➢ Features of Python

Definition of Python:

- Python is a **high-level, easy-to-learn programming language** that supports multiple programming styles, such as **object-oriented, structured, and functional programming**.

Features of Python

1. Interactive

- Python supports **interactive mode**, allowing you to test and debug code in real-time.
- You can execute a single line of code using the **Python shell** or **IDLE**.

Example:

```
>>> print("Hello MSBTE!")
Hello MSBTE!
```

Real-Life Use: Interactive mode helps while testing a small function in programs.

---

2. Object-Oriented

- Python supports the **object-oriented programming paradigm** (OOP), which focuses on objects.
- OOP concepts include **classes, objects, inheritance**, and polymorphism.

Example:

```
class Student:
    def __init__(self, name):
        self.name = name
s = Student("Rahul")
print(s.name)
```

Output: Rahul

Real-Life Use:

Object-oriented features are used in **game development** and **banking systems**.

---

3. Interpreted

- Python code is **executed line by line** by the Python interpreter, making debugging easier.
- It doesn't need **compilation** like Java or C++.

Example:

```
x = 10
print(x * 5)
```

Executes directly without compiling.

Real-Life Use:

Interpreted nature is useful for **quick scripting tasks**.

---

4. Platform Independent

- Python programs can run on **different operating systems** (Windows, macOS, Linux) without modification.
- The **Python interpreter** handles compatibility.

Example:

Writing the same Python code on Windows and running it on Linux without changes.

```
print("This works on all OS!")
```

Real-Life Use:

Used in **web development and data analysis** across various platforms.

Diagram

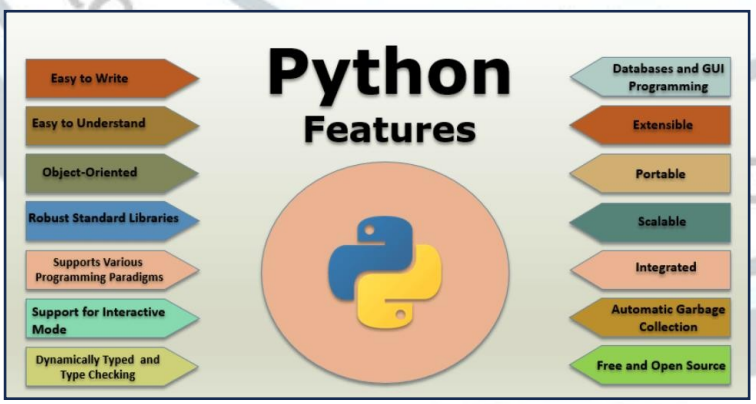

Fig (a). Python Features

Advantages of Python

1. Easy to debug due to its **interpreted nature**.
2. Flexible across platforms (**write once, run anywhere**).
3. Efficient for real-world tasks like **machine learning and web apps**.
4. Reusable and scalable, thanks to **object-oriented concepts**.

Disadvantages of Python

1. Slower than compiled languages like C.
2. Requires the **Python interpreter** to execute.
3. Not suitable for applications demanding **low-level hardware access**.

➢ Python Building Blocks

Definition of Python Building Blocks:

- Python programs are built using basic elements like **identifiers, keywords, indentation, variables, and comments**. These elements define the structure and readability of Python code.

1. Identifiers

- **Definition**:
  Identifiers are the **names** used for variables, functions, and other programming elements.
- **Rules**:
  1. Must **begin with a letter** (A-Z, a-z) or an underscore _.
  2. Cannot use **spaces or special characters** (e.g., @, $, %).
  3. Case-sensitive (age and Age are different).
- **Example**:

```
name = "Rahul"  # Valid
_id = 101       # Valid
2value = 50     # Invalid: Cannot start with a number
```

---

2. Keywords

- **Definition**:
  Keywords are reserved words in Python that have predefined meanings and **cannot be used as identifiers**.
- **Examples**: if, else, while, def, return, etc.
- **Real-Life Use**: Keywords define **control flow** or code behaviour.
- **List of Some Python Keywords**:
  False, True, None, break, continue, class, try, except.
- **Code Example**:

```
if True:
    print("MSBTE rocks!")  # 'if' is a keyword
```

---

3. Indentation

- **Definition**:
  Python uses **indentation** (spaces or tabs) to define the **block of code**, replacing {} or other markers.
- **Rules**:
  1. Indentation should be **consistent** within a block.
  2. A **colon (:)** signifies the start of an indented block.
- **Example**:

```
if True:
    print("This is indented!")  # Correct
print("This is outside!")      # Unindented
```

- **Common Error**:
  IndentationError occurs if it's inconsistent.
  Example: IndentationError: unexpected indent.

---

4. Variables

- **Definition**:
  Variables are **containers to store data**. In Python, variables are created when a value is assigned.
- **Rules**:
  1. Must follow the **identifier rules**.
  2. Can change values during program execution.
- **Example**:

```
age = 25
age = age + 1  # Variable updated
print(age)     # Output: 26
```

- **Real-Life Use**:
  Used to **store user inputs** or program data.

---

5. Comments

- **Definition**:
  Comments are lines that **start with #** and are **ignored by the Python interpreter**.
  - **Single-line comment**: Starts with #.
  - **Multi-line comment**: Written using triple quotes (''' or """).
- **Code Example**:

```
# This is a single-line comment
'''This is a
multi-line comment'''
print("Hello MSBTE!")
```

- **Real-Life Use**:
  Helps to explain the logic or mark TODOs in large programs.

Advantages of Python Building Blocks

1. Simplifies the **code structure** for better readability.
2. Reduces errors with clear indentation rules.
3. Built-in keywords help write **clearer logic**.
4. Comments make the code easy to maintain and understand.

Disadvantages

1. Overuse of comments may clutter the code.
2. Strict indentation can lead to errors if not used consistently.

➢ Python Environment Setup - Installation & Working of IDE

(Note : Ha topic warti question aata parayant paper la aala nhi just baghun ja safe side because after all aaplyala aaple nashib mahit aahe **)**

Definition of Python Environment Setup

- Python environment setup includes the process of **installing Python** on your system and using an **IDE (Integrated Development Environment)** to write, execute, and debug Python programs easily.

---

1. Installation of Python

Follow these steps to install Python:

Step 1: Download Python

- Visit the official Python website: python.org.
- Click on the **Download** button and choose the appropriate version for your operating system (Windows, macOS, or Linux).

Step 2: Install Python

- Run the downloaded installer.
- Select the option **"Add Python to PATH"** (important for command-line usage).
- Click **Install Now** and wait for the installation to complete.

Step 3: Verify Installation

- Open the **command prompt/terminal**.
- Type python --version or python3 --version.
- If Python is installed, it will display the installed version, e.g., Python 3.10.2.

---

2. Working of IDE

What is an IDE?

- An IDE (Integrated Development Environment) is software that provides tools for **writing, editing, and debugging code**. It simplifies Python programming with features like syntax highlighting and error checking.

Popular Python IDEs

1. **IDLE** (comes with Python).
2. **PyCharm** – Popular for Python programming.
3. **Visual Studio Code** – Lightweight and extensible.
4. **Jupyter Notebook** – Best for data science and research.

Diagram

A diagram showing the process:

- Installation:
  Python Download ➔ Add to PATH ➔ Verification via python --version.
- IDE Usage:
  Write Code ➔ Save File ➔ Execute Code ➔ View Output.

Advantages of Using IDEs

1. **Code Completion**: Saves time with intelligent suggestions.
2. **Debugging Tools**: Identifies errors quickly.
3. **Code Organization**: Projects are managed efficiently.
4. **User-Friendly**: Easier to use than command-line interfaces.

Disadvantages

1. **Heavy Software**: Some IDEs (e.g., PyCharm) require high system resources.
2. **Dependency on Features**: May reduce the ability to work with simple editors like Notepad.

➢ Running a Simple Python Script to Display "Welcome" Message

1. What is a Python Script?

- A **Python script** is a file containing Python code that can be executed by the Python interpreter.
- Scripts typically have the **.py extension** and are used to automate tasks or create programs.

2. Steps to Write and Run a Python Script

Step 1: Open a Text Editor or IDE

- You can use any text editor (e.g., Notepad, Visual Studio Code, or PyCharm) or the built-in **IDLE**.

Step 2: Write the Code

- Enter the Python code in the editor:

```
print("Welcome")
```

Step 3: Save the Script

- Save the file with the extension .py, e.g., welcome.py.

Step 4: Run the Script

1. **Using IDLE**:
   - Press **F5** in IDLE or click **Run > Run Module**.
   - **Output**:

     Welcome
2. **Using Command Prompt/Terminal**:
   - Navigate to the script's directory using the cd command.
   - Run the script with the following command:

     ```
     python welcome.py
     ```
   - **Output**:

     Welcome

Step 5: Verify the Output

- The message **"Welcome"** will be displayed on the screen.

❖ UNIT 01. INTRODUCTION AND SYNTAX OF PYTHON PROGRAM.

Code Example

```
# Python script to display a welcome message

print("Welcome")
```

Diagram

Illustration of the process:

Write Code (editor) ➔ Save as .py file ➔ Run Script (Terminal/IDE) ➔ View Output

---

Real-Life Uses of Python Scripts

1. Automating simple tasks, e.g., printing messages.
2. Creating introductory programs for beginners.

Advantages of Running Python Scripts

1. **Easy Execution**: Requires only Python installation.
2. **Cross-Platform Compatibility**: Run the script on any operating system.
3. **Debugging Friendly**: Easy to debug small scripts.

Disadvantages

1. Might be **difficult for beginners** without prior setup knowledge.
2. Requires Python to be installed on the system.

➢ Python Data Types

1. What are Data Types?

- **Definition**: Data types define the type of data a variable can store in Python.
- Python provides **built-in data types** to work with different kinds of data.

2. Common Python Data Types

A. Numbers

- **Definition**: Used to store numeric values.
- **Types**:
    1. **Integer**: Whole numbers, e.g., 10, -5.
    2. **Float**: Decimal numbers, e.g., 3.14, -7.5.
    3. **Complex**: Numbers with real and imaginary parts, e.g., 3+5j.
- **Example**:

```
x = 10      # Integer
y = 3.14    # Float
z = 2 + 3j  # Complex
print(x, y, z)
```

B. String

- **Definition**: A collection of characters enclosed in **single (' ')** or **double (" ") quotes**.
- **Characteristics**:
    - Immutable (cannot change the value once created).
    - Used for text processing.
- **Example**:

```
print(name)
print(name[0])    # Accessing the first character
print(name[0:3])  # Slicing
```

C. Tuples

- **Definition**: Ordered and immutable collection of items.
- **Features**:
  - Elements are enclosed in **parentheses ( )**.
  - Useful for fixed collections of items.
- **Example**:

```
fruits = ("apple", "banana", "cherry")
print(fruits[0])    # Access the first element
```

D. Lists

- **Definition**: Ordered and mutable collection of items.
- **Features**:
  - Enclosed in **square brackets [ ]**.
  - Allows duplicate elements.
- **Example**:

```
colors = ["red", "blue", "green"]
colors[1] = "yellow"  # Changing value
print(colors)
```

E. Dictionary

- **Definition**: Unordered collection of key-value pairs.
- **Features**:
  - Keys must be unique, values can be duplicate.
  - Enclosed in **curly braces { }**.
- **Example**:

```
student = {"name": "John", "age": 20, "grade": "A"}
print(student["name"])   # Accessing value by key
```

Diagram

A chart showing each datatype with examples:

Datatype	Symbol	Features	Example
Numbers	None	Integer, Float, Complex	`x = 10; y = 3.5; z = 2+3j`
String	`' '` / `" "`	Immutable, Ordered	`name = "Python"`
Tuple	`( )`	Immutable, Ordered	`fruits = ("apple", "banana")`
List	`[ ]`	Mutable, Ordered	`colors = ["red", "green"]`
Dictionary	`{ }`	Mutable, Key-Value pairs	`student = {"name": "John"}`

Advantages of Knowing Datatypes

1. Helps **store and manipulate data** efficiently.
2. Reduces errors by restricting invalid operations.
3. Facilitates advanced features like indexing and slicing.

➢ Declaration and Use of Data Types

1. Understanding Data Type Declaration in Python

- **Dynamic Typing**: Python is a **dynamically typed language**, which means you don't need to declare the data type explicitly. The interpreter automatically determines the type of the variable when you assign a value.
- **Syntax**:

  variable_name = value
- **Example:**

```
x = 10      # Integer
y = 3.14    # Float
name = "Python"  # String
```

---

2. Common Data Types in Python

A. Numbers

B. String

C. Lists

D. Tuple

E. Dictionary

---

3. Key Points on Declaration

1. **No explicit declaration**:

```
a = 10       # Python knows `a` is an integer.

b = "text"   # Python identifies `b` as a string.
```

2. **Reassignment**: Variable types can be reassigned dynamically.

```
x = 10       # Integer

x = "Python"  # Now a String
```

---

4. Advantages of Python's Data Type Declaration

1. **Simplicity**: No need for explicit data type declaration.
2. **Flexibility**: Variables can store different data types at runtime.
3. **Time-Saving**: Reduces effort in writing type definitions.

❖ UNIT 01. INTRODUCTION AND SYNTAX OF PYTHON PROGRAM.

❖ Summer 2022

1. How to give single and multiline comment in python. **(2marks)**
2. List Data types used in python. Explain any two with example. **(4marks)**

❖ Winter 2022

1. List Python features. (Any four). **(2marks)**
2. Describe indentation in Python. **(2marks)**
3. Write different data types in python with suitable example. **(6marks)**

❖ Summer 2023

1. List features of Python. **(2marks)**
2. Describe Multiline comment in python. **(2marks)**

❖ Winter 2023

1. Enlist applications for python programming. **(2marks)**
2. Describe the Role of indentation in python. **(2marks)**
3. Explain building blocks of python. **(6marks)**

❖ Summer 2024

1. What is dictionary? **(2marks)**
2. Explain with example: i) Indentation ii) Variables **(4marks)**
3. List data types used in python. Explain any two with example. **(4marks)**

❖ Winter 2024

1. List features of Python. **(2marks)**
2. State how to perform comments in Python. **(2marks)**
3. Determine various data types available in Python with example. **(6marks)**