## Capítulo 1: Introdução – Dia 1 - 25/08/2025

**1 - Instalação do Python**

* Acesse o site oficial: https://www.python.org/downloads
* Baixe a versão mais recente do Python para seu sistema operacional.
* Durante a instalação, marque a opção "Add Python to PATH".
* Conclua a instalação clicando em "Install Now".

**2 - Uso do ambiente de desenvolvimento IDLE**

* O IDLE é instalado junto com o Python.
* Para abrir:
  + Windows: Menu Iniciar → "IDLE (Python X.X)"
  + Linux/Mac: idle no terminal.
* Ele possui:
  + Um console interativo (Python Shell)
  + Um editor de scripts (File → New File)

**3 - Instalação e uso do PyCharm**

* Acesse https://www.jetbrains.com/pycharm/download
* Baixe a versão Community (gratuita).
* Instale e abra o PyCharm.
* Para criar um projeto:
  + File → New Project → Escolha a pasta e configure o interpretador Python.
* Para criar um script:
  + Clique com o botão direito na pasta → New → Python File.

**4 - Testes**

## Dia 2 – 25/08/2025

**Capítulo 2: Classes e Objetos**

**5 - Armazenamento de dados em programas**

Em programação de computadores uma variável é um elemento da linguagem que ocupa um ou mais bytes na memória do computador. Esse local da memória é capaz de reter, ou seja, armazenar um valor. No programa, a variável é identificada por um nome ou identificador. Desta forma podemos entender que "do ponto de vista" do programador a variável é um nome que contém um valor; e "do ponto de vista" do computador a variável é um endereço de memória que retém um conjunto de bits que representam esse valor.

* Variáveis armazenam dados como texto, números ou listas.
* Exemplo:

**Exemplo:**

qtde = 2

puni = 36.90

ptot = qtde \* puni

msg = 'Total ='

print(msg, ptot)

Total = 73.8

**6 - Classes e objetos em Python**

• **Classe** é um modelo que define como um objeto é construído. A classe é codificada no programa e fica à disposição do programador para a ser usada no momento em que for necessário criar um objeto;

• **Objeto** é construído a partir de uma classe. Ele é real, ocupa memória e consome tempo de processamento do computador;

**Exemplo:**

class Pessoa:

def \_\_init\_\_(self, nome):

self.nome = nome

p1 = Pessoa("João")

print(p1.nome)

**7 - Objetos de classes**

* Objetos podem ter atributos (dados) e métodos (funções).

**Objetos de classe simples:**

| ****Classe (Tipo)**** | ****Descrição**** |
| --- | --- |
| int | Números inteiros, positivos, negativos e zero. Ex: -3, 0, 42. |
| float | Números reais com parte decimal separada por ponto. Ex: 3.14, -0.5. |
| complex | Números complexos com parte real e imaginária (usa-se j para parte imaginária). Ex: 3 + 2j. |
| bool | Valores booleanos: True (verdadeiro) ou False (falso), com inicial maiúscula. |
| None | Representa ausência de valor. Usado para indicar que uma variável ou objeto não tem valor atribuído. Pode ser comparado usando is None. |

### >>> i = 12

>>> type(i)

<class 'int'>

>>> f = 7.38

>>> type(f)

<class 'float'>

>>> c = 1.57 + 3.17j

>>> type(c)

<class 'complex'>

>>> b = True

>>> type(b)

<class 'bool'>

>>> d = False

>>> type(d)

<class 'bool'>

>>> x = None

>>> type(x)

<class 'NoneType'>

### ✅ Observações:

* int, float e complex são **tipos numéricos**.
* bool é utilizado em **operações lógicas e controle de fluxo**.
* None é um **objeto especial** que indica **ausência de valor**.

**Objetos de classe composto:**

| ****Classe**** | ****Descrição**** |
| --- | --- |
| str | Armazena **texto** (cadeia de caracteres). Pode usar aspas simples ' ou duplas ". Ex: 'Olá', "Texto!!" |
| list | **Lista** de elementos ordenados e mutáveis. Pode conter qualquer tipo de dado. Usa colchetes [ ]. Ex: [1, 2, 3] |
| tuple | Semelhante à lista, mas **imutável** (somente leitura). Usa parênteses ( ). Ex: (1, 2, 3) |
| range | Gera uma **sequência numérica** (usado com for). Pode receber 1, 2 ou 3 parâmetros: start, stop, step. Ex: range(1, 10, 2) |
| dict | **Dicionário**: coleção de pares chave: valor. Usa chaves { }. Ex: {'maçã': 9.75} |
| set | **Conjunto** de elementos únicos e não ordenados. Usa chaves { }. Ex: {1, 2, 3} |
| frozenset | Igual ao set, mas **imutável** (não permite adicionar ou remover elementos). |
| bytes | Armazena **dados binários imutáveis**. Usado para manipulação de arquivos, som, imagens, rede etc. |
| bytearray | Versão **mutável** do bytes. |
| memoryview | Permite **acesso eficiente à memória de objetos binários**, como bytes e bytearray. |

### 📌 Diferença entre set e dict

| ****Estrutura**** | ****Exemplo**** | ****Descrição**** |
| --- | --- | --- |
| set | {1, 2, 3} | Conjunto sem repetição, sem chave. |
| dict | {'chave': 'valor'} | Par chave-valor, como um dicionário mesmo. |

### ✅ Extras:

* str, list, tuple são **sequências**.
* dict é uma **estrutura de mapeamento**.
* set e frozenset seguem a **teoria de conjuntos**.
* bytes, bytearray e memoryview são usados com **dados binários**.

**Exemplo:**

class Carro:

def \_\_init\_\_(self, modelo, cor):

self.modelo = modelo

self.cor = cor

def buzinar(self):

print("Biiiii!")

meu\_carro = Carro("Fusca", "Azul")

meu\_carro.buzinar()

### ****8 - Comando de Atribuição****

* **Atribuição** em Python é feita com o operador =.
* Permite **associar um valor a uma variável**.

**Exemplo:**

x = 10

nome = "Python"

* Pode-se fazer **atribuições múltiplas**:

a, b = 5, 7

* Python também aceita **atribuição encadeada**:

x = y = z = 0

## ****Dia 3 – 23/08/2025****

### ****9 - Id dos Objetos Python (Parte 1)****

* Em Python, **tudo é objeto**.
* Cada objeto possui um **identificador único** (ID), acessado com id(objeto).
* O id representa a **posição na memória** onde o objeto está armazenado (de forma abstrata).

**Exemplo:**

a = 5

b = a

print(id(a), id(b)) # Mesmo ID → apontam para o mesmo objeto

### ****10 - Id dos Objetos Python (Parte 2)****

* Em Python, **cada objeto tem um identificador único na memória**.
* Esse identificador pode ser visto com a função id(objeto).
* É como o “endereço de memória” do objeto.

### ✅ a****. Objetos Imutáveis:**** int****,**** str****,**** tuple

Esses objetos **não podem ser alterados após a criação**.

#### 🔁 Reutilização de memória:

Python é inteligente e **reaproveita** o mesmo objeto para economizar memória.

Veja um exemplo:

a = 10

b = 10

print(id(a), id(b)) # Mesmo id!

print(a is b) # True

Mesmo que a e b pareçam variáveis diferentes, elas apontam para o **mesmo objeto** na memória (se o valor for igual e for imutável).

Agora veja com str:

x = "python"

y = "python"

print(id(x), id(y)) # Mesmo id!

print(x is y)

### ✅ b****. Objetos Mutáveis: list, dict, etc****

Esses objetos **podem ser alterados após a criação**.

#### 🔁 ID muda ao criar um novo objeto, mas não ao modificar:

lista1 = [1, 2, 3]

lista2 = lista1

lista1.append(4)

print(lista1) # [1, 2, 3, 4]

print(lista2) # [1, 2, 3, 4]

print(id(lista1), id(lista2)) # Mesmo id!

**Ou seja: modificar não muda o ID, mas se você criar uma nova lista, aí sim:**

lista1 = [1, 2, 3]

print(id(lista1)) # Exemplo: 140034839520256

lista1 = lista1 + [4] # Cria nova lista!

print(id(lista1)) # ID diferente

⚠️ A operação lista1 + [4] **cria um novo objeto**, então o id muda.

### ✅ ****Resumo prático:****

| Tipo de Objeto | Pode ser alterado? | Mesmo ID após modificação? | Exemplo |
| --- | --- | --- | --- |
| int, str, tuple | ❌ Imutável | ❌ Novo objeto (novo ID) | x = x + 1 cria novo int |
| list, dict, set | ✅ Mutável | ✅ Mesmo ID ao modificar | lista.append(1) mantém ID |

### ****11 - Modelo de Dados de Python****

* Python adota um modelo orientado a objetos: **tudo é objeto**.
* Cada objeto tem:
  + **Tipo** (type(objeto))
  + **Valor**
  + **Identidade** (id(objeto))
* O comportamento dos objetos é definido por **métodos especiais** (ex: \_\_add\_\_, \_\_len\_\_, etc.).
* Esse modelo é a base do **funcionamento interno do Python**.

**>>> L = [44, 17, 26, 35, 20]**

**>>> type(L)**

**<class 'list'>**

**>>> id(L)**

**140450933453824**

**>>> len(L)**

**5**

**>>> print(L[0])**

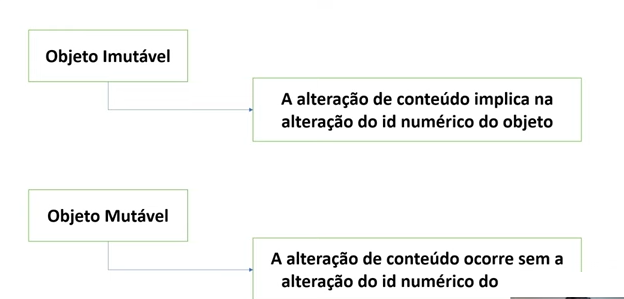
**44**

**>>> L[0] = 12**

**>>> id(L)**

**140450933453824**

**>>>**



### 

### ****12 - Operadores Aritméticos****

| Operador | Descrição | Exemplo | Resultado |
| --- | --- | --- | --- |
| + | Adição | 2 + 3 | 5 |
| - | Subtração | 5 - 2 | 3 |
| \* | Multiplicação | 4 \* 2 | 8 |
| / | Divisão real | 5 / 2 | 2.5 |
| // | Divisão inteira | 5 // 2 | 2 |
| % | Módulo (resto) | 5 % 2 | 1 |
| \*\* | Potência | 2 \*\* 3 | 8 |