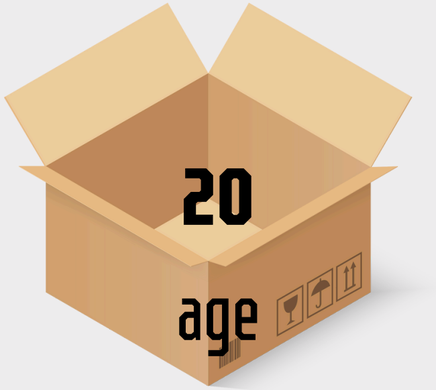
INTRO TO PYTHON PROGRAMMING

WHAT IS PYTHON ?

- Python is a high-level programming language — it's designed to be easy to read and write, with simple syntax that feels close to human language.
- Why Python?- Used in web dev, data science, AI/ML, automation, game dev, and more.
- Huge ecosystem - Rich libraries (NumPy, Pandas, PyTorch, Flask, etc.) support almost any use case.

VARIABLES IN PYTHON

A variable is like a box where you can store information. You give the box a name, and you can put things like numbers or words inside.



In Python, you just write the name and use an equal sign to store something in it.

For example , age = 20

OPERATORS IN PYTHON

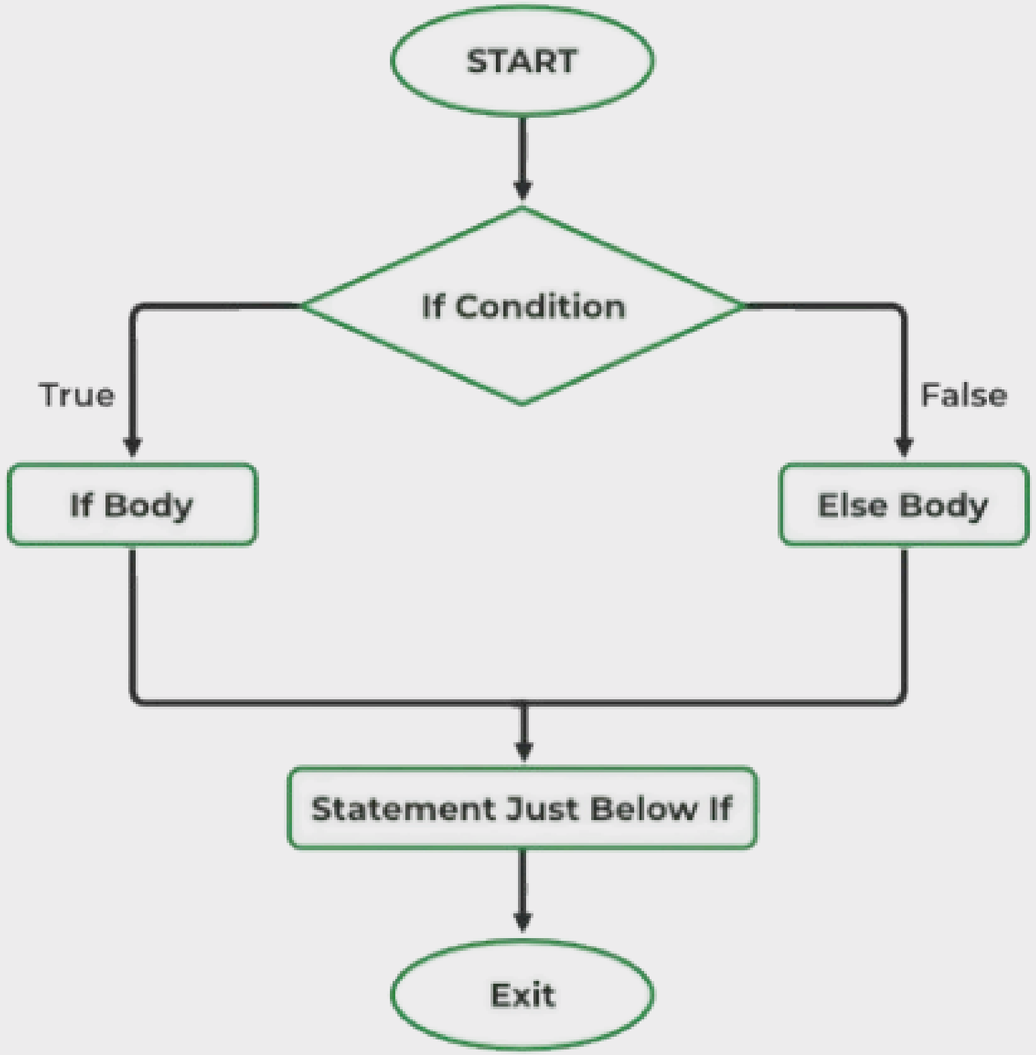
There are several types of Operators in Python.

Arithmetic Operators: Perform basic math operations like addition, subtraction, multiplication, division, and modulo.

Logical Operators: Use AND, OR, and NOT to combine or invert conditions.

Comparison Operators: Compare values to check equality, inequality, greater than, or less than.

CONDITIONAL STATEMENTS IN PYTHON



Data Structures in Python

There are 4 main Built-in Data Structures in python :

Lists

Lists are mutable built-in data type that stores a set of values - integer, float,string.

Example :

marks = [87,64,33,95]

Student = [“Karan”, 85, “Delhi”]

Tuples

Tuples are immutable built-in data type that let us create a sequence of values.

Example :

tup = (87,64,33,95)

Sets

A set is a mutable collection of unique elements.

Example:

fruits = {"apple", "banana", "cherry", "apple"}

Student = {”Karan”, 85, “Delhi”}

Dictionary

A dictionary is a collection of key-value pairs. Each key is unique and immutable and can have any value (int/string/float/boolean)

Example:

```
Student = {
"name": "Karan",
"age": 18,
"course": “Python"
“exam”: True
}
```

Lists

List Methods

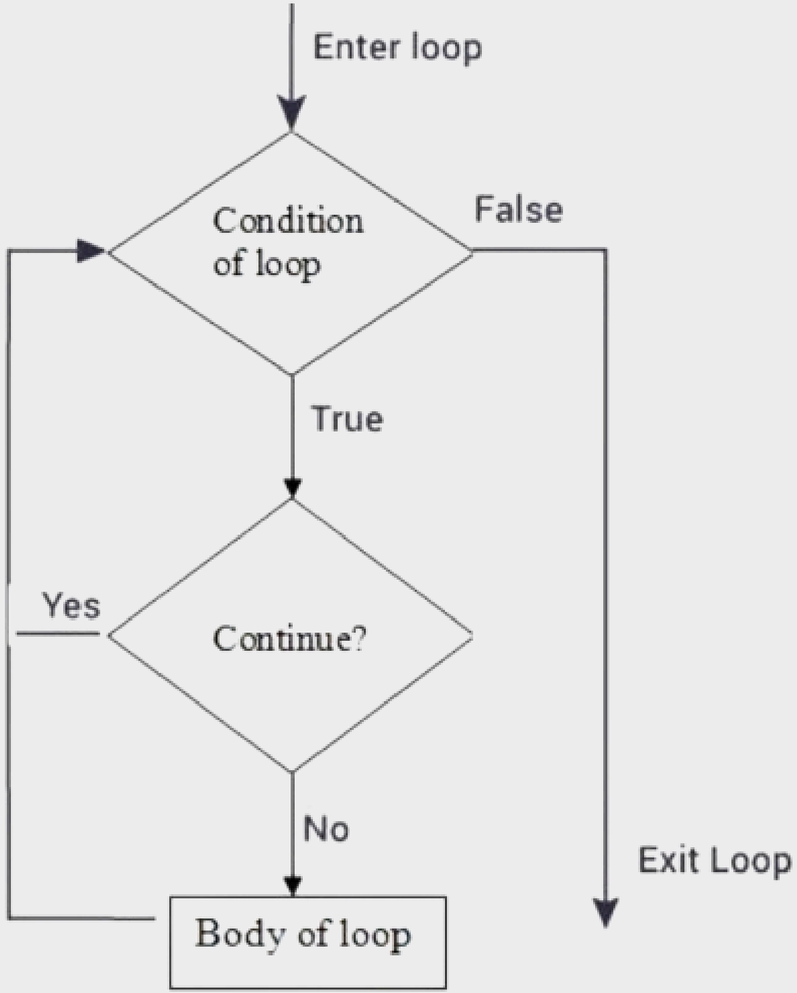
- list.append(4) #adds one element at the end
  example : list = [2, 1, 3] → [2,1,3,4]
- list.sort( ) #sorts in ascending order → [1,2,3]
- list.sort(reverse=True) #sorts in descending order → [3,2,1]
- list.reverse( ) #reverses list → [3,1,2]
- list.insert(idx, el) #insert element at index
  example : list.insert(5,2) → [2,1,5,3]
- list.remove(1) #removes first occurrence of element
- list.pop(idx) #removes element at idx

The indexing method is the same for Lists, tuples and sets

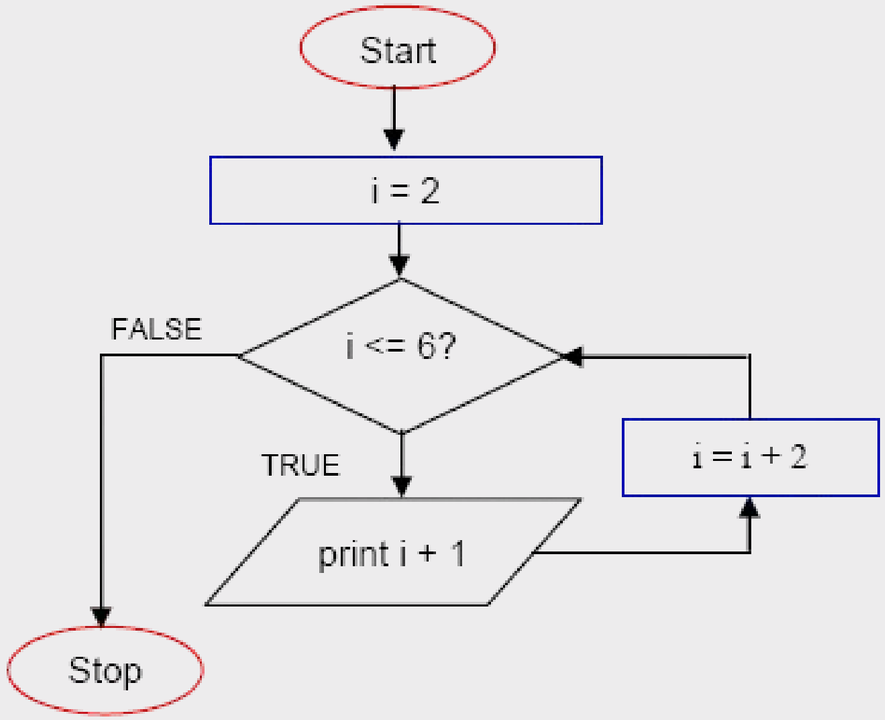
LOOPS IN PYTHON

- A loop is used to repeat a block of code multiple times.
- Helps avoid writing the same code again and again.
- Two main types:
    a. For Loop → iterates over a sequence (list, string, tuple, range, etc.).
    b. While Loop → keeps running until a condition becomes False.

While Loop



For Loop



INTRO TO OOPS (OBJECT ORIENTED PROGRAMMING

Why does OOP Exist?

- Problem: Managing complex real-world data (like multiple cars) with only separate variables (e.g., car1_color, car2_speed) gets messy and unscalable.
- Solution: OOP groups related data (Attributes) and actions (Methods) into single, logical units called Objects. This creates highly organized and reusable code.

Class vs Object

Concept	Definition	Analogy	Python Code
Class	The Blueprint or template. Defines what an object will be	The design of a house.	class Car()
Object/ Instance	The actual Instance created from the class. Has real data.	The actual house built.	my_car = Car() Export to Sheets

Attributes (Data)

- Attributes are the characteristics or data stored by an object.
- Example: A Car object has color and speed.
- The special _init_ method is used to set an object's starting attributes when it's created.

init and self

Term	Role	Intuition
init	The Constructor method. Runs automatically when an object is created to initialize its attributes.	The factory process that fills in the car's initial specs (color, model).
self	Refers to the specific object being worked on. It must be the first parameter in methods.	How the object refers to itself internally (e.g., "set my color to red").

Export to Sheets

Methods (Actions)

- Methods are functions defined inside a class that represent the actions or abilities of the object. They are the verbs.
- Methods use self to access the object's own attributes.

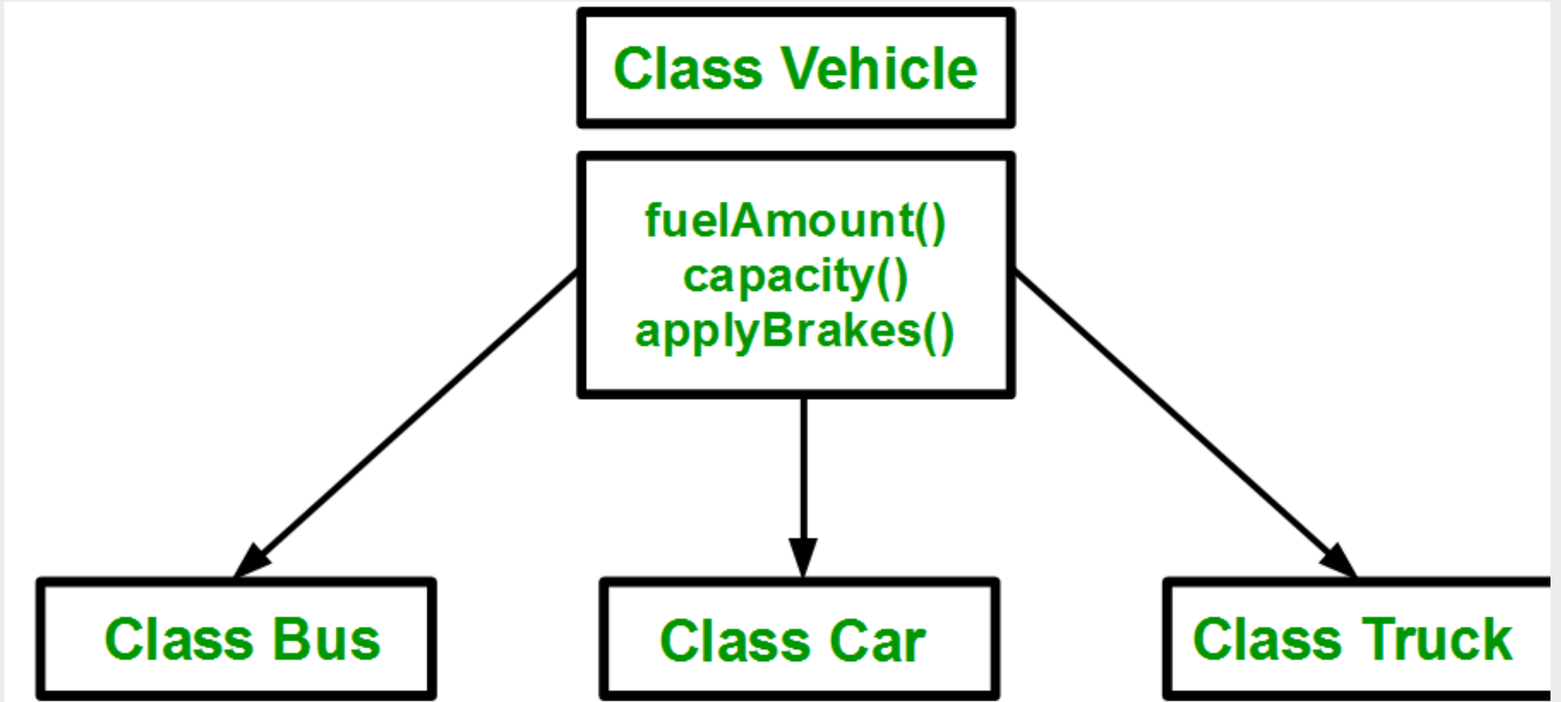
Functions vs Methods

Feature	Function	Method
Independence	Independent tool.	Bound tool. Belongs to an object.
Location	Defined outside a class.	Defined inside a class.
First Parameter	Does not use self.	Must use self.
Call Example	add(2, 3)	my_math.add(2, 3)

Export to Sheets

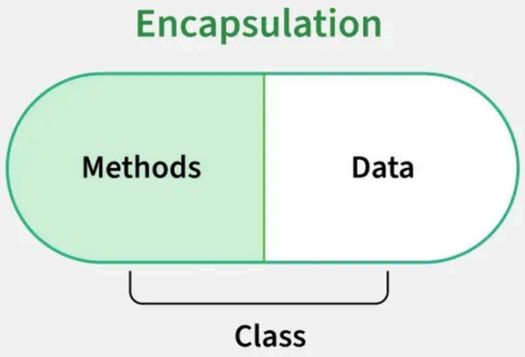
What is Inheritance?

- Inheritance is an object-oriented programming (OOP) concept where one class (called the child or subclass) inherits attributes and methods from another class (the parent or superclass).
- It allows a new class to reuse, extend, or modify the behavior of an existing class without rewriting its code.



What is Encapsulation?

- Encapsulation is the OOP principle of bundling data (attributes) and methods (functions) that operate on that data into a single class, while restricting direct access to some of the class's internal data to protect it from unintended changes.
- Example: Bank Account / ATM System
- Private data: Account balance, PIN number, personal info
- Public methods: deposit(), withdraw(), check_balance()
- Users cannot directly modify the account balance - it's hidden.
- Users can only interact through controlled methods that enforce rules (like checking PIN, ensuring sufficient funds).



What is Polymorphism?

- Polymorphism allows the same method to be used differently depending on the object.
- It allows the same method name to be used for different classes at once.
- Makes code flexible, reusable, and easier to maintain.

POLYMORPHISM

Class Dog()

Class Cat()

Class Cow()

sound()

woof!

meow!

moo!

Lets look at a small application of OOPs in AI!

What are dunder methods?

- Dunder methods are special methods in Python whose names start and end with double underscores. They let you define or customize the behavior of built-in Python operations for your objects.

- Common Uses:
- Object Constructor→ _init_
- String representation → _str_
- Arithmetic operations → _add_, _sub_ | a + b → a._add_(b)
- Comparison → _eq_, _lt_, _gt_
- Container behavior → _len_, _getitem_, _setitem_

Lets try to implement a vector class in Python

THANK YOU