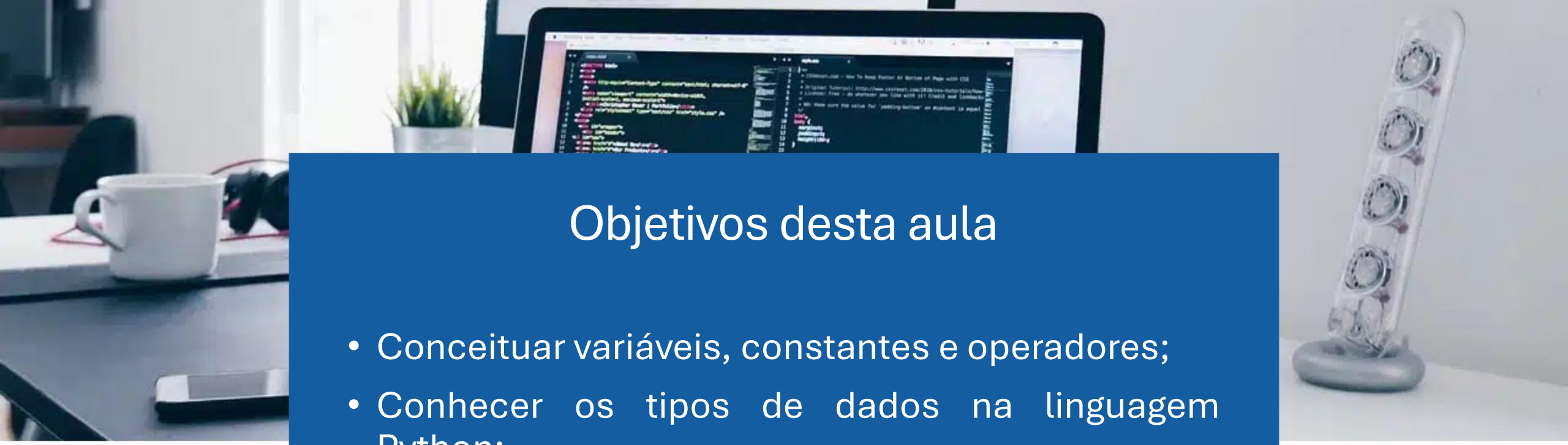
Aula #2:

Variáveis, constantes e operadores

Prof. Dr. Luiz Álvaro de Oliveira Júnior



Objetivos desta aula

- Conceituar variáveis, constantes e operadores;
- Conhecer os tipos de dados na linguagem Python;
- Entender a ordem de precedência das operações matemáticas;

Sumário

O que são variáveis?

Nas linguagens de programação, uma **variável** é uma estrutura que **armazena dados** temporariamente na memória do computador para serem usados ou modificados ao longo da execução do programa.

Essencialmente, as variáveis funcionam como se fossem **caixinhas nomeadas que guardam informações**, permitindo que o programa seja dinâmico, adaptável e interativo.

As variáveis são reconhecidas por um **identificador**, um **tipo** e um **valor**.

O que são variáveis?

Identificador: é o nome que escolhido por você para a variável (ex.: idade, altura, peso...)

Tipo: define que tipo de informação a variável guarda no programa (ex.: texto, número inteiro, número real, número complexo, booleano...)

Valor: é o dado propriamente dito atribuído à variável, ou seja, o valor que ela guarda.

```
idade = 28          #número inteiro
```

O que são variáveis?

As variáveis tornam o código flexível, pois permitem que seus valores mudem conforme o necessário, e também mais fácil de ler e compreender, uma vez que normalmente as nomeamos de forma a facilitar o entendimento de seu significado após simples leitura.

Vejamos algumas recomendações para nomear as variáveis:

Variáveis locais e globais

Variáveis podem ser locais ou globais.

As **variáveis locais** são aquelas que são criadas dentro de funções e existem localmente somente nelas, não sendo possível acessar essas variáveis externamente.

Já as **variáveis globais** são aquelas que são criadas fora das funções e podem ser usadas em qualquer parte do código. Por padrão, toda variável criada fora de uma função é global e essa definição é implícita.

Variáveis locais e globais

As funções não podem alterar valores das variáveis globais (externas a elas) exceto se tais variáveis forem listadas explicitamente na função.

Para fazer isso, é necessário usar a palavra reservada **global** antes de listar as variáveis globais que a função pretende alterar. Sem o **global**, o Python cria cópias locais dessas variáveis na função, o que pode causar erros ou comportamentos inesperados.

Recomendações para nomear variáveis

- **Não utilizar somente números ou caracteres especiais (@,#,$)**.
- Usar nomes que **descrevam claramente o propósito da variável**. Exemplos: nome, idade, total_alunos.
- **Não usar palavras reservadas** da linguagem. Exemplos: if, else, def, for, while, and...
- Em caso de **nome composto, separe com underline ou não separe**. Exemplos: total_vendas, nomeCliente.
- Evitar **nomes de uma letra**, a menos que o contexto seja muito específico, como os contadores dos loops. Exemplos: i, j, k, x, y.

Recomendações para nomear variáveis

- Use sempre a **mesma forma de escrever o nome da variável** a cada vez que ela for chamada no código, pois o Python é ***case sensitive***, isto é, diferencia caracteres maiúsculos e minúsculos. As variáveis Total e total são diferentes!
- Nome de variáveis normalmente começam com letras minúsculas
- Evite abreviações ambíguas, pois elas podem gerar dúvida sobre o significado da variável. Ex: cnt poderia significar contagem, conteúdo, contador...
- Use comentários quando o nome da variável não bastar para explicitar seu significado. Para comentar, use #.

Tipos de dados

Características

- Linguagem interpretada
- Tipagem dinâmica: não requer declaração dos tipos das variáveis.
- Tipagem forte: operações entre tipos incompatíveis não são permitidas sem conversão explícita.

Tipos

- Inteiro (int)
- Ponto flutuante (float)
- Complexo (complex)
- Caractere (string)
- Booleano (boolean)

Tipos de dados

O tipo **inteiro** (int) representa os números inteiros, ou seja, **aqueles que não têm parte decimal**, sejam eles positivos ou negativos. Zero também é um número inteiro.

```
pecas = 15
x = -2
```

O tipo **ponto flutuante** (float) representa **os números com parte decimal**, sejam eles positivos ou negativos. Um número inteiro pode ser definido como float.

```
massa = 78.53
x = -2.0
```

Tipos de dados

O tipo **complexo** (complex) é usado para representar números complexos, sendo comum seu uso em cálculos científicos, especialmente na Engenharia Elétrica. Uma variável de tipo complexo terá **duas partes**: a **real** e a **imaginária**, sendo que a imaginária contém uma letra “j” como sufixo.

```
a = -3 + 7j
b = 8 – 0j
```

Tipos de dados

O tipo **string** (caractere) representa um conjunto ou sequência de caracteres dispostos em determinada ordem, com ou sem significado, embora geralmente represente palavras, frases ou textos, mas também permite a representação de números. Representado entre aspas duplas ou simples.

```
senha = "Ab64at31@n4ge9he"
universidade = "pucgoias"
num_sapato = "41"
```

Tipos de dados

```
x = 10       #o número inteiro 10
y = 5        #o número inteiro 5
z = x + y    #a soma de um inteiro com outro inteiro
```

OK !

```
x = 10       #o número inteiro 10
y = "5"       #a string (caractere) 5
z = x + y    #a soma de um inteiro com um caractere
```

ERRO !

```
x = 10       #o número inteiro 10
y = 5.5      #número real (float) 5,5
z = x + y    #a soma de inteiro com float é um float
```

OK !

Tipos de dados

O tipo **booleano** (bool) é um tipo lógico que, na prática, assume apenas dois valores: **True** (verdadeiro) ou **False** (falso). São essenciais na tomada de decisão.

A **avaliação é implícita**, ou seja, em Python 0, None, """" (string vazia), [] (lista vazia) e {} (dicionário vazio) são automaticamente considerados False. Qualquer outro valor é automaticamente True.

Apesar de serem lógicos, os booleanos possuem equivalentes numéricos, ou seja, eles valem respectivamente 1 (True) e 0 (False).

Tipos de dados

Em Python, 0 é considerado falso (False) e qualquer número diferente de zero é considerado verdadeiro (True), **não apenas 1**. **Isso vale quando o número é avaliado em expressões condicionais ou lógicas**.

Conversão de tipos numéricos

Em Python, quando é possível, mudar o tipo de dado é algo bastante simples. Vejamos algumas funções:

str(): **converte para string** os tipos inteiro, float e complex.

int(): **converte para inteiro** os tipos string, bool e float **se a string for numérica**. Na conversão para inteiro, a parte decimal é descartada e ocorre truncamento do número para o inteiro mais próximo.

float(): **converte para float** os tipos inteiro e string **se a string for numérica**. Na conversão, ocorre acréscimo de casas decimais nulas.

Constantes

As **constantes** são semelhantes às variáveis com uma grande diferença: **seu valor não se altera durante a execução do programa**. Ela representa um dado fixo, útil para manter referências estáveis como coeficientes físicos, limites de segurança ou parâmetros padrão em engenharia, por exemplo.

Normalmente as constantes são identificadas com letras maiúsculas.

```
PI = 3.141592       #número float (real)
```

Operadores

Operadores são símbolos ou palavras-chave que instruem o computador a realizar operações sobre variáveis e valores. Eles tornam possível processar dados e tomar decisões.

Operadores

Tipo	Exemplos	Descrição
Aritméticos	`+` , `-` , `*` , `/` , `%` , `**`	Realizam operações matemáticas
Relacionais	`==` , `!=` , `>` , `<` , `>=` , `<=`	Comparam valores e retornam verdadeiro ou falso
Lógicos	`and` , `or` , `not`	Usados para lógica booleana (E, OU, NÃO)
De atribuição	`=` , `+=` , `-=` , `*=` , `/=`	Atribuem ou atualizam valores em variáveis
Bitwise (binários)	`&` , `|` , `^` , `<<` , `>>`	Manipulam bits diretamente (úteis em eletrônica)

Expressões aritméticas

Uma expressão aritmética é formada por números (operandos) e operadores. Na sua forma mais simples, uma expressão contém apenas um número, então, o valor da expressão com apenas um número é o próprio número.

Quando uma expressão contém um operador, o Python realizará a operação usando os operandos fornecidos, resolvendo um operador por vez.

O Python, naturalmente, segue as regras de precedência dos operadores que nós aprendemos na matemática.

Expressões aritméticas

Prioridade	Operador (operação)	Descrição
1	()	Parênteses – forçam a ordem desejada
2	**	Exponenciação
3	+ , -	Sinais unários (positivo/negativo)
4	* , / , // , %	Multiplicação, divisão, divisão inteira, módulo
5	+ , -	Adição e subtração
6	< , >	Comparações
7	not	Negação lógica
8	and	Conjunção lógica
9	or	Disjunção lógica

Exercícios

Exercício #1

#	Expressão	Resultado
(a)	xis = 2 + 3 -4 + 8 /2	xis = 5.0
(b)	var = -5 + 5 * 5*(1 + 4)	var = 120
(c)	valor = 30 / 2 / (3 * 5) + 2	valor = 3.0
(d)	conta = - 9 + 9 / 2 * 6	conta = 18.0
(e)	variavel = 35 – (7 * 2) + 5 ** 2 – 3	variavel = 43
(f)	paginas = 5 // 2	paginas = 2
(g)	tentativas = 5 % 2	tentativas = 1
(h)	indice = -(-3)**2 + 2**-2	indice = -8.75