**Programming:** It is the process of writing instructions (called code) that a computer can understand and follow to perform tasks.

**Why python?**

Python code is easy to write and read, making it beginner-friendly.

Data Science & AI (Pandas, NumPy, TensorFlow)

**Import**

**Import** means bringing ready-made tools (called modules or packages) into your Python program so you can use them.

**Example:**

import math

print(math.sqrt(16)) # Gives 4.0

**Packages& its importance**

A package is a folder that contains many Python files (called modules) that work together.

You use packages to organize code and reuse useful tools.

Example: pandas is a package used to work with data tables.  
You can import and use it like this: import pandas as pd

**Importance of packages:**

* Keep code neat and organized
* Save time by using existing code
* Make large projects easier to manage

**Flow charts, shapes, significance, and description**

A flowchart is a picture that shows the steps of a program or process, using different shapes and arrows.

* oval: start / end – shows where the program starts/ends
* parallelogram: input / output- shows input from user or program output
* rectangle: calculations or process- shows an action or step
* diamond: selection structures, Decisions- shows a yes/no, if/else choices

**Conditions**

make decisions — like “if this is true, do something.”

age = 18

if age >= 18:

print ("You are an adult")

else:

print ("You are a minor")

if - checks a condition

else - Runs if all above conditions are false

elif - (else if) checks another condition if the first is false

**Operations**: Operations are symbols used to do math or compare values in Python**.**

**Types of Operations:**

**1. Arithmetic Operators (for math)**

| Symbol | Use | Example | Output |
| --- | --- | --- | --- |
| + | Add | 2 + 3 | 5 |
| - | Subtract | 5 - 2 | 3 |
| \* | Multiply | 4 \* 2 | 8 |
| / | Divide | 10 / 2 | 5.0 |
| // | Floor divide | 9 // 2 | 4 |
| % | Remainder | 9 % 2 | 1 |
| \*\* | Power | 2 \*\* 3 | 8 |

**2. Comparison Operators (for checking)**

| Symbol | Meaning | Example | Output |
| --- | --- | --- | --- |
| == | Equal | 5 == 5 | True |
| != | Not Equal | 5 != 3 | True |
| > | Greater Than | 5 > 3 | True |
| < | Less Than | 3 < 5 | True |
| >= | Greater or Equal | 5 >= 5 | True |
| <= | Less or Equal | 4 <= 6 | True |

**3. Logical Operators (for combining conditions)**

| Operator | Meaning | Example | Output |
| --- | --- | --- | --- |
| and | Both must be true | True and True | True |
| or | At least one is true | True or False | True |
| not | Reverse the condition | not True | False |

**Loops**

A loop repeats a task again and again until a condition is met.

**For Loop** – Use it when you know how many times to repeat.

for i in range(5):

print(i)

output: 0 1 2 3 4

**While Loop** – Use it when you don't know how many times, but stop when a condition is false.

x = 1

while x <= 3:

print(x)

x += 1

output – 1 2 3

**break** – stop the loop immediately.

for i in range(5):

if i == 3:

break

print(i)

output – 0 1 2

**continue** – skip one time and move to the next.

**Functions**

A function is a block of code that runs only when you call it.  
It helps you reuse code.

Example: def greet(name):

print("Hello", name)

greet("Naveen”) # Output: Hello Naveen

**Parameters and Arguments**

A **parameter** is a **placeholder** in a function.

An **argument** is the **actual value** you pass when calling the function.

Example: def greet(name): # "name" is a parameter

print("Hello", name)

greet("Naveen”) # "Naveen" is an argument

**Different types of arguments**

**Positional Arguments:** Values are passed in the same order as parameters.

def greet(name, age):

print(name, age)

greet("Naveen", 25)

**Keyword Arguments**

You name the arguments, so order doesn’t matter.

greet(age=25, name="Naveen")

**Default Arguments**

You give a default value. If no argument is passed, it uses that value.

def greet(name="Friend"):

print("Hello", name)

greet() # Hello Friend

greet("Naveen") # Hello Naveen

**Variable-Length Arguments**

Use \*args (for many values) and \*\*kwargs (for many keyword values).

def add(\*numbers):

print(sum(numbers))

add(1, 2, 3) # Output: 6

def profile(\*\*info):

for key, value in info.items():

print(key, ":", value)

profile(name="Naveen", age=25, country="India")

Output:

name : Naveen

age : 25

country : India

**Data structures**

Data structures are ways to store and organize data

| **Data Structure** | **What It Stores** | **Example** |
| --- | --- | --- |
| **List** | Ordered items | ["apple", "banana", "cherry"] |
| **Tuple** | Ordered, unchangeable items | ("red", "green", "blue") |
| **Set** | Unordered, unique items | {"a", "b", "c"} |
| **Dictionary** | Key-value pairs | {"name": "Naveen", "age": 25} |
| **List:**  fruits = ["apple", "banana", "cherry"]  print(fruits[0]) # apple  **Tuple:**  colors = ("red", "green", "blue")  print(colors[1]) # green  **Set:**  nums = {1, 2, 3, 2}  print(nums) # {1, 2, 3} — removes duplicates  **Dictionary:**  student = {"name": "Naveen", "age": 25}  print(student["name"]) # Naveen |  |  |

**Key values:**

Key = the name (or label)

Value = the information (or data)

**Dictionary**: A **dictionary** stores **data in pairs**:

* Each key has a value.
* Like a real dictionary:  
  "word" → "meaning"

# Creating a dictionary

car = {

"brand": "Toyota",

"model": "Camry",

"year": 2020

}

# Accessing values

print("Brand:", car["brand"]) # Output: Toyota

print("Model:", car["model"]) # Output: Camry

print("Year:", car["year"]) # Output: 2020

# Changing a value

car["year"] = 2023

# Adding a new key-value pair

car["color"] = "Red"

# Deleting a key-value pair

del car["model"]

# Looping through the dictionary

print("\nCar Information:")

for key, value in car.items():

print(key, ":", value)

**output:** Brand: Toyota Model: Camry Year: 2020

Car Information: brand : Toyota year : 2023 color : Red

**JSON**: **JavaScript Object Notation**.  
It is a lightweight format used to store and exchange data (often between servers, APIs, and applications).

**List of datatypes and properties**

| **Data Type** | **Example** | **What It Stores** | **Notes** |
| --- | --- | --- | --- |
| int | 5, 100, -20 | Whole numbers | No decimal points |
| float | 3.14, 0.99, -10.5 | Decimal numbers | Has a dot (.) |
| str | "Naveen", 'hello' | Text (string of characters) | Written in quotes |
| bool | True, False | Yes/No values | Useful in conditions |
| list | [1, 2, 3], ["a", "b"] | Ordered, changeable items | Allows duplicates |
| tuple | (1, 2, 3) | Ordered, unchangeable items | Faster, safer than list |
| set | {1, 2, 3} | Unordered, unique items | Removes duplicates automatically |
| dict | {"name": "Naveen"} | Key-value pairs | Like a real dictionary |
| NoneType | None | Represents no value | Used when nothing is assigned yet |