

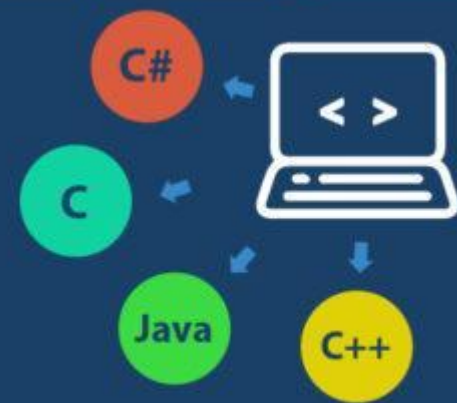
Linguagem de Programação

Professora: Letícia Pieper Jandt

Análise e Desenvolvimento de Sistemas

60 Horas

Python



Python é uma linguagem de programação criada pelo matemático e programador Guido van Rossum em 1991.

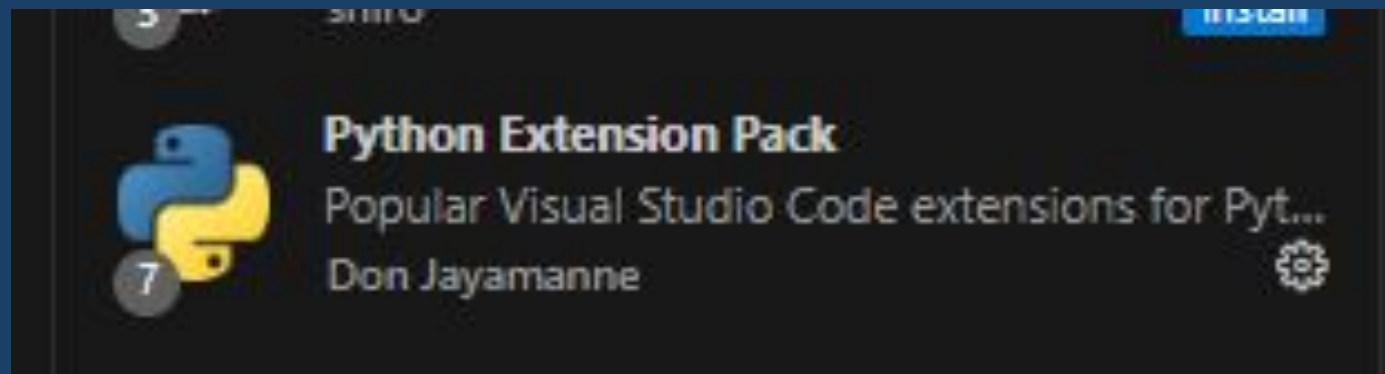
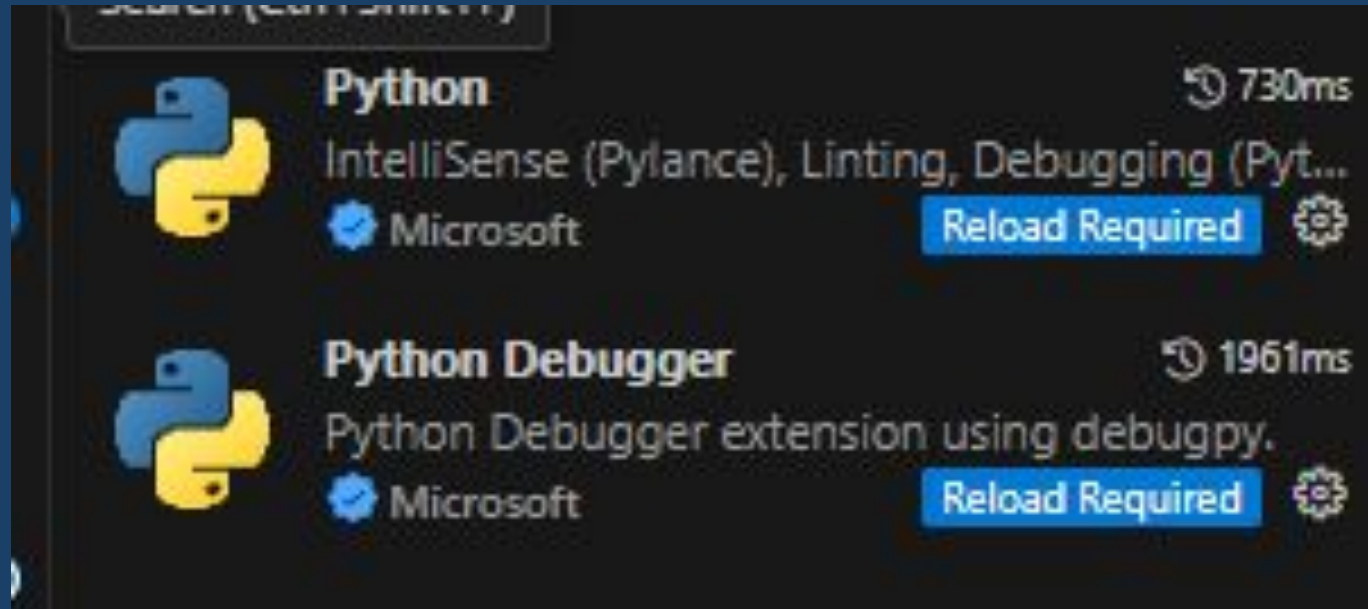
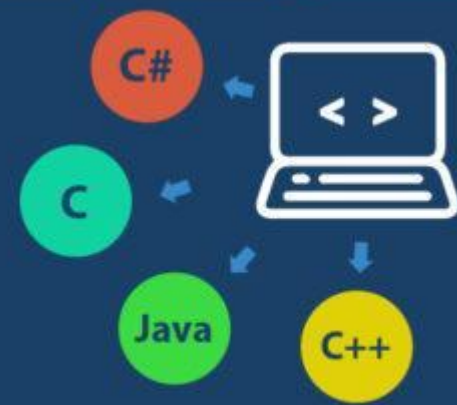
Embora muita gente faça associação direta com cobras, por causa das serpentes píton ou pitão, a origem do nome se deve ao grupo humorístico britânico Monty Python, que inclusive, em 2019, completou 50 anos de existência.

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

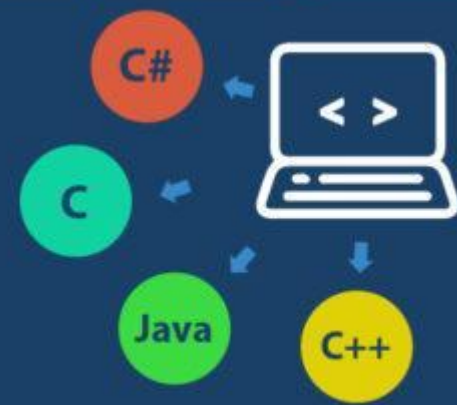


Figura 3 – *Monty Python*, série britânica de humor que deu nome à linguagem.

INSTALANDO



CARACTERÍSTICAS DA LINGUAGEM PYTHON

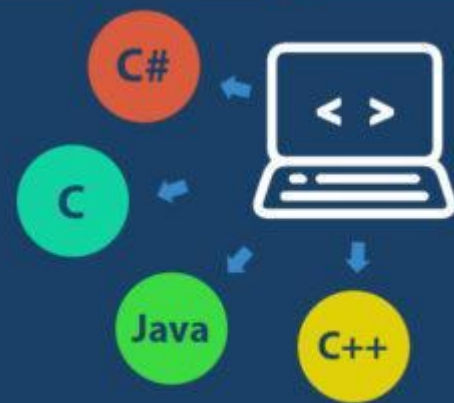


Python é conhecida como uma linguagem de aspectos bastante interessantes e de fácil aprendizagem. O objetivo inicial da linguagem era permitir código enxuto e menos verboso, ou seja, com menos caracteres especiais, menos sintaxes complexas e mais estruturas de código simples.

Por isso, se destaca:

- A facilidade para aprender, ler e compreender;
- Ser multiplataforma;
- Possui modo interativo;
- 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; e
- Uma imensidão de módulos de extensão, os quais permitem expandir o poder da linguagem Python.

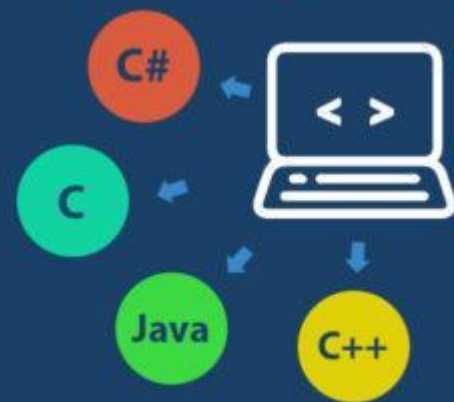
Dica:



Aos que gostam de videoaulas, existem diversas iniciativas para o ensino de Python. Dentre as que se destacam, há o curso Python para Zumbis do professor Fernando Masanori;

<https://www.pycursos.com/python-para-zumbis>

VARIÁVEIS E CONSTANTES



Em Python, podemos declarar uma variável simplesmente definindo um nome para ela e atribuindo-lhe um valor.

Atribuir valores a variáveis nada mais é do que armazenar um valor (ou um conjunto deles) em um determinado endereço de memória referenciando-o por meio de um nome.

VARIÁVEIS E CONSTANTES

```
aluno = "Leticia"
```

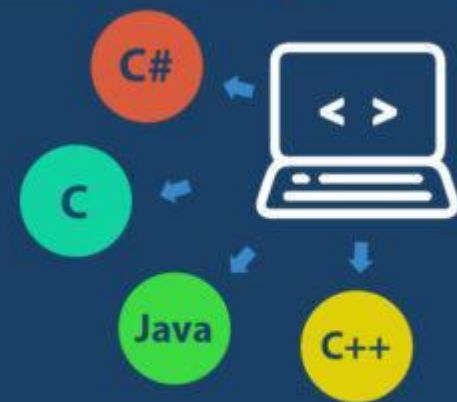
```
periodo_corrente = 10
```

```
media = 9.99
```

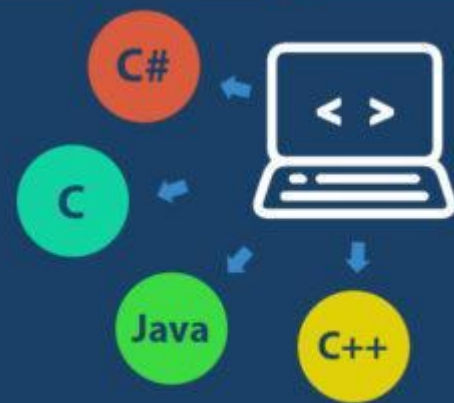
```
presenca = True
```

```
print(aluno, periodo_corrente, media, presenca)
```

*Observe que as variáveis declaradas apresentam nomes significativos para o tipo de informação que desejamos armazenar.



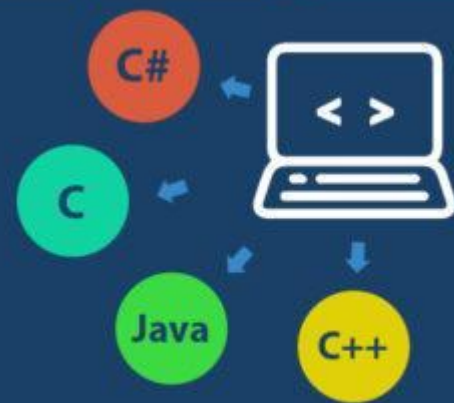
VARIÁVEIS E CONSTANTES



A definição de nomes semânticos para variáveis é sempre recomendada porque eles contribuem com a documentação do programa.

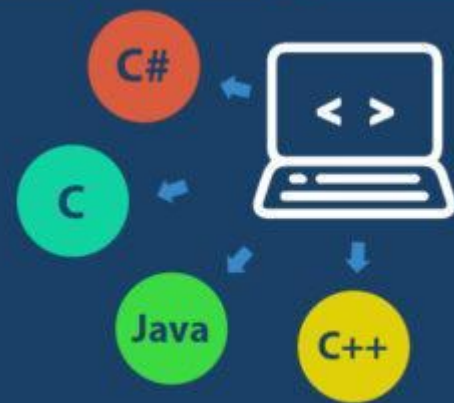
Por inúmeras vezes, vemos alunos declarando nomes de variáveis que não apresentam o menor significado para a informação que está sendo armazenada.

VARIÁVEIS E CONSTANTES



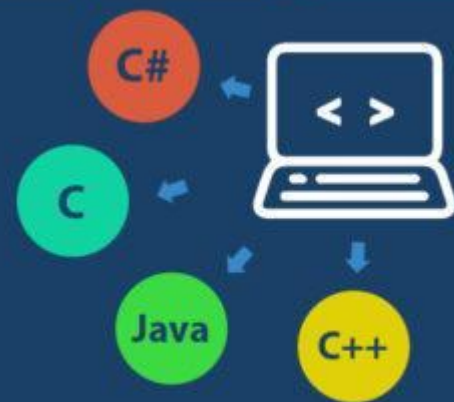
Por exemplo, se as variáveis `aluno`, `periodo_corrente`, `media` e `presenca` fossem substituídos por `x`, `y`, `z` e `w`, respectivamente, e alguém lhe dissesse “Em `x`, é armazenada a string Alan Turing, em `y` o número 10, em `z` o número 9.98 e em `w` o valor False.” você conseguiria deduzir que se trata do nome de um aluno, do seu período corrente ou demais valores? Certamente não.

VARIÁVEIS E CONSTANTES



Para o interpretador Python, qualquer coisa que seja delimitada por um par de aspas simples ou dupla é considerada uma string. Por exemplo, “Olá”, “3”, ‘3.14’ e ‘Verdadeiro’ são valores do tipo string.

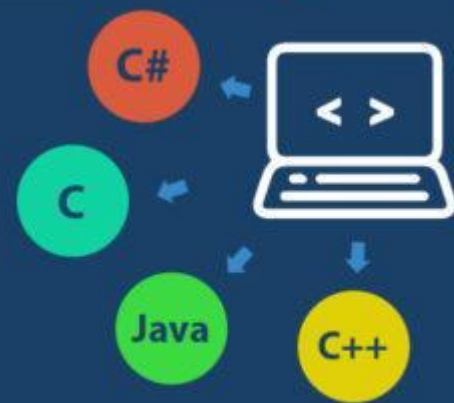
VARIÁVEIS E CONSTANTES



Em Python, convencionou-se que o nome de uma variável deve começar com letras minúsculas como `aluno`, `periodo_corrente`, `saldo`, `nota`, etc.

Caso o nome de uma variável possua mais de uma palavra, deve-se separá-las com um underscore (`_`) como `periodo_corrente` ou `indice_rendimento_academico`.

VARIÁVEIS E CONSTANTES

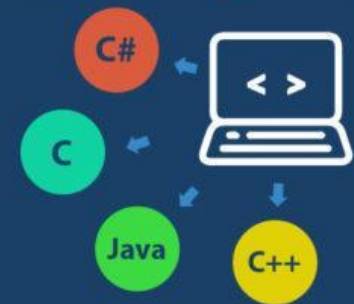


Essa última padronização é chamada de snake case.

Python é case sensitive.

Há nomes de variáveis que não são permitidos em Python. Variáveis cujos nomes começam com números (por exemplo, 2aluno) não são permitidas, embora aluno2 seja um nome válido.

VARIÁVEIS E CONSTANTES

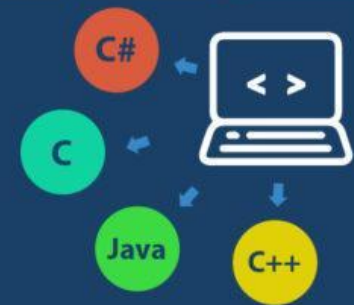


Além disso, deve-se observar também que as palavras reservadas da linguagem Python não podem ser usadas como nomes de variáveis.

Portanto, não é permitido declarar variáveis com os seguintes nomes:

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

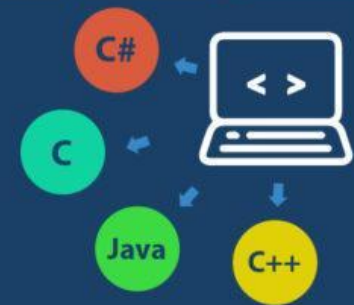
VARIÁVEIS E CONSTANTES



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.

Por exemplo, o valor aproximado de π 3.14159 e, portanto, ele nunca deverá ser modificado

VARIÁVEIS E CONSTANTES



Python *não possui* uma maneira explícita de declarar uma constante, diferentemente de outras linguagens de programação como C, C++ e Java, as quais usam as palavras reservadas *define*, *define* e *final*, respectivamente.

como fazemos constantes em python então?

VARIÁVEIS E CONSTANTES



Uma estratégia para “simular” o comportamento de uma constante seria, por exemplo, declarar uma função para retornar o valor da constante ou simplesmente definir uma variável no início do programa e, em nenhum momento, alterar o seu valor.

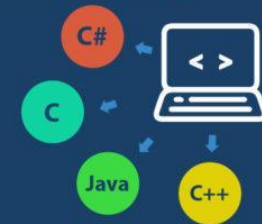
#nome_materia será uma constante. Assim, seu valor não será alterado

```
nome_materia = "Linguagem de programação - python"
print(nome_materia)
```

```
def obtem_nome_materia():
    return "Linguagem de programação - python"
```

```
constante = obtem_nome_materia()
print(constante)
```

TIPOS DE DADOS



As linguagens de programação, em geral, usam o termo “tipo primitivo” para representar a informação em sua forma mais elementar, tais como inteiro, real, lógico ou caractere.

Na documentação oficial da linguagem Python, o termo “tipo primitivo” não é utilizado, mas sim “tipos built-ins” (ou tipos construídos). O motivo é que, para Python, tudo é um objeto.



TIPOS DE DADOS

Python possui vários tipos de dados e os principais deles 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).
- 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.

O valor True pode ser representado por 1 e o False por 0 e, por isso, alguns autores consideram valores do tipo bool como sendo do tipo inteiro.

TIPOS DE DADOS



Diferentemente de outras linguagens de programação como C, C++ e Java, Python possui tipagem dinâmica.

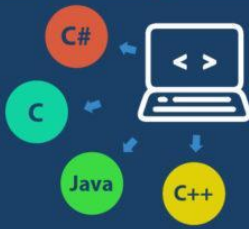
Uma linguagem de programação que possui tipagem dinâmica como Python, PHP ou JS não exige que o programador declare, explicitamente, o tipo de dado que será armazenado por cada variável.

Essa característica permite que, ao longo da execução de um programa, uma mesma variável armazene valores de tipos distintos.



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



Além de realizar as operadores de adição e de multiplicação, também apresentam um comportamento diferente quando são utilizadas strings.

Por exemplo, o comando `print("Olá, " + "mundo!")` concatena a string *Olá com a string mundo!* e resulta em Olá, mundo!

Por outro lado, 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

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

ENTRADA DE DADOS E CONVERSÃO DE TIPOS



Em Python, utiliza-se a função `input()`, que é a forma que o usuário interage com o programa informando dados de entrada.

```
aluno = input("Digite seu nome: ")  
periodo = input("Digite seu período corrente: ")  
ra = input("Digite seu RA: ")  
presenca = input("Informe se você está aqui rss: ")
```

```
print(type(aluno))  
print(type(periodo))  
print(type(ra))  
print(type(presenca))
```

É importante destacar que a função `input()` sempre retornará o valor recebido no formato de uma string, mesmo que ele seja de um tipo de dado diferente como, por exemplo, um número real.

ENTRADA DE DADOS E CONVERSÃO DE TIPOS



Como contornar esse problema? – Funções de conversão `int()`, `float()` e `bool()`, `str()` e `complex()`.

Os nomes das funções de conversão são bastante sugestivos.

```
aluno = input("Digite seu nome: ")
```

```
periodo = int(input("Digite seu período corrente: "))
```

```
altura = float(input("Digite sua altura: "))
```

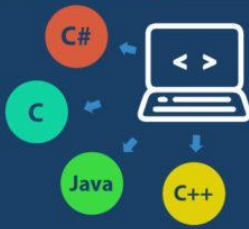
```
presenca = bool(input("Informe se você está aqui rss: ")) #digite 0 ou 1
```

```
print(type(aluno))
```

```
print(type(periodo))
```

```
print(type(altura))
```

```
print(type(presenca))
```

Exemplo de cálculo

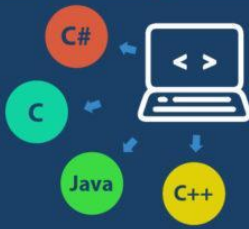
```
base = input("Digite a base: ")  
altura = input("Digite a altura: ")  
area = base * altura  
print(area)
```

Deu certo?

Exemplo de cálculo

Resolvido:

```
base = int(input("Digite a base: "))  
altura = int(input("Digite a altura: "))  
area = base * altura  
print(area)
```



FORMATAÇÃO DE STRINGS



Uma das maneiras mais simples de implementar a formatação de strings é utilizando a *Literal Strings Interpolation* ou, simplesmente, *f-Strings*.

Elas foram incluídas na versão Python 3.6.

A sintaxe das f-Strings é bastante simples e o seu uso garante a incorporação de expressões dentro do texto literal. É importante observar também que elas garantem a execução das expressões em tempo de execução.

FORMATAÇÃO DE STRINGS



Comece com f ou F, contendo expressões envolvidas por um par de chaves {...}, modificadas dentro da string a ser formatada.

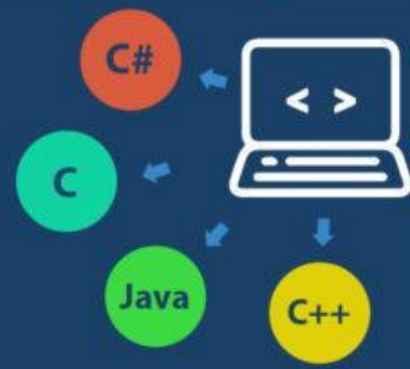
As f-Strings consideram tudo que está fora do par de chaves como sendo um texto literal e, portanto, na saída, o texto será replicado sem nenhuma alteração.

FORMATAÇÃO DE STRINGS



```
from datetime import datetime
ano_atual = datetime.now().year
clube = "Grêmio"
campeonato_mundial = 1
ano_fundacao = 1903
print(f"{clube} possui {campeonato_mundial} títulos mundiais.")
print(f"São {ano_atual - ano_fundacao} anos de existência.")
```

Exercícios



1. Construa um programa no qual um usuário informe a sua estatura em metros e o programa converta-a para centímetros.

(Tente multiplicar por 100)

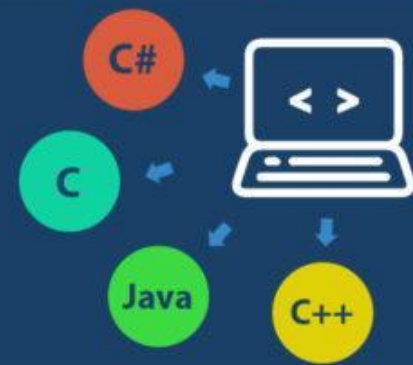
2. Construa um programa que receba do usuário a variação do deslocamento de um objeto (em metros) e a variação do tempo percorrido (em segundo). Ao fim, o programa deve calcular a velocidade média, em m/s, do objeto. (velocidade = deslocamento / tempo)

Exercícios

3. Construa um programa que leia 2 números reais informados pelo usuário. Ao fim, o programa deve calcular e imprimir:

- a. a soma dos dois valores
- b. o produto entre eles

4. Construa um programa que receba do usuário o valor do salário mínimo atual. Na sequência, o programa deve solicitar que o usuário informe o valor do seu salário mensal. Ao fim, o programa deve calcular a quantidade de salários mínimos recebidos pelo usuário.



Exercícios

5. Imagine a situação em que existem 2 copos com sucos de fruta. O primeiro copo está com suco de laranja, enquanto o segundo está com suco de acerola. Você deseja mudar os sucos de copo, isto é, colocar o suco de laranja no segundo copo e o suco de acerola no primeiro copo. No entanto, não é desejável que eles se misturem.

Agora, vamos transformar esta situação em um programa. Nas duas linhas abaixo, nós colocamos o suco de laranja no copo1 e o suco de acerola no copo2:

```
copo1 = "laranja"  
copo2 = "acerola"
```

Continue o programa de modo a transferir o suco de acerola para o copo1 e o suco de laranja para o copo2. Ao fim, imprima as variáveis suco1 e suco2.

Conteúdo complementar

https://www.youtube.com/playlist?list=PLqsF5rntN2nbVwlkRtBbv6gD9vv_fUxxt