Introdução à linguagem Python

Programação de Novas Tecnologias

- A linguagem Python foi criada em 1991 pelo programador holandês Guido van Rossum.
- A primeira versão, 0.9.0, foi lançada em fevereiro de 1991.
- O nome da linguagem é inspirado no grupo humorístico britânico Monty Python.

python

Popularidade Python

O índice TIOBE de fevereiro de 2024 mostra as seguintes linguagens de programação como as 10 mais populares:

1. Python
2. C
3. C++
4. Java
5. C#
6. JavaScript
7. Sql
8. Go
9. Visual Basic
10. PHP

Python é frequentemente usado em ciência de dados, inteligência artificial, web development, automação de tarefas, entre outras áreas.

Popularidade Python

- **Python** é uma linguagem de programação de alto nível e fácil de aprender, com uma sintaxe limpa e clara.
- **Python** tem uma vasta biblioteca padrão e muitas bibliotecas de terceiros, o que permite aos desenvolvedores criar aplicativos de maneira eficiente.
- **Python** é uma linguagem interpretada, o que significa que não é necessário compilar o código antes de executá-lo.
- **Python** é uma linguagem multiplataforma, o que significa que pode ser executada em vários sistemas operacionais (Windows, Linux, MacOs).
- **Python** tem uma comunidade ativa e vibrante, com muitas pessoas contribuindo para o desenvolvimento da linguagem e criando **bibliotecas** de código aberto para outras pessoas usarem.

Instalação do Python

- Em ambiente Desktop existe instalador direto para Windows, Linux e MacOs.
- Existem sites que rodam serviços online para execução de código fonte Python. (Ambiente de testes)
- Portanto, qualquer um pode programar em Python até no seu celular.

Sintaxe x Semântica

- Sintaxe:
  - **Regras**: Define como as palavras-chave, operadores e símbolos da linguagem de programação são combinados para formar frases válidas.
  - **Analogia**: É como a gramática de uma língua natural, definindo a estrutura e a ordem das palavras em uma sentença.

Sintaxe x Semântica

- Ex:
  - **x = 1 + 2 (Python)**
  - **$x = 1 + 2; (PHP)**
  - **int x = 1 + 2; (Java)**
- **Erros sintático (Python):**
  - **x = 12.5 * w**

```
File "<main.py>", line 1, in <module>

NameError: name 'w' is not defined
```

Sintaxe x Semântica

- Semântica:
  - **Significado**: Define o significado e a interpretação das frases válidas na linguagem de programação.
  - **Regras**: Define como as instruções do código são traduzidas em ações e operações pelo computador.
  - **Analogia**: É como o significado de uma frase em uma língua natural, definindo o que ela transmite.

Sintaxe x Semântica

- Ex:

```
n1 = 9.0
n2 = 4.5
media = n1 / n2
print(media)
```

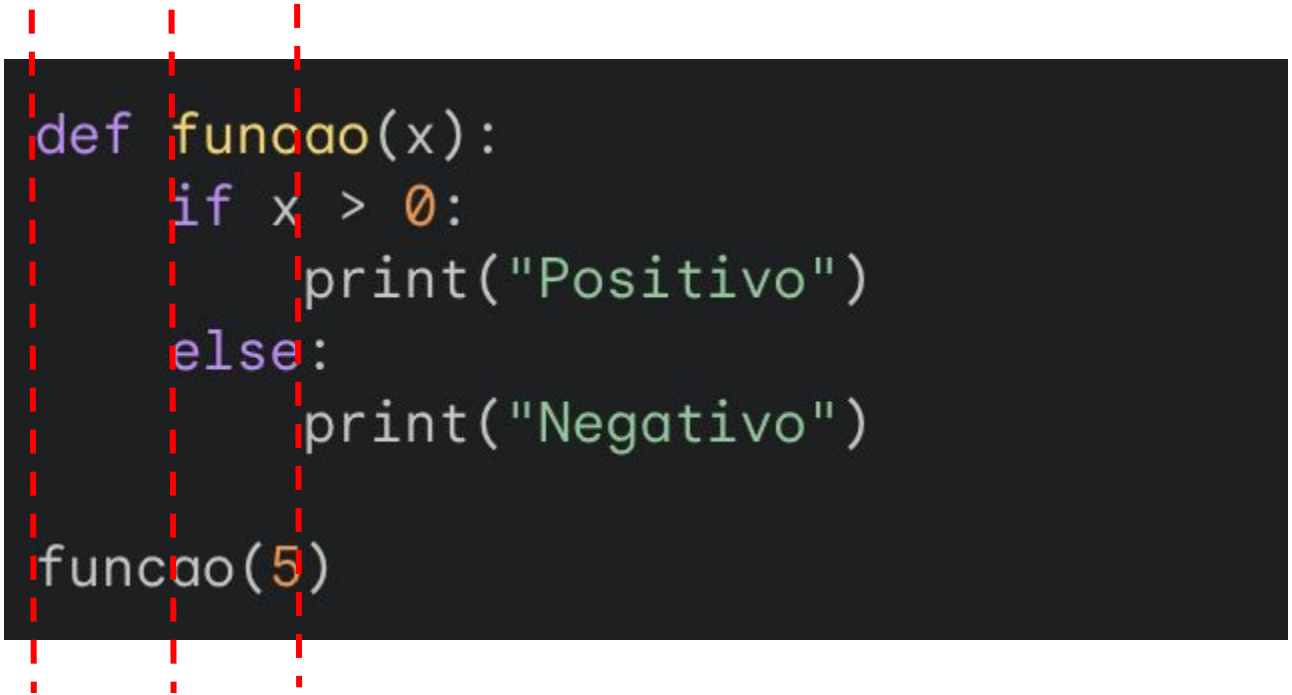
Resultado: 2

Sintaxe Python

- Indentação:
  - A indentação é crucial para a estrutura do código Python.
  - Blocos de código são delimitados por espaços, não por chaves.
  - Quatro espaços são a indentação padrão, mas o número é irrelevante, desde que consistente.
  - A indentação incorreta gera erros de sintaxe.

Sintaxe Python

- Indentação:



Variáveis

- Variáveis em programação são espaços na memória do computador que armazenam dados.
- Elas são nomeadas e podem ser de diferentes tipos, como números, textos ou listas.
- O valor de uma variável pode ser alterado durante a execução do programa.
- As variáveis são essenciais para armazenar e manipular dados em programas.

Variáveis e Tipos de dados

- **Variáveis numéricas:** São usadas para armazenar valores numéricos, como inteiros (int), números de ponto flutuante (float) e números complexos (complex).
    - num = 13
    - altura = 1.68
- **Variáveis de texto (ou strings)**: São usadas para armazenar texto ou sequências de caracteres, como palavras, frases e até mesmo código fonte. Strings em Python são representadas entre aspas simples (') ou duplas ("").
    - nome = "Joaquina Mariana"
    - frase = 'Quando surge o alviverde imponente'

Variáveis e Tipos de dados

- **Variáveis booleanas:** São usadas para armazenar valores lógicos, como **True** ou **False**. São frequentemente usadas para fazer comparações ou testes lógicos em expressões condicionais.
  - ehVerdade = True
  - ehPrimo = False
- **Variáveis de lista (Arrays):** São usadas para armazenar uma coleção de valores em uma única variável. As listas podem armazenar vários tipos de dados, como números, strings, outras listas e objetos.
  - numeros = [10,20,30,40,50]
  - diasDaSemana = ["Segunda","Terça","Quarta","Quinta","Sextou"]

Variáveis e Tipos de dados

- É importante lembrar que, em Python, as variáveis são **dinamicamente tipadas**, o que significa que o tipo de uma variável é determinado automaticamente pelo tipo de dado que ela contém.
- Além disso, as variáveis em Python são sensíveis a maiúsculas e minúsculas (**case sensitive**), ou seja, "var" e "VAR" são variáveis diferentes.

Variáveis e Tipos de dados

- Tipo de variáveis:
  - **Int** - números inteiros
  - **Float** - números com ponto flutuante
  - **String** - Texto
  - **Boolean** - variáveis lógicas
  - **Arrays** - lista de dados

Variáveis e Tipos de dados

- Tipo de variáveis:
  - a = "Bernadete Fabiana"
  - b = 21.3
  - c = False
  - d = ["a", "b", "c"]
  - e = 2048
  - f = "4096"

Operadores

- Existem vários tipos de operadores disponíveis em Python, que podem ser divididos em diferentes categorias, como operadores aritméticos, de comparação, lógicos, de atribuição, entre outros.
- A seguir, apresentamos exemplos de cada tipo de operador.

Operadores aritméticos

- São usados para realizar operações matemáticas básicas.

  Os operadores aritméticos em Python incluem:
  - Soma (+)
  - Subtração (-)
  - Multiplicação (*)
  - Divisão (/)
  - Módulo (%) usado para encontrar o resto da divisão entre dois valores
  - Exponenciação (**) usado para elevar um valor a uma potência

Operadores aritméticos

- Exemplos:
  - print(5 + 4). #resultado 9
  - print(5 * 4). #resultado 20
  - print(15 % 4). #resultado 3 (15 / 4 = 12. Resto = 3)
  - print(2 ** 3). #resultado 8 (2^3)
  - print(8+2*23-5) ?

Operadores de comparação

- São usados para comparar dois valores e retornar um valor booleano (True ou False).
- Os operadores de comparação em Python incluem:

Operadores de comparação

- **Igual (==):** usado para verificar se dois valores são iguais
- **Diferente (!=):** usado para verificar se dois valores são diferentes
- **Maior que (>):** usado para verificar se um valor é maior que outro
- **Menor que (<):** usado para verificar se um valor é menor que outro
- **Maior ou igual (>=):** usado para verificar se um valor é maior ou igual a outro
- **Menor ou igual (<=):** usado para verificar se um valor é menor ou igual a outro

Operadores de comparação

- Exemplos:

```
print("Redes A" == "Redes B") #Resultado False

print("Moto" != "moto") #Resultado True

print(120 < 200) #Resultado True

print(10 >= 10) #Resultado True

print(1 < -9999) #Resultado False

print("Segunda" < "Sexta") #Resultado False
```

Operadores Lógicos

- Em Python, existem três operadores lógicos principais:
  - Negação (not)
  - E (and)
  - OU (or).

Operadores Lógicos

- Negação (not)
- O operador de negação (not) é usado para inverter o valor de uma expressão booleana. Se a expressão for verdadeira, o valor será falso e vice-versa.

  Exemplo:

```
a = True
b = (not a)
print(b) #Resultado é False
```

Operadores Lógicos

- E (and)
- O operador E (and) é usado para verificar se duas expressões booleanas são verdadeiras. **Se ambas as expressões forem verdadeiras**, o valor será verdadeiro. Caso contrário, o valor será falso. Exemplo:

```
a = True
b = False
c = (a and b)
print(c) # resultado é False
```

Operadores Lógicos

- OU (or)
- O operador OU (or) é usado para verificar se pelo menos uma das expressões booleanas é verdadeira. **Se pelo menos uma das expressões for verdadeira**, o valor será verdadeiro. Caso contrário, o valor será falso. Exemplo:

```
a = True
b = False
c = (a or b)
print(c) #Resultado é True
```

Operadores Lógicos

- É possível usar os operadores lógicos em conjunto com os operadores de comparação.
- Por exemplo, é possível verificar se uma variável está dentro de um determinado intervalo usando os operadores lógicos e os operadores de comparação. Exemplo:

```
a = 10
b = 5
c = 15
d = (a > b and a < c)
print(d) #Resultado é True
```