INTRODUÇÃO À LINGUAGEM PYTHON

Professor Valdinei Saugo

valdinei@saugo.com.br

Linguagem Python

Em 1989, Guido Van Rossum, publicou a primeira versão do Python, através do Instituto de Pesquisa Nacional para Matemática e Ciência da Computação, Holanda.

Derivada da linguagem C, Python surgiu para ser uma alternativa mais simples e produtiva que a própria linguagem C.

Linguagem Python

Em 1991 Python ganha sua primeira versão estável, começando a gerar uma comunidade de desenvolvedores empenhada em aprimorá-la, porém só no ano de 1994 que foi lançada a versão 1.0, ou seja, a primeira versão oficial e não mais de testes. Desde seu lançamento oficial, Python já passou por diversos aperfeiçoamentos em sua estrutura e bibliotecas.

Linguagem Python

Atualmente a linguagem Python está integrada em diversas novas tecnologias, assim como é fácil implementá-lo em sistemas obsoletos. Grande parte das distribuições Linux possuem Python nativamente e seu reconhecimento fez que virasse a linguagem padrão do curso de ciências da computação do MIT desde 2009.

Linguagem Python

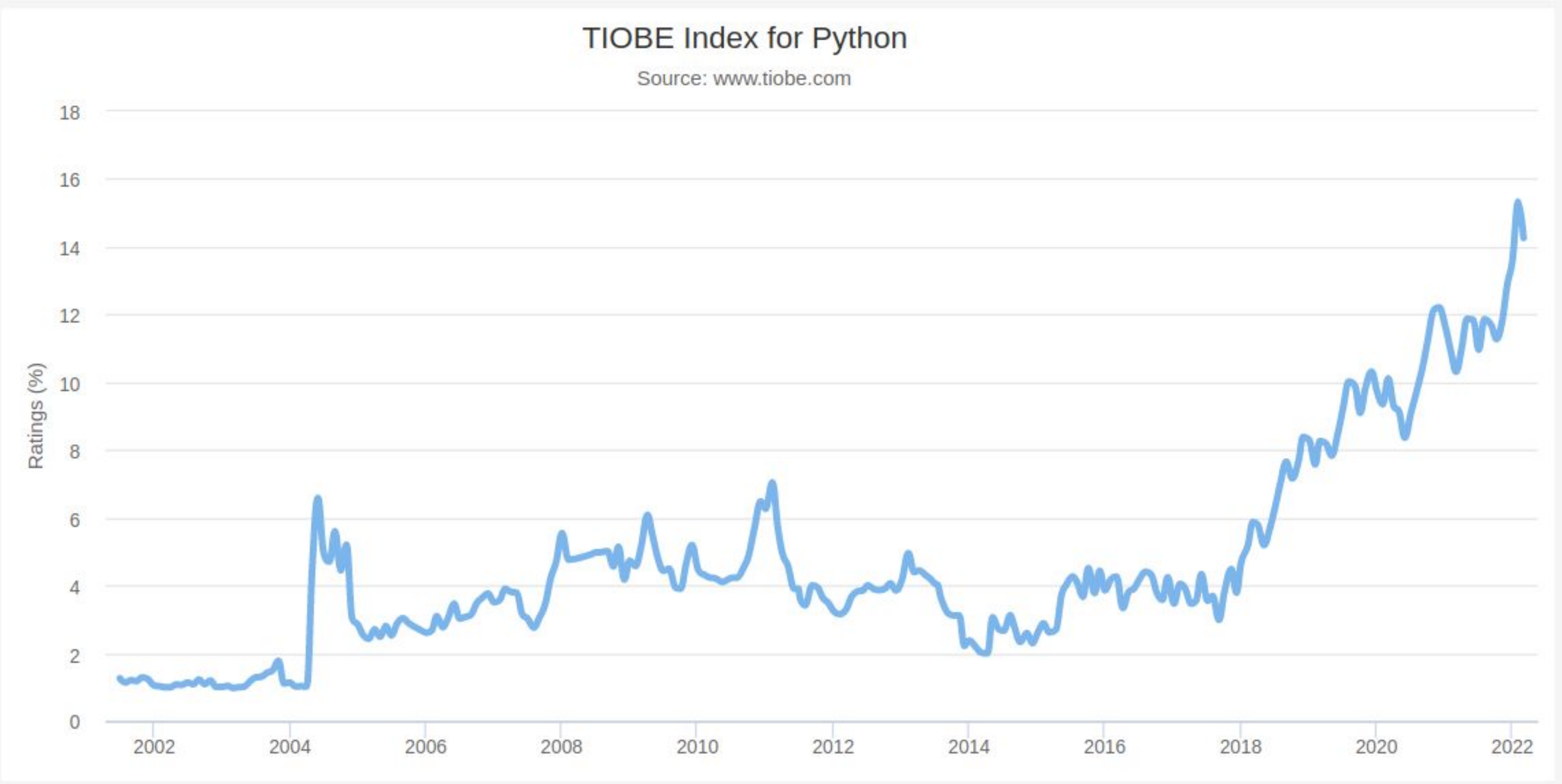
Python é uma linguagem de programação moderna, com ela é possível criar qualquer tipo de sistema para qualquer propósito e plataforma. Com Python é possível desenvolver para qualquer sistema operacional, web, mobile, data science, machine learning, internet das coisas etc. Exemplos de aplicações que usam parcial ou totalmente Python:

- Youtube
- Google
- Instagram
- DropBox
- Spotfy
- Reddit

Linguagem Python

De acordo com as estatísticas de sites de linguagens, Python é uma das linguagens com maior crescimento em relação as demais no mesmo período. De acordo com o Tiobe Index foi a principal linguagem de programação dos anos 2007, 2010, 2018 e 2020.

Linguagem Python



Linguagem Python

Em países mais desenvolvidos tecnologicamente e até mesmo escolas de ensino fundamental estão adotando o ensino de programação em sua grade de disciplinas, boa parte delas ensinando linguagem Python. Espera-se que a linguagem Python cresça exponencialmente, uma vez que novas áreas como Data Science, Machine Learning, aplicações de IoT se popularizem ainda mais.

Linguagem Python

Estudos indicam que no Brasil em 2030 haverá uma oferta cerca de um milhão de novas vagas demandando profissionais de programação, nos mais variados segmentos de mercado (comércio, indústria, iot, data science, inteligência artificial) certamente uma parcela dessa demanda será para programadores com conhecimentos em Python.

Variáveis e Tipos de Dados

Python é uma linguagem “batteries included”, termo em inglês para pilhas incluídas, ou seja, ele já vem com o necessário para seu funcionamento pronto para uso. O clássico exemplo “Olá mundo” para ser apresentado na tela é escrito da seguinte forma:

```
print(“Olá Mundo!”)
```

Tipos de Dados

Tipo	Descrição	Exemplo
int	Número inteiro, sem casas decimais	15
float	Número real, com casas decimais	27.8
bool	Booleano / Valores Lógicos	0 / True
str	Texto com qualquer caractere alfanumérico	‘Curso linguagem Python’
list	Listas	[4, ‘João’, 45.8]
dict	Dicionários	{‘nome’:’Agenor Silva’,’idade’:23}

Variáveis e Tipos de Dados

Note que cada tipo de dado possui uma sintaxe própria (forma de escrita) para que o interpretador os reconheça como tal. Exemplos:

msg = “Programação Python”

idade = 23

valor_produto = 125,78

opcao = True

lista_nomes = [1, 6, ‘Maria’, ‘Joana’]

dicionario_01 = {‘nome’:’Jonas’, ‘idade’:35}

.

Declaração de Variáveis

A linguagem Python é um tipo de linguagem de programação conhecida como dinamicamente tipada, ou seja, Python não obriga o desenvolvedor a tipar explicitamente as variáveis de um programa, ele automaticamente identifica o tipo de dado quando usamos o sinal de atribuição **‘=‘**.

Exemplo declaração de uma variável para armazenando um número inteiro:

numero01 = 10

Declaração de Variáveis

Leitura de dados pelo teclado: para facilitar a interação do usuário com um programa em Python, podemos ler conteúdo digitado pelo teclado e armazenar em variáveis. Para isso usamos o comando '***input***', veja o exemplo:

```
1  entrada = input("Digite um número inteiro:")
2  numero_01 = int(entrada)
3  entrada = input("Digite outro número inteiro:")
4  numero_02 = int(entrada)
5  print("A soma dos números é : ", numero_01 + numero_02)
```

Declaração de Variáveis

Declarando variáveis explicitando seu tipo: conforme visto anteriormente é possível declarar uma variável determinando seu tipo e depois atribuir um valor, veja o exemplo:

```
1  variavel_01: int
2  variavel_02: int
3  variavel_01 = 20
4  variavel_02 = 55
5  print(variavel_01 + variavel_02)
```

Exemplos Práticos

Crie um programa Python para ler via teclado:

1. Uma variável int
2. Uma variável flot
3. Uma variável boolean
4. Uma variável str
5. Imprima os conteúdos das variáveis

Operadores Aritméticos

Operadores Aritméticos	
Operador	Função
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto inteiro da divisão
**	Exponenciação
//	Divisão que retorna um número inteiro arredondado

Operadores Relacionais

Operadores relacionais são em essência aqueles que fazem a comparação entre dois ou mais operandos. Estes operadores possuem uma sintaxe própria que deve ser respeitada para que não haja conflito com o interpretador de código Python. O resultado de uma comparação relacional será dada como um valor booleano, **True** ou **False.**

Operadores Relacionais

Operadores Relacionais	
Operador	Função
>	Maior que
<	Menor que
>=	Maior ou igual que
<=	Menor ou menor que
==	Igual a
!=	Diferente de

Operadores Relacionais

```
1  variavel_01 = 12
2  variavel_02 = 7
3  print(variavel_01 > variavel_02)
```

```
True
```

```
1  variavel_01 = 12
2  variavel_02 = 7
3  print(variavel_01 == variavel_02)
```

```
False
```

Operadores Relacionais

```
1  variavel_01 = 12
2  variavel_02 = 7
3  print(variavel_01 >= variavel_02)
```

```
True
```

```
1  variavel_01 = 12
2  variavel_02 = 7
3  print(variavel_01 <= variavel_02)
```

```
False
```

Exemplos Práticos

Leia dois números via teclado e imprima os resultados para

1. num01 == num02
2. num01 + num02 > num01 / 2
3. num01 != num02
4. num01 >= 0
5. num02 >= -1

Operadores Lógicos

Os operadores lógicos, possuem a mesma base lógica dos operadores relacionais, retornar como valores das comparações valores lógicos **True** ou **False.** A principal diferença que com operadores lógicos é possível montar expressões lógicas de maior complexidade, podendo inclusive incluir operadores matemáticos e relacionais na mesma expressão.

Operadores Lógicos

Operadores Lógicos	
Operador	Função
and	Operador Lógico E
or	Operador Lógico OU
not	Operador de negação

Operadores Lógicos

Por exemplo, a expressão **7 != 3** tem como valor **True** (pois: 7 é diferente de 3, verdade), mas se prepararmos a expressão **7 != 3 and 3 > 5** o resultado será **False.**

7 ! = 3 -> True

3 > 5 -> False

True and False -> False

.

Tabela Verdade Operadores and - or

Tabela verdade operador lógico and				
V	and	V	->	True
V	and	F	->	False
F	and	V	->	False
F	and	F	->	False

Tabela verdade operador lógico or				
V	or	V	->	True
V	or	F	->	True
F	or	V	->	True
F	or	F	->	False

Operadores Lógico not

A aplicação do operador ***not*** é inverter o resultado de um tipo lógico ou expressão booleana, vejamos o exemplo:

a = 12

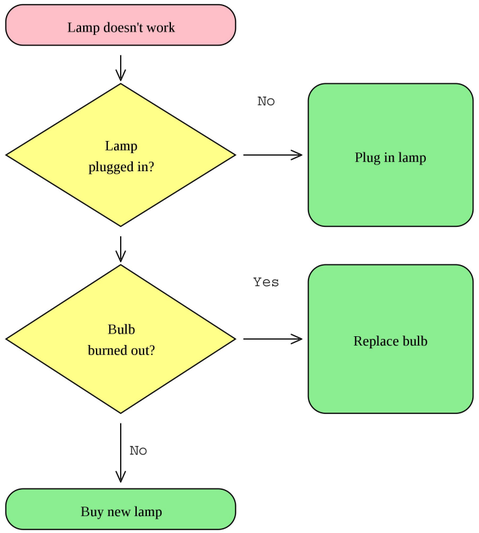
b = 3

print(a > b) -> False

print(not a > b) -> True

Estruturas de Decisão

Quando estudamos sobre lógica de programação e algoritmos é importante entendermos que toda ação tem uma reação, dessa forma, quando transcrevemos ideias para código uma coisa que muito ocorre é nos depararmos com tomadas de decisão, as quais irão influenciar os rumos de execução do nosso programa.



Estruturas de Decisão

A sintaxe de programação do Python para trabalhar com condicionais é bastante simples em comparação com outras linguagens. A seguir veremos as estruturas de decisão da linguagem Python:

- **if**
- **else**
- **elif**

Estruturas de Decisão - comando if

```
1  num_01 = 15
2  num_02 = 45
3  num_03 = 5
4  if num_01 > num_02:
5      print("num_01 é maior que num_02")
```

Um comando if sempre será seguido de uma instrução que sendo verdadeira, irá executar um bloco de código indentado a ela.

Estruturas de Decisão - if e else

```
1  num_01 = 15
2  num_02 = 45
3  num_03 = 5
4  if num_01 > num_02:
5      print("num_01 é maior que num_02")
6  else:
7      print("Opção inválida")
```

Um comando **if** sempre será seguido de uma instrução que sendo verdadeira, irá executar um bloco de código indentado a ela, caso a validação retorne False, podemos combinar o comando **else** para identificar que a tentativa no if foi falsa.

Estruturas de Decisão - elif

```
1  num_01 = 15
2  num_02 = 45
3  num_03 = 5
4  if num_01 > num_02:
5      print("num_01 é maior que num_02")
6  elif num_01 < num_02:
7      print("num_01 é menor que num_02")
8  else:
9      print("num_01 e num_02 são iguais")
```

Neste código observa-se a aplicação do comando **elif**, que deve ser aplicado quando for necessário novas validações após o comando if. Quando nenhuma das validações forem corretas podemos finalizar a estrutura com o comando **else** para indicar que nenhuma condição foi satisfeita.

Exemplos Práticos

a. Verifique se um número é maior que zero e menor que 34
b. Verifique se um número é par ou ímpar
c. Leia 3 números inteiros e verifique se os mesmo formam um triângulo

Estruturas de Repetição

A linguagem Python possui dois comandos em **loop**, ou seja, em estruturas de repetição. Usaremos repetição para executar várias vezes uma mesma instrução. Os comandos que veremos na sequência são:

- **for**
- **while**

Estruturas de Repetição

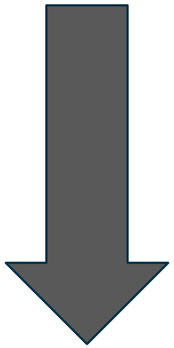
FOR

O comando **for** será utilizado em situações que precisamos trabalhar repetições onde conhecemos seus limites, ou seja, quando sabemos onde começa e termina. Exemplo:

```
1   for x in range(0, 6):
2       print(x)
```

Estruturas de Repetição

```
1   for x in range(0, 6):
2       print(x)
```



```
0
1
2
3
4
5
```

Note que no início da linha existe o comando **for** seguido de uma variável temporária **x**, logo em seguida o comando **in range**, que basicamente define um intervalo a percorrer (de 0 até 6) e finalizando o comando print para exibir o conteúdo de x.

A contagem dos elementos do intervalo foi de 0 a 5, o número 6 está fora do range, servindo apenas como orientação um contador para as repetições.

Estruturas de Repetição

```
1  lista_de_nomes = ['João', 'Ana', 'Maria', 'Valdir']
2  for n in lista_de_nomes:
3      print(n)
```

```
João
Ana
Maria
Valdir
```

Outra situação comum é quando já temos uma lista de elementos e queremos percorrê-la para exibir seu conteúdo. Neste exemplo como já temos uma lista com 4 nomes e quando o **for** é executado, ele internamente percorrerá todos os valores contidos na lista e inclui os nomes na variável **n** para no final exibir pelo comando **print.**

Estruturas de Repetição

WHILE

O comando **while** será executado enquanto em sua instrução houver uma condição verdadeira.

Exemplo:

```
1   x = 1
2   while x < 8:
3       print(x)
4       x = x + 1
```

Estruturas de Repetição

```
1   x = 1
2   while x < 8:
3       print(x)
4       x = x + 1
```

```
1
2
3
4
5
6
7
```

Declarada a variável **x** com valor inicial 1 e em seguida colocada a condição de que enquanto o valor de **x** for menor que 6, imprime o valor de **x** e acrescenta 1 repetidamente. Perceba que isto é um **loop,** ou seja, a cada ação o bloco de código salva seu último estado e repete a instrução, até atingir a condição deixar de ser verdadeira.

Estruturas de Repetição

Outra possibilidade é que durante a execução do **while**, é possível programar um **break**, que é um comando que interrompe a execução de um determinado bloco de código ou instrução. Normalmente o uso de break se dá quando é utilizada mais de uma condição que, se a instrução de código atingir qualquer uma dessas condições ele para sua execução. Por exemplo:

Estruturas de Repetição

```
1   x = 1
2   while x < 10:
3       print(x)
4       x = x + 1
5       if x == 5:
6           break
```

Enquanto o valor da variável x for menor que 10, continue imprimindo seu conteúdo e adicionando 1 ao seu valor, mas se em algum momento x for igual a 4, pare a repetição.

Exercício

Elaborar um código para realizar as funções de uma calculadora.

Apresentar um menu com as seguintes opções:

1 - para soma

2 - para subtração

3 - para multiplicação

4 - para divisão

Se o usuário informar uma opção inválida, o programa deve retornar ao menu de opções.