**Introduction to Python**

* Overview of Python programming language
* Applications and features
* Concepts in programming and algorithms

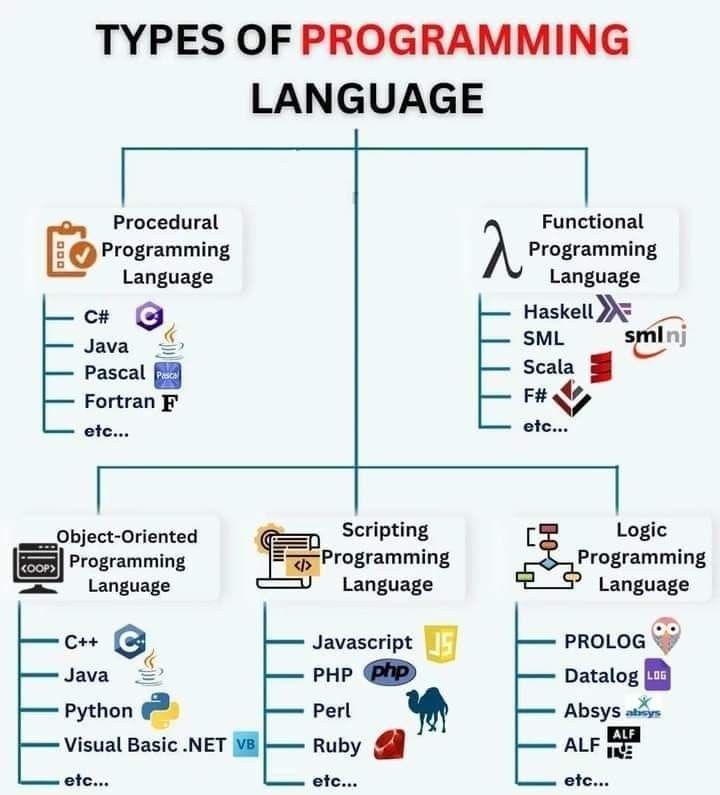
**What is Python?**

* **Definition**: Python is a high-level, interpreted programming language known for its readability and simplicity.
* **History**: Developed by Guido van Rossum and released in 1991.
* **Uses**: Web development, Data Science, AI/ML, Automation, Scripting, etc.



**Types of Programming Languages**

1. **Low-Level Languages**: Close to machine code (e.g., Assembly)
2. **High-Level Languages**: Abstracted from hardware (e.g., Python, Java)
3. **Compiled Languages**: Need compilation before running (e.g., C, C++)
4. **Interpreted Languages**: Run directly using an interpreter (e.g., Python, JavaScript)



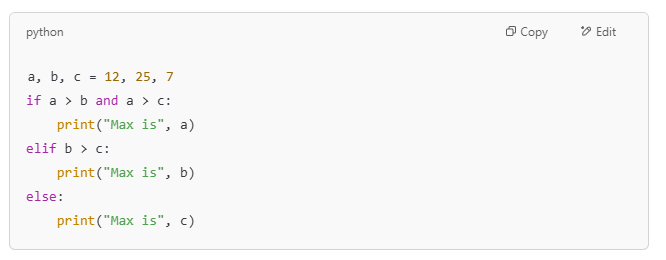
**Polyglot Concept**

* **Polyglot Programmer**: Skilled in multiple programming languages.
* **Benefits**:
  + Flexibility in development.
  + Better problem-solving.
  + Exposure to diverse paradigms.

**What is an Algorithm?**

* **Definition**: A step-by-step procedure to solve a problem or perform a task.
* **Importance**: Algorithms are the foundation of all computer programs.

**Example**:  
Find the max of 3 numbers:



**Getting Started with Python**

**Syntax & Keywords**

* Python syntax is clean and uses indentation to define code blocks.
* **Example**:

if True:

print("This is valid syntax")

**The print() Function**

* **Purpose**: Display output in the console.
* **Example**:

print("Hello, Python!")

print("Sum of 5 and 10 is", 5 + 10)

**Python Essentials**

**RAM / Memory**

* **RAM** stores variables and program state during runtime.

**Variables and id() Function**

* **Variables** store data in memory.
* **id()** returns a unique identifier for an object.

**Example**:

a = 42

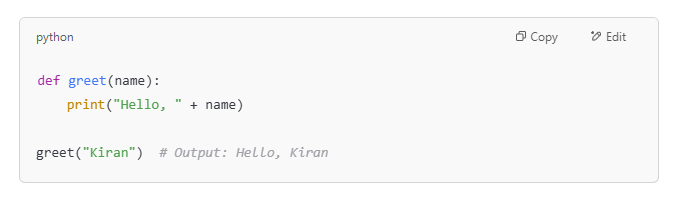
b = 42

print(id(a)) # Same as 'b'

**Functions in Python**

**Creating Custom Functions**

* Functions allow code reuse.
* **Example**:



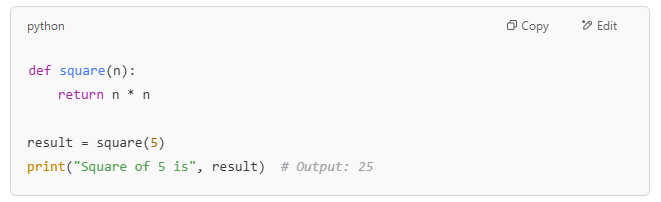
**Memory Layout & id() in Functions**

* Each function has its own scope and memory.
* **Example**:



**Function Calls**

* Functions are reusable blocks that perform tasks.
* **Example with return value**:



**Generators in Python**

**Generator Functions**

* Use yield to generate values one at a time, pausing the function’s state.
* **Example**:



**yield vs return**

* **return**: Exits the function and returns a value.
* **yield**: Pauses the function and saves its state.

**Example**:



**Conclusion**

* **Python** is a versatile and easy-to-learn language.
* **Algorithms** and functions are key to efficient programming.
* **Generators** help with memory efficiency in large datasets.