**🐍 What is Python?**

**Python** is a high-level, interpreted programming language known for its **simplicity**, **readability**, and **versatility**. It was created by **Guido van Rossum** and first released in **1991**.

Python emphasizes **code readability** and allows developers to express concepts in fewer lines of code compared to many other languages

**✅ Why Do We Need Python?**

Here are some key reasons why Python is widely used and needed:

**1. Easy to Learn and Use**

* Clean and readable syntax (like writing English).
* Great for beginners and professionals alike.

**2. Versatile Applications**

Python is used in many domains:

* **Web Development** (e.g., Django, Flask)
* **Data Science & Machine Learning** (e.g., Pandas, NumPy, Scikit-learn, TensorFlow)
* **Automation/Scripting** (e.g., automating tasks, bots)
* **Software Development** (e.g., backend systems, APIs)
* **Cybersecurity & Ethical Hacking**
* **Game Development** (e.g., Pygame)

**3. Large Community and Libraries**

* Massive ecosystem of libraries and frameworks.
* Strong community support and documentation.

**4. Cross-Platform**

* Runs on Windows, macOS, Linux, and even mobile devices.

**5. Integration Friendly**

* Easily integrates with other languages like C, C++, Java, and .NET.

**🧠 Built-in Data Types in Python**

Here’s a categorized overview:

**🔢 Numeric Types**

* int – Integer numbers (e.g., 5, -10)
* float – Floating-point numbers (e.g., 3.14, -0.001)
* complex – Complex numbers (e.g., 2 + 3j)

**🔤 Text Type**

* str – String (e.g., "Hello", 'Python')

**📦 Sequence Types**

* list – Ordered, mutable collection (e.g., [1, 2, 3])
* tuple – Ordered, immutable collection (e.g., (1, 2, 3))
* range – Sequence of numbers (e.g., range(0, 10))

**📚 Mapping Type**

* dict – Key-value pairs (e.g., {"name": "Yuvraj", "age": 25})

**🧮 Set Types**

* set – Unordered, unique elements (e.g., {1, 2, 3})
* frozenset – Immutable version of a set

**✅ Boolean Type**

* bool – Logical values (True or False)

**🧼 None Type**

* NoneType – Represents the absence of a value (None)

**🧠 High-Level Language**

**✅ Definition:**

A high-level language is designed to be **easy for humans to read and write**. It abstracts away most of the hardware details and focuses on logic and functionality.

**✅ Examples:**

* Python
* Java
* C#
* JavaScript
* Ruby

**✅ Features:**

* Easy syntax (close to English)
* Portable across platforms
* Requires a compiler or interpreter
* Slower execution compared to low-level languages
* Ideal for application development, web, data science, etc.

**🧠 Low-Level Language**

**✅ Definition:**

A low-level language is **closer to machine code** and provides **direct control over hardware**. It’s harder for humans to read but faster and more efficient for machines.

**✅ Examples:**

* Assembly language
* Machine code (binary)

**✅ Features:**

* Fast execution
* Requires deep understanding of hardware
* Not portable (hardware-specific)
* Used in system programming, embedded systems, device drivers

| **Feature** | **High-Level Language** | **Low-Level Language** |
| --- | --- | --- |
| **Abstraction** | High (closer to human logic) | Low (closer to machine hardware) |
| **Readability** | Easy to read and write | Difficult to read, more technical |
| **Hardware Control** | Minimal control over hardware | Direct control over hardware |
| **Execution Speed** | Slower (due to abstraction) | Faster (optimized for hardware) |
| **Portability** | Highly portable across systems | Not portable, hardware-specific |
| **Ease of Learning** | Beginner-friendly | Requires deep technical knowledge |
| **Examples** | Python, Java, C#, JavaScript | Assembly, Machine Code |
| **Use Cases** | Web apps, AI, data science, automation | OS development, embedded systems, drivers |
| **Translation Required** | Needs compiler or interpreter | May be directly executed by CPU |

| **Feature** | **List** | **Tuple** |
| --- | --- | --- |
| **Syntax** | list = [1, 2, 3] | tuple = (1, 2, 3) |
| **Mutability** | ✅ Mutable (can be changed) | ❌ Immutable (cannot be changed) |
| **Methods Available** | Many (e.g., append, remove, sort) | Few (e.g., count, index) |
| **Performance** | Slightly slower (due to flexibility) | Faster (due to immutability) |
| **Use Case** | When data may change | When data should remain constant |
| **Memory Usage** | More memory | Less memory |
| **Can be used as key in dict** | ❌ No (lists are not hashable) | ✅ Yes (tuples are hashable) |

| **Feature** | **Dictionary (dict)** | **Set (set)** |
| --- | --- | --- |
| Structure | Key-value pairs | Unique values only |
| Syntax | {"key": "value"} | {value1, value2} |
| Duplicates | Keys must be unique | No duplicates allowed |
| Order | Insertion order (Python 3.7+) | Unordered |
| Use Case | Fast lookup by key | Membership tests, set operations |
| Mutability | Mutable | Mutable |

year = int(input("Enter a year: "))

**if** (year % 400 == 0) **or** (year % 4 == 0 **and** year % 100 != 0):

**print**(f"{year} is a Leap Year")

**else**:

**print**(f"{year} is not a Leap Year")