers = {1, 2, 3, 4}

**Type**: set  
Example: {5, 7, 9}

**2.Python variables and memory allocation**.

| **Concept** | **Description** |
| --- | --- |
| Variable | A name that refers to a value |
| Memory Allocation | Python stores data in memory, variables point to it |
| Dynamic Typing | No need to declare data type |
| id () function | Returns memory address of a value |
| Mutable types | Can change: list, dict, set |
| Immutable types | Cannot change: int, str, tuple |

**3**. Python operators: arithmetic, comparison, logical, bitwise.

| **Category** | **Common Operators** |
| --- | --- |
| Arithmetic | +, -, \*, /, //, %, \*\* |
| Comparison | ==, !=, >, <, >=, <= |
| Logical | and, or, not |
| Bitwise | &, ` |

**4.Conditional Statements.**

**1.** Introduction to conditional statements: if, else, elif.

|  |  |
| --- | --- |
| If | -> Run block if condition is True |

|  |  |
| --- | --- |
| Elif-> | Check another condition if above is False |

|  |  |
| --- | --- |
| Else-> | Run if all above conditions are False |
| 2. Nested if-else conditions. |  |

->A **nested if-else** means placing one if (or else) **inside another.**

**5. Looping (For, While):**

**1.Loops in Python**

**Loops are used to execute a block of code repeatedly until a certain condition is met.**

**🔹 for loop:**

**Used to iterate over a sequence (like list, tuple, string, etc.).**

**Syntax:**

**python**

**for item in sequence:**

**# code block**

**Example:**

**python**

**fruits = ['apple', 'banana', 'cherry']**

**for fruit in fruits:**

**print(fruit)**

**🔹 while loop:**

**Used when you want to repeat a block of code as long as a condition is True.**

**Syntax:**

**python**

**while condition:**

**# code block**

**Example:**

**python**

**i = 1**

**while i <= 5:**

**print(i)**

**i += 1**

**✅ 2. How Loops Work in Python**

* **Start: The loop checks the condition (for while) or picks the next item (for for).**
* **Execute: If the condition is True or there's a next item, it runs the code block.**
* **Repeat: Goes back to check the next condition or item.**
* **Stop: When the condition becomes False or there are no more items.**

**✅ 3. Using Loops with Collections**

**Python collections include lists, tuples, strings, sets, dictionaries, etc.**

**✅ for loop with collections:**

**List Example:**

**python**

**numbers = [10, 20, 30]**

**for n in numbers:**

**print(n)**

**Tuple Example:**

**python**

**names = ('Alice', 'Bob', 'Charlie')**

**for name in names:**

**print(name)**

**Dictionary Example:**

**python**

**student = {'name': 'Ravi', 'age': 16}**

**for key in student:**

**print(key, ":", student[key])**

**✅ while loop with collections:**

**You can use indexing:**

**python**

**colors = ['red', 'green', 'blue']**

**i = 0**

**while i < len(colors):**

**print(colors[i])**

**i += 1**