**PYTHON DICAS!**

**Material criado por:**

###### © Prof. Iraê César Brandão - Desenvolvedor Web / Gestor de TI / Empresário / Professor. Linkedin : <https://www.linkedin.com/in/irae-cesar-brandao-a2112b69/>

**Data da ultima atualização: 01/05/2023 - Sem fins lucrativos.**

**PYTHON**

É uma linguagem de programação versátil , gratuito, de código aberto e multiplataforma de alto nível e poderosa, usada para desenvolvimento web, com uma grande comunidade de desenvolvedores e muitas bibliotecas e ferramentas disponíveis para diferentes áreas de aplicação:

- Análise de dados, automação de tarefas;

- Machine Learning e Inteligência Artificial;

- Aplicações desktop, graças a muitas bibliotecas e frameworks disponíveis.

Projetada para ser fácil de entender, ensinar, escrever, instalar e com uma sintaxe clara e legível.

**INSTALAÇÃO DO PYTHON**

O método exato para instalar o Python pode variar dependendo do seu sistema operacional e da versão específica do Python que você deseja instalar, basta escolhe seu sistema operacional e fazer download do aplicativo em: <https://www.python.org/downloads/windows/>

**EDITOR, CONSOLE E DEBUG**

Para usá-lo basta saber Python 3 contém um aplicativo muito simples, mas extremamente útil chamado IDLE, que é o ambiente integrado de desenvolvimento e aprendizagem, Oferece recursos como destaque de sintaxe, preenchimento automático e recuo inteligente.

Podemos utilizá-lo também on-line acessando os links abaixo:

**Google Colaboratory PYTHON->** <https://colab.research.google.com/>

**JupyterLite - Notebook ->** <https://jupyter.org/try-jupyter/lab/>

**ELEMENTOS DA LINGUAGEM DE PROGRAMAÇÃO**

As linguagens de programação têm vários elementos que permitem a criação de algoritmos e programas, como:

- **Sintaxe:** é a forma como as instruções e expressões são escritas na linguagem (estrutura, pontuação e regras);

- **Semântica :** é o significado das instruções e expressões em um programa e como se relacionam com o mundo real e com outras instruções do programa;

- **Tipos de dados:** são as diferentes categorias de dados que podem ser usadas em um programa, como números, texto, booleanos, entre outros;

- Variáveis: são os nomes que se atribuem aos valores de dados em um programa, permitindo que eles sejam referenciados e manipulados ao longo da execução do programa;

- **Estruturas de controle:** são as instruções que controlam o fluxo de execução do programa, como condicionais (if-else), loops (for, while) e chamadas de função;

- **Bibliotecas e frameworks (Lexis):** são conjuntos de código pré-existente que podem ser usados em programas para realizar tarefas específicas, como processamento de imagens, manipulação de dados, construção de interfaces gráficas, entre outras.

Todos esses elementos são combinados de diferentes maneiras em cada linguagem, mas permitem que os programadores expressem soluções de problemas de forma estruturada.

**INDENTAÇÃO**

No Python, a indentação é usada para indicar o nível de código aninhado, que é representado por uma certa quantidade de espaços ou tabulações no início de uma linha. A convenção padrão é usar quatro espaços para cada nível de indentação, sendo essa obrigatória em Python. Erros de indentação podem levar a problemas na execução do código.

**COMPILAÇÃO E INTERPRETAÇÃO**

Python combina os processos de compilação e interpretação, onde o código-fonte é interpretado pelo interpretador e depois compilado em bytecode antes da execução.

O bytecode é uma forma intermediária de código que melhora o desempenho e eficiência do programa, e é armazenado em arquivos . O interpret ador verifica o código-fonte linha por linha e exibe uma mensagem de erro se houver erros de sintaxe.

Existem quatro itens importantes na composição das linguagens de programação:

1. a ordem alfabética para escrita (com script reconhecível);
2. o domínio do dicionário léxico da linguagem;
3. a obediência às regras sintáticas da linguagem;
4. coerência semântica do programa (o programa tem de fazer sentido).

**IMPLEMENTAÇÕES**

As implementações conhecidas são: CPython. Esta é a implementação original e a é a versão do Python que mais vem sendo sendo desenvolvido e a mesma está escrita com a linguagem. Existem diversas implementações como: Jython, Cythotn, MicroPython, Python for .Net, IronPython, Pypy e outras.

Maiores informações vejam: documentação Python : <https://docs.python.org/pt-br/3/> .

**FUNÇÕES**

**Algumas das funções mais usadas em Python incluem:**

**print():** exibe uma mensagem na tela

**input():** solicita a entrada de dados do usuário

**len():** retorna o comprimento de uma sequência, como uma string ou uma lista

**range():** cria uma sequência de números

**open():** abre um arquivo para leitura ou escrita

**append():** adiciona um elemento a uma lista

**split():** divide uma string em substrings com base em um delimitador

**join():** une uma sequência de strings em uma única string usando um separador

**strip():** remove espaços em branco de uma string

**sort():** classifica uma lista em ordem crescente ou decrescente.

Essas são apenas algumas das funções mais comuns em Python e existem muitas outras que podem ser usadas para realizar diferentes tarefas.

**MENSAGENS DE ERROS**

Algumas das mensagens de erro mais comuns em Python e quando ocorrem:

**SyntaxError:** quando o interpretador encontra uma sintaxe inválida em um programa Python.

**NameError:** quando o interpretador não consegue encontrar o nome de uma variável ou função usada no programa.

**TypeError:** quando uma operação ou função é aplicada a um tipo de dado incompatível.

**IndexError:** quando uma lista, tupla ou sequência é acessada com um índice inválido.

**ValueError:** quando uma função recebe um argumento com valor inválido ou incorreto.

**KeyError:** quando um dicionário é acessado com uma chave que não existe.

**AttributeError:**  quando um objeto não tem um determinado atributo ou método.

Essas são apenas algumas das mensagens de erro mais comuns em Python, mas há muitas outras possíveis, dependendo do tipo de problema no programa. É importante ler com atenção as mensagens de erro para identificar o problema e corrigi-lo.

**COMO USAR O PYTHON**

**Função print() em Python é usada para exibir informações na tela. Algumas orientações para o seu uso incluem:**

* O argumento da função print() pode ser uma string ou uma expressão que deve ser convertida em string.
* Para imprimir várias informações em uma única linha, separe-as por vírgulas.
* O argumento opcional sep é usado para definir o separador entre os valores impressos.
* O argumento opcional end é usado para definir o caractere final após a impressão dos valores.
* Para formatar a saída, é possível usar as strings de formatação ou o método format().
* Para imprimir em arquivos em vez de na tela, é possível redirecionar a saída usando a sintaxe print(..., file=nome\_do\_arquivo).

Alguns exemplos de uso da função **print()** em Python:

Imprimir uma string simples:

**print("Olá, mundo!")**

Imprimir o valor de uma variável:

**nome = "João"**

**print("Bem-vindo,", nome)**

Concatenar várias strings e variáveis:

**idade = 30**

**cidade = "São Paulo"**

**print("Eu tenho", idade, "anos e moro em", cidade)**

Usar formatação de strings com f-strings:

**produto = "Notebook"**

**preco = 2500.50**

**print(f"O {produto} custa R${preco:.2f}")**

Imprimir em arquivos externos:

**with open("arquivo.txt", "w") as f:**

**print("Esse texto será impresso no arquivo.", file=f)**

Esses são apenas alguns exemplos, a função print() é muito versátil e pode ser usada de diversas maneiras.

**COMENTÁRIOS**

Comentários em Python são utilizados para explicar o código ou desabilitar uma linha de código temporariamente. Eles são indicados pelos caracteres e são ignorados pelo interpretador do Python.:

* **# (hashtag ou cerquilha):**  Iniciando o comentários de uma única e podem ser adicionados no final de uma linha de código ou em uma linha separada.

**#coding: utf-8**

**#tudo que estiver a frente da cerquilha será ignorado**

**print(teste) #a instrução atrás da cerquilha será interpretada**

* Já são iniciados e finalizados com **" "" "** **(três aspas duplas)** ou

**''' ''' ( três aspas simples)** para os comentários de várias linhas.

**#coding: utf-8**

**'''**

**toda informações contida entre 3 aspas SIMPLES**

**é considerada como caracteres que devem ser ignorados.**

**'''**

**toda informações contida entre 3 aspas DUPLAS**

**é considerado texto que deve ser ignorado.**

Maiores informações: COMENTÁRIOS EM PYTHON : <http://excript.com/python/comentarios-em-python.html>

**USO DO END E DO SEP**

O end e o sep são parâmetros opcionais da função print() em Python, que permitem personalizar a maneira como a saída do programa é formatada.

* O parâmetro **end** :

define o que será adicionado ao final da impressão da linha atual. Por padrão, o end é uma nova linha (\n), o que significa que a cada chamada do print(), o cursor será movido para a próxima linha. É possível substituir o valor padrão do end por qualquer outra string, incluindo um espaço vazio, uma quebra de linha, um caractere especial ou qualquer outra coisa.

* O parâmetro **sep:**

define o separador entre os argumentos do print(). Por padrão, o sep é um espaço em branco, mas pode ser personalizado para qualquer outra string. Isso é especialmente útil quando se quer imprimir múltiplos valores, por exemplo, para criar uma tabela ou imprimir dados em colunas.

Ambos os parâmetros são usados em conjunto com a função print(). Por exemplo, o seguinte código imprime uma lista de números em uma única linha, separados por vírgulas e sem espaço entre eles:

**numbers = [1, 2, 3, 4, 5]**

**print(\*numbers, sep=',', end='')**

O  **\*** **(Asterisco)** antes do nome da lista desempacota os valores da lista, permitindo que sejam passados como argumentos separados para a função print(). O sep=',' define a vírgula como separador e end='' remove a nova linha padrão do final da impressão. O resultado será: **1,2,3,4,5** .

**USO DE ASPAS**

No caso de ser preciso usar uma aspas simples dentro de um texto de aspas simples basta usar **\'** e para usar aspas duplas dentro de aspas duplas você utilizaria **\" , aspas triplas '''** ou **"""** que tem a mesma função praticamente.

Observe os exemplos:

**print('''string**

**em de mais**

**de uma linha''')**

**print('Sinal de aspas simples: \'')**

**print("Sinal de aspas dupla: \"")**

**print("""Sinal de aspas simples e dupla: " ' """)**

**print('''Sinal de aspas simples e dupla: " ' ''')**

**LITERAIS**

Em Python, as literais que são comumente usados: inteiros, booleanos e strings. A seguir listamos as literais

* **Literais de Inteiros:** são números inteiros positivos ou negativos, sem decimais. São representados sem nenhuma pontuação ou símbolos, como por exemplo: 5, -10, 0 e 100.
* **Literais de Ponto flutuante** (ou simplesmente **float**), que contêm (ou são capazes de conter) a parte fracionária (Exemplos: 3.14, 6.02e23 (representa o número 6,02 x 10^23).
* **Literais de Booleanos:** representam valores lógicos verdadeiro (True) ou falso (False). São frequentemente usados em condicionais e loops.

Exemplo de boleano:

**x = 5**

**y = 10**

**if x == 5 and y == 10:**

**print("x é igual a 5 e y é igual a 10")**

**if not(x == 5 or y == 5):**

**print("nem x nem y são iguais a 5")**

* **Literais de Strings:** são sequências de caracteres que representam texto. Podem ser escritas entre aspas simples ou duplas, como por exemplo: "Olá, mundo!", 'Python é incrível', "1234".
* **Literais de Sequência:** são sequências de caracteres que podem ser utilizadas para representar dados de texto ou binários em Python, e elas são definidas entre aspas simples (exemplo: texto = "Olá, mundo!"), aspas duplas ou aspas triplas, tuplas e listas (Exemplos: tupla = (1, 2, 3) e lista = [4, 5, 6]).

Exemplo de lista:

**# criando uma lista de frutas**

**#posições dos elementos na lista: 0 , 1 , 2, 3**

**frutas = ['maçã', 'banana', 'laranja', 'uva']**

**# acessando um elemento pelo índice**

**print(frutas[1]) # saída: 'banana'**

**# alterando um elemento laranja por abacaxi posição 2**

**frutas[2] = 'abacaxi'**

**# adicionando um elemento morango no final da lista**

**frutas.append('morango')**

**# removendo um elemento pelo índice na posição 0 maçã**

**del frutas[0]**

**# imprimindo a lista atualizada**

**print(frutas) # saída: ['banana', 'abacaxi', 'uva', 'morango']**

* **Literais de complexos:** representam números complexos, que possuem uma parte real e uma parte imaginária (Exemplos: 3+4j (representa o número complexo 3 + 4i).
* **Literal None:** indicar que uma variável ou objeto não tem valor atribuído ou que uma função não retornou nenhum valor.

Esses tipos de literais são usados em todo o código Python para armazenar e processar informações.

**NÚMEROS OCTAL E HEXADECIMAL**

- **Números octais**: são representados pelo prefixo "0o" seguido de dígitos de 0 a 7.

- **Números hexadecimais:** são representados pelo prefixo "0x" seguido de dígitos de 0 a 9 e letras de A a F.

Esses tipos de números são usados para representar valores em sistemas numéricos diferentes do sistema decimal, que é o sistema padrão usado na maioria dos programas de computador. O Python permite a conversão de números entre diferentes sistemas numéricos usando funções embutidas, como oct() e hex().

SINAIS ARITMÉTICOS (OPERAÇÕES MATEMÁTICAS)

Os sinais aritméticos em Python são utilizados para realizar operações matemáticas. Eles incluem:

* **Adição (+):** utilizado para somar dois valores.
* **Subtração (-):** utilizado para subtrair um valor de outro.
* **Multiplicação (\*):** utilizado para multiplicar dois valores.
* **Divisão (/):** utilizado para dividir um valor pelo outro.
* **Divisão inteira (//):** retorna o resultado da divisão sem o resto da divisão.
* **Módulo (%):** retorna o resto da divisão entre dois valores.
* **Exponenciação (\*\*):** eleva um valor à potência de outro.

**Não dividir (% ou /) por zero (0**): A divisão por zero não funciona (por zero nunca dividir, divisão inteira ou encontra resto de divisão por zero)

**OUTROS SINAIS UTILIZADOS**

Além dos sinais aritméticos, Python possui outros sinais que são utilizados em diversas operações. Alguns exemplos são:

* **Operadores de comparação:** utilizados para comparar valores e retornar um valor booleano (True ou False). Exemplos: < (menor que), > (maior que), == (igual), <= (menor ou igual), >= (maior ou igual), != (diferente).
* **Operadores lógicos:** utilizados para combinar valores booleanos e retornar um novo valor booleano. Exemplos: and (e), or (ou), not (negação).
* **Operadores de atribuição:** utilizados para atribuir valores a variáveis. Exemplos: = (atribuição simples), += (atribuição com adição), -= (atribuição com subtração), \*= (atribuição com multiplicação), /= (atribuição com divisão), entre outros.
* **Operadores de identidade:** utilizados para comparar a identidade de objetos (se são o mesmo objeto na memória). Exemplos: is (é), is not (não é).
* **Operadores de pertencimento:** utilizados para verificar se um valor está contido em uma sequência. Exemplos: in (está em), not in (não está em).

MATERIAL EM ELABORAÇÃO