**Aula 1 de Python – Introdução**

**História do Python**

O Python foi criado por Guido van Rossum no final dos anos 1980 e lançado oficialmente em 1991. Ele buscava um projeto de hobby que fosse fácil de entender e usasse uma sintaxe clara e direta. A ideia era criar uma linguagem tão poderosa quanto C, mas muito mais simples de escrever e ler. O nome "Python" não tem relação com a cobra, mas sim com o grupo de comédia britânico "Monty Python’s Flying Circus", do qual Guido era fã.

Desde então, o Python se tornou uma das linguagens mais populares do mundo, muito usada em áreas como ciência de dados, inteligência artificial, automação, desenvolvimento web e muito mais. A linguagem é mantida pela Python Software Foundation (PSF) e possui uma grande comunidade ativa de desenvolvedores.

**Características do Python**

Python é uma linguagem de programação interpretada, de alto nível e multiparadigma, permitindo tanto programação orientada a objetos quanto programação funcional e imperativa. Uma das suas maiores vantagens é a sintaxe simples e legível, que permite escrever menos código para fazer mais coisas. Python também é multiplataforma, ou seja, pode ser executado em Windows, Linux e macOS sem alterações no código.

Outras características marcantes incluem o gerenciamento automático de memória, uma vasta biblioteca padrão e a capacidade de ser estendida com módulos escritos em outras linguagens, como C e C++. Além disso, Python favorece a legibilidade do código através da indentação obrigatória.

**Instalação do Interpretador**

Para programar em Python, é necessário instalar o interpretador, que é o software que vai ler e executar o código. A instalação oficial é feita através do site [python.org](https://www.python.org/). No Windows, é importante marcar a opção "Add Python to PATH" durante a instalação. Em sistemas Linux e macOS, muitas vezes o Python já vem instalado, mas pode ser atualizado usando o terminal.

Após a instalação, é possível verificar se tudo deu certo digitando python --version ou python3 --version no terminal ou prompt de comando. Isso mostrará a versão instalada do Python.

**Introdução ao Shell do Python**

O Shell do Python é um ambiente interativo que permite testar comandos linha por linha. Após instalar o Python, basta abrir o terminal e digitar python para entrar no Shell. No Shell, é possível fazer operações matemáticas, criar variáveis e testar funções rapidamente, sem a necessidade de salvar arquivos.

O Shell é útil para pequenos testes, mas para programas maiores é melhor usar editores de código ou IDEs, onde podemos salvar, organizar e executar nossos scripts.

**Introdução ao PyCharm e VSCode**

O PyCharm e o Visual Studio Code (VSCode) são duas ferramentas populares para programar em Python. O PyCharm é uma IDE focada em Python, muito poderosa, com recursos como autocompletar, depuração e integração com sistemas de controle de versão. Existe uma versão gratuita (Community) que já é bem completa.

O VSCode é um editor de código mais leve e mais genérico, mas com extensões pode se transformar em um ambiente excelente para programar em Python. Uma vantagem do VSCode é que ele é mais rápido de instalar e ocupa menos espaço, além de suportar várias outras linguagens no mesmo editor.

**Variáveis**

Variáveis em Python são usadas para armazenar dados. Elas são criadas simplesmente atribuindo um valor a um nome. Não é necessário declarar o tipo da variável, pois o Python é uma linguagem de tipagem dinâmica, ou seja, o tipo é inferido automaticamente com base no valor atribuído.

nome = "Maria"

idade = 25

Nesse exemplo, nome é uma variável do tipo string e idade é uma variável do tipo inteiro.

**Tipos de Variáveis**

Os principais tipos de variáveis que veremos são:

* **int**: números inteiros, como 10, -5
* **float**: números decimais, como 3.14, -0.01
* **str**: textos (strings), como "Python", "123"
* **bool**: valores lógicos True ou False
* **list**: coleção de valores mutáveis, ex: [1, 2, 3]
* **tuple**: coleção de valores imutáveis, ex: (1, 2, 3)
* **dict**: conjunto de pares chave-valor, ex: {"nome": "Ana", "idade": 20}

Cada tipo de variável tem suas próprias características e usos específicos dentro dos programas.

**Regras para Declaração de Variáveis no Python**

Existem algumas regras que devemos seguir ao criar variáveis:

* O nome deve começar com uma letra ou underline \_.
* Não pode começar com números.
* Pode conter letras, números e underlines.
* Não pode ter espaços.
* Não pode ser uma palavra reservada do Python (como if, while, for, etc).

Boas práticas também recomendam usar nomes de variáveis que sejam descritivos e fáceis de entender.

Exemplos válidos:

nome\_completo = "João Silva"

\_idade = 30

salario1 = 2500.50

Exemplos inválidos:

1nome = "Maria"     # começa com número

nome completo = "José"  # contém espaço

for = "laço"        # palavra reservada

**Entrada e Saída de Dados (print e input)**

A função print() é usada para exibir mensagens no console. Já a função input() é usada para capturar informações digitadas pelo usuário.

Exemplo de print():

print("Bem-vindo ao curso de Python!")

Exemplo de input():

nome = input("Digite seu nome: ")

print("Olá,", nome)

Vale lembrar que o input() sempre retorna uma string. Se quiser trabalhar com números, é necessário converter o valor.

**Conversão de Tipo de Dados**

Como o input() sempre retorna string, precisamos converter para o tipo desejado usando as funções de casting:

* int() para inteiro
* float() para decimal
* str() para string
* bool() para booleano (atenção: qualquer string não vazia vira True)

Exemplo de conversão:

idade = int(input("Digite sua idade: "))

altura = float(input("Digite sua altura: "))

print("Você tem", idade, "anos e", altura, "metros de altura.")

**Formas Diferentes de Fazer o Print**

Existem várias maneiras de formatar a saída no Python:

**1. Usando o sinal de +** (atenção: só funciona com strings):

nome = "Carlos"

print("Olá " + nome + "!")

**2. Usando o f-string** (Python 3.6+):

nome = "Carlos"

idade = 28

print(f"Olá {nome}, você tem {idade} anos.")

**3. Usando format()**:

nome = "Carlos"

idade = 28

print("Olá {}, você tem {} anos.".format(nome, idade))

O método format() permite uma formatação mais dinâmica, especialmente útil em versões antigas do Python.

**Atividades de Receber Dados e Mostrar na Tela**

**Atividade 1  
➡️ Peça para o usuário digitar seu nome e exiba uma mensagem de boas-vindas usando print.**

**Atividade 2  
➡️ Solicite a idade do usuário, converta o valor para inteiro e mostre na tela uma frase dizendo quantos anos ele tem.**

**Atividade 3  
➡️ Peça dois números decimais para o usuário, calcule a soma deles e exiba o resultado usando f-string.**

**Atividade 4  
➡️ Solicite o nome de um produto e seu preço. Depois, mostre as informações organizadas na tela (produto e preço).**

**Continuação: Operadores e Expressões em Python**

**➡️ Operadores Aritméticos**

Os operadores aritméticos são usados para realizar operações matemáticas básicas entre valores numéricos. No Python, temos: adição (+), subtração (-), multiplicação (\*), divisão (/), divisão inteira (//), resto da divisão (%) e potência (\*\*). Eles funcionam tanto com números inteiros quanto com números de ponto flutuante.

Esses operadores podem ser combinados em expressões, e o Python segue a ordem matemática normal para resolver.  
**Exemplo:**

a = 10

b = 3

print(a + b)   # 13

print(a - b)   # 7

print(a \* b)   # 30

print(a / b)   # 3.333...

print(a // b)  # 3 (divisão inteira)

print(a % b)   # 1 (resto)

print(a \*\* b)  # 1000 (potência)

**➡️ Operadores de Atribuição**

Operadores de atribuição servem para guardar um valor dentro de uma variável, ou atualizar seu valor com base em operações. O operador principal é o =, mas também temos formas abreviadas, como +=, -=, \*=, /=, //=, %=, \*\*=, que combinam uma operação aritmética com a atribuição.

Esses operadores ajudam a deixar o código mais curto e claro.  
**Exemplo:**

x = 5

x += 3   # equivale a x = x + 3 → x agora é 8

x \*= 2   # equivale a x = x \* 2 → x agora é 16

**➡️ Operadores de Comparação**

Operadores de comparação servem para comparar dois valores e retornar um resultado **booleano** (True ou False). São eles: igual (==), diferente (!=), maior (>), menor (<), maior ou igual (>=), menor ou igual (<=).

Esses operadores são muito usados em condições, como em estruturas de decisão (if).  
**Exemplo:**

a = 5

b = 8

print(a == b)  # False

print(a != b)  # True

print(a > b)   # False

print(a <= b)  # True

**➡️ Operadores Lógicos**

Os operadores lógicos permitem combinar várias comparações. Em Python temos: and (e), or (ou), not (não). Eles retornam True ou False baseados na lógica das expressões.

Esses operadores são muito usados para testar múltiplas condições ao mesmo tempo.  
**Exemplo:**

idade = 20

tem\_carteira = True

print(idade >= 18 and tem\_carteira)   # True (tem 18+ e carteira)

nome = "Ana"

print(nome == "Ana" or nome == "Maria")  # True

print(not idade < 18)  # True (nega o resultado de idade < 18)

**➡️ Ordem de Precedência**

Quando várias operações aparecem em uma mesma expressão, o Python segue uma **ordem de precedência** para decidir o que calcular primeiro. A ordem principal é:

1. Parênteses ()
2. Potência \*\*
3. Multiplicação \*, Divisão /, Divisão inteira //, Resto %
4. Adição + e Subtração -
5. Operadores de comparação (==, !=, >, <, etc)
6. Operadores lógicos (not, and, or)

Essa ordem é parecida com a matemática tradicional. Você pode sempre usar parênteses para controlar a ordem de avaliação.  
**Exemplo:**

resultado = 3 + 5 \* 2  # resultado será 13 (multiplicação vem antes da soma)

resultado = (3 + 5) \* 2  # resultado será 16 (parênteses mudaram a ordem)

**Atividades**

**Atividade 5**  
➡️ Peça dois números inteiros ao usuário e mostre: a soma, a subtração, a multiplicação, a divisão, a divisão inteira, o resto da divisão e o número elevado ao outro.

**Atividade 6**  
➡️ Crie uma variável pontos começando com valor 0. Depois adicione 10 pontos e multiplique por 2. Mostre o valor final.

**Atividade 7**  
➡️ Peça dois números para o usuário e diga se o primeiro é maior, menor ou igual ao segundo.

**Atividade 8**  
➡️ Solicite ao usuário sua idade e se possui carteira de motorista (S/N). Use operadores lógicos para dizer se ele pode dirigir (idade >= 18 e resposta == 'S').

**Atividade 9**  
➡️ Peça ao usuário três números e calcule a média, mas garanta que a soma seja feita primeiro, depois dividida (use parênteses para garantir a ordem).