

INTRODUÇÃO À EQUAÇÕES DIFERENCIAIS

Equações Diferenciais de 1ª Ordem

Profa.: Josiele da Silva Teixeira

Centro Federal de Educação Tecnológica Celso Suckow da Fonseca - CEFET
Campus UNED Nova Friburgo

October 9, 2024

1. INTRODUÇÃO A PYTHON

1.1. ORIGEM DA LINGUAGEM PYTHON

1.2. INSTALAÇÃO DO AMBIENTE PYTHON

2. EXECUÇÃO SEQUENCIAL

2.1. VARIÁVEIS E CONSTANTES

2.2. TIPOS DE DADOS

2.3. OPERADORES ARITMÉTICOS

2.4. OPERADORES DE ATRIBUIÇÃO

2.5. ENTRADA DE DADOS E CONVERSÃO DE TIPOS

3. ESTRUTURAS CONDICIONAIS

3.1. ONDE ENCONTRAMOS ESTRUTURAS CONDICIONAIS?

3.2. OPERADORES

3.3. OPERADORES DE COMPARAÇÃO OU RELACIONAIS

3.4. OPERADORES LÓGICOS

3.5. TABELA VERDADE

3.6. ESTRUTURA IF...ELSE

3.7. ESTRUTURA ELIF

Introdução a Python

Origem da Linguagem Python

- * Python é uma linguagem de programação criada pelo matemático e programador Guido Van Rossum em 1991.
- * Embora muita gente faça associação direta com cobras, por causa das serpentes píton ou pitão, a origem do nome se deve ao grupo humorístico britânico *Monty Python*.

Guido Van Rossum - Criador da linguagem Python



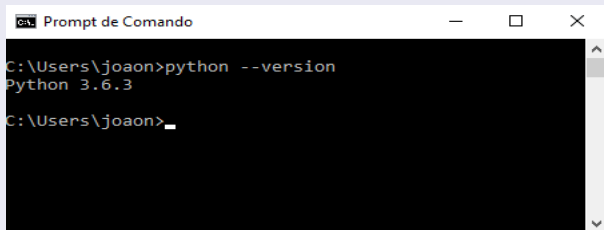
Logo oficial da linguagem Python

- * A insistência em associar Python a serpentes ajudou na criação do símbolo oficial da linguagem



Instalação do ambiente Python

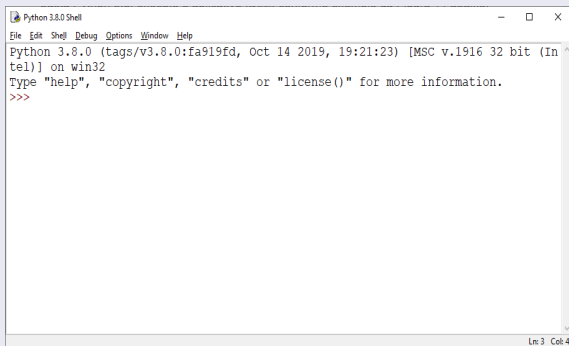
- * O site oficial é **www.python.org**
- * Para verificar se já está instalado abra a console de comandos do sistema operacional e digite o comando **python --version**



```
C:\> Prompt de Comando
C:\Users\joaon>python --version
Python 3.6.3
C:\Users\joaon>_
```

IDLE - Ambiente de desenvolvimento e aprendizado integrado padrão para Python

- * Agora, é bom testar. Vá nos programas instalados do Windows e procure a pasta Python 3.8, execute o aplicativo IDLE



```
Python 3.8.0 Shell
File Edit Shell Debug Options Window Help
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

- *IDE - Integrated Development Environment (Ambiente de Desenvolvimento Integrado): PyCharm, Jupyter Notebook, Visual Studio Code

“Hello World”

- * Após instalação da linguagem Python, veja o seu primeiro programa funcionando!!
- * Vamos escrever juntos um “Hello World”
- * Se você ainda não sabe, sempre que se inicia o aprendizado de uma nova linguagem, tradicionalmente, o primeiro programa é um “Hello World”
- * A quem diga que quem pula essa etapa não consegue avançar nos estudos de programação.
- * Por via das dúvidas, siga o conselho.

CARACTERÍSTICAS DA LINGUAGEM PYTHON

O objetivo inicial da linguagem era permitir código enxuto e menos verboso, ou seja, com menos caracteres especiais, menos sintaxes complexas e mais estruturas de código simples.

- * facilidade para aprender, ler e compreender;
- * Usa indentação para marcação de blocos;
- * Quase nenhum uso de palavras-chave associadas com compilação;
- * Possuir coletor de lixo para gerenciar automaticamente o uso de memória;
- * Programação orientada a objetos;
- * Programação funcional;
- * Uma imensidão de módulos de extensão, os quais permitem expandir o poder da linguagem Python.

The Zen of Python

Para muitos, Python possui uma filosofia de programação. Parte da cultura da linguagem gira ao redor de *The Zen of Python*, que é um poema escrito pelo programador Tim Peters. Você pode conhecê-lo simplesmente digitando **import this**.

VARIÁVEIS E CONSTANTES

- * **Variável** é um espaço alocado na memória de um computador para armazenar dados ou expressões que estão sendo utilizados por um programa em execução. Existe apenas durante a execução do programa.
- Em Python, podemos declarar uma variável simplesmente definindo um nome para ela e atribuindo-lhe um valor.

```
1  aluno = "Matheus Silva"  
2  periodo_corrente = 10  
3  gravidade = 9.81  
4  v = True
```

VARIÁVEIS E CONSTANTES

- * Python é *case sensitive* - há distinção de letras minúsculas para letras maiúsculas.
- * Palavras reservadas da linguagem Python não podem ser usadas como nomes de variáveis.

and	as	assert	break	class	continue	def
del	elif	else	except	False	finally	for
from	global	if	import	in	is	lambda
None	nonlocal	not	or	pass	raise	return
True	try	while	with	yield		

VARIÁVEIS E CONSTANTES

- * Ao contrário de variáveis, cujos valores variam durante a execução de um programa, quando um valor é atribuído a um **constante**, este não poderá ser modificado até que o programa seja encerrado.

```
1  from math import pi
2  from math import tau, e
3  print(pi)
4  print(tau, e)
```

TIPOS DE DADOS

- * Os principais tipos de dados do Python são:
 - **int** – armazena valores numéricos inteiros
 - **float** – armazena valores numéricos com ponto flutuante
 - **complex** – armazena valores numéricos complexos
 - **bool** – armazena valores lógicos (True ou False). O valor True pode ser representado por 1 e o False por 0
 - **str** – armazena cadeias de caracteres
 - **list** – armazena conjuntos de elementos que podem ser acessados por meio de um índice
 - **dic** – armazena um conjunto de elementos que podem ser acessados por meio de uma chave

TIPOS DE DADOS

- * Diferentemente de outras linguagens de programação o Python possui **tipagem dinâmica**, ou seja, não exige que o programador declare, explicitamente, o tipo de dado que será armazenado por cada variável. Logo, ao longo da execução de um programa, uma mesma variável armazene valores de tipos distintos.

```
1  exemplo = "Ada Lovelace"  
2  exemplo = 10  
3  exemplo = 9.99  
4  exemplo = True
```

- * Para analisar o tipo de uma variável usamos a função `type(variável)`

TIPOS DE DADOS

- * Para analisar o tipo de uma variável usamos a função **type(variável)**

```
1  exemplo = "Ada Lovelace"
2  print(type(exemplo)) #Imprime <class 'str'>
3  '''Ada despertou uma admiração pelos trabalhos desenvolvidos por é
4  Charles Babbage que resultou no interesse de ser sua aluna. Babbage
5  considerado o projetista do primeiro computador de uso geral.'''
6  exemplo = 10
7  print(type(exemplo)) #Imprime <class 'int'>
8  exemplo = 9.99
9  print(type(exemplo)) #Imprime <class 'float'>
10 exemplo = True
11 print(type(exemplo)) #Imprime <class 'bool'>
```


OPERADORES ARITMÉTICOS

- * Utilizados para realizar operações matemáticas.

OPERADOR	DESCRIÇÃO	EXEMPLO DE APLICAÇÃO
+	Adição	<code>print(4 + 2)</code> #resulta em 6
-	Subtração	<code>print(4 - 2)</code> #resulta em 2
*	Multiplicação	<code>print(4 * 2)</code> #resulta em 8
/	Divisão	<code>print(4 / 3)</code> #resulta em 1.3333
//	Quociente inteiro da divisão	<code>print(4 // 3)</code> #resulta em 1
%	Resto da divisão inteira	<code>print(4 % 2)</code> #resulta em 0
**	Potenciação	<code>print(4 ** 2)</code> #resulta em 16

OPERADORES ARITMÉTICOS

- * Os operadores de **adição** e de **multiplicação** apresentam um comportamento diferente quando são utilizadas **strings**.
- O comando **print("Olá, " + "mundo!")** concatena a string **Olá** com a string **mundo!** e resulta em **Olá, mundo!**
- O operador de multiplicação, quando combinado a strings, repete a string **N** vezes, por exemplo, **print(2 * "ABC")** replica 2 vezes a string **ABC**, resultando em **ABCABC**.

OPERADORES DE ATRIBUIÇÃO

- * Utilizados para atribuir valores a variáveis. Combinam a operação com o processo de atribuição, permitindo que expressões sejam mais concisas e eficientes.

OPERADOR	DESCRIÇÃO	EXEMPLO DE APLICAÇÃO
=	Atribuição simples	<code>x = 2</code> <code>#x recebe 2</code>
+=	Atribuição de adição	<code>x += 2</code> <code>#equivale a x = x + 2</code>
-=	Atribuição de subtração	<code>x -= 2</code> <code>#equivale a x = x - 2</code>
*=	Atribuição de multiplicação	<code>x *= 2</code> <code>#equivale a x = x * 2</code>
/=	Atribuição de divisão	<code>x /= 2</code> <code>#equivale a x = x / 2</code>
%=	Atribuição de resto inteiro da divisão	<code>x %= 2</code> <code>#equivale a x = x % 2</code>
**=	Atribuição de potência	<code>x **= 2</code> <code>#equivale a x = x ** 2</code>
//=	Atribuição de quociente inteiro da divisão	<code>x //= 2</code> <code>#equivale a x = x // 2</code>

ENTRADA DE DADOS E CONVERSÃO DE TIPOS

- * Em um cenário real, geralmente o usuário interage com o programa informando dados de entrada. Em Python, utiliza-se a função **input()**, que sempre retornará o valor recebido no formato de uma string, mesmo que ele seja de um tipo de dado diferente.

```
1  aluno = input("Digite seu nome: ")
2  periodo_corrente = input("Digite seu período corrente: ")
3  ira = input("Digite seu IRA(Índice de Rendimento Acadêmico): ")
4  laurea = input("Informe se você é o aluno laureado(a) da turma: ")
5  print(type(aluno))
6  print(type(periodo_corrente))
7  print(type(ira))
8  print(type(laurea))
```

CONVERSÃO DE TIPOS

- * Para contornar esse problema basta usar as funções de conversão **int()**, **float()** e **bool()**, respectivamente.

```
1  aluno = input("Digite seu nome: ")
2  periodo_corrente = int(input("Digite seu período corrente: "))
3  ira = float(input("Digite seu IRA(Índice de Rend. Acadêmico): "))
4  laurea = bool(input("Informe se você foi laureado(a) na turma: "))
5  print(type(aluno))
6  print(type(periodo_corrente))
7  print(type(ira))
8  print(type(laurea))
```

ONDE ENCONTRAMOS ESTRUTURAS CONDICIONAIS?

- * Instruções de um programa Python são executadas na ordem em que foram inseridas no código, ou seja, uma após a outra, do início ao fim. Esse procedimento é conhecido como **execução sequencial**.
- * Em várias ocasiões, é necessário decidir a ordem de execução dos comandos a partir de condições pré-estabelecidas.
- * As estruturas condicionais são responsáveis por definir, a partir do resultado de um teste condicional, quais comandos serão executados e/ou quais deixarão de ser executados.

Obs.: Uma informação importante: em Python, assim como em outras linguagens de programação, uma expressão pode ser considerada verdadeira se ela possui valor diferente de zero. Caso contrário, é considerada falsa.

ESTRUTURA IF...ELSE

- * Temos a palavra-chave **if**, seguida de um teste lógico, valor ou variável e o caractere dois pontos (:).
- * Nas linhas seguintes, temos um bloco de instruções, que em Python, é caracterizado por uma ou várias instruções indentadas.

```
if <condição> :  
    <instrução caso condição seja verdadeira>
```

Exemplo:

```
1 nome = input("Digite seu nome completo: ")  
2 if len(nome) > 50: #len() retorna o número de caracteres da string  
3     print("Seu nome é grande, ele possui mais de 50 letras. ")  
4     print(f"Ele possui {len(nome)} caracteres.")
```

ESTRUTURA IF...ELSE

- * A palavra-chave **else** é aplicada nas condições em que é necessário executar instruções quando o teste do if não for satisfeito.
- * Sua sintaxe é simples, quando encerradas as instruções do if, no mesmo alinhamento do if, adiciona-se a palavra-chave **else** e o caractere dois pontos (:)

```
1  nota1 = int(input("Informe a nota do bimestre 1 (0-100): "))
2  nota2 = int(input("Informe a nota do bimestre 2 (0-100): "))
3  media = (nota1 + nota2) / 2
4  if media >= 60:
5      print("Aprovado")
6  else:
7      print("Reprovado")
```


ESTRUTURA ELIF

- * A palavra-chave **elif** é uma espécie de contração do comando **else** seguido de um comando **if** e é utilizado para diminuir o uso de indentação, facilitando a visualização do código e evitando o crescimento lateral do código por causa do excesso de indentação.
- * É utilizada em substituição ao comando **else**, mas permitindo a realização de um novo teste.

ESTRUTURA ELIF

- * Sua sintaxe também é simples, quando encerradas as instruções do **if**, no mesmo alinhamento do **if**, adiciona-se a palavra-chave **elif** seguida do teste e o caractere dois pontos (:)

```
1  numeroCamisas = int(input("Número de camisas: "))
2  valorCamisa = 12.50
3  valorFinal = numeroCamisas * valorCamisa
4  if numeroCamisas <= 5:
5      valorFinal = valorFinal * (1 - 3/100)
6  elif numeroCamisas <= 10:
7      valorFinal = valorFinal * (1 - 5/100)
8  else:
9      valorFinal = valorFinal * (1 - 7/100)
10 print(f"Valor final: R$ {valorFinal:.2f}")
```

OPERADORES

- * Operadores comparativos e lógicos estão diretamente ligados com estruturas condicionais, pois na maioria das vezes que utilizamos comandos **if** ou **elif**, temos um teste associado com a utilização de operadores comparativos.

OPERADORES DE COMPARAÇÃO OU RELACIONAIS

- * Comparam ou relacionam valores.

OPERADOR	DESCRIÇÃO	EXEMPLO DE APLICAÇÃO
<code>==</code>	Igual	<code>3 == 2</code> #resulta em False
<code>!=</code>	Diferente	<code>3 != 2</code> #resulta em True
<code>></code>	Maior que	<code>3 > 2</code> #resulta em True
<code><</code>	Menor que	<code>3 < 2</code> #resulta em False
<code>>=</code>	Maior ou igual a	<code>3 >= 2</code> #resulta em True
<code><=</code>	Menor ou igual a	<code>3 <= 2</code> #resulta em False

OPERADORES LÓGICOS

OPERADOR	EXPRESSÃO	EXEMPLO DE APLICAÇÃO
and	X and Y	True and False #resulta em False
or	X or Y	True or False #resulta em True
not	not X	not False #resulta em True
		not True #resulta em False