

Curso baseado no
livro de Eric
Mattes, **Python
Crash Course**

Python

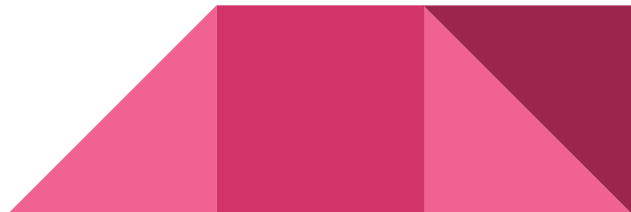
Curso Intensivo

Prof. Cláudio Fleury
Abr-22

Cap.2 - Variáveis e Tipos de Dados

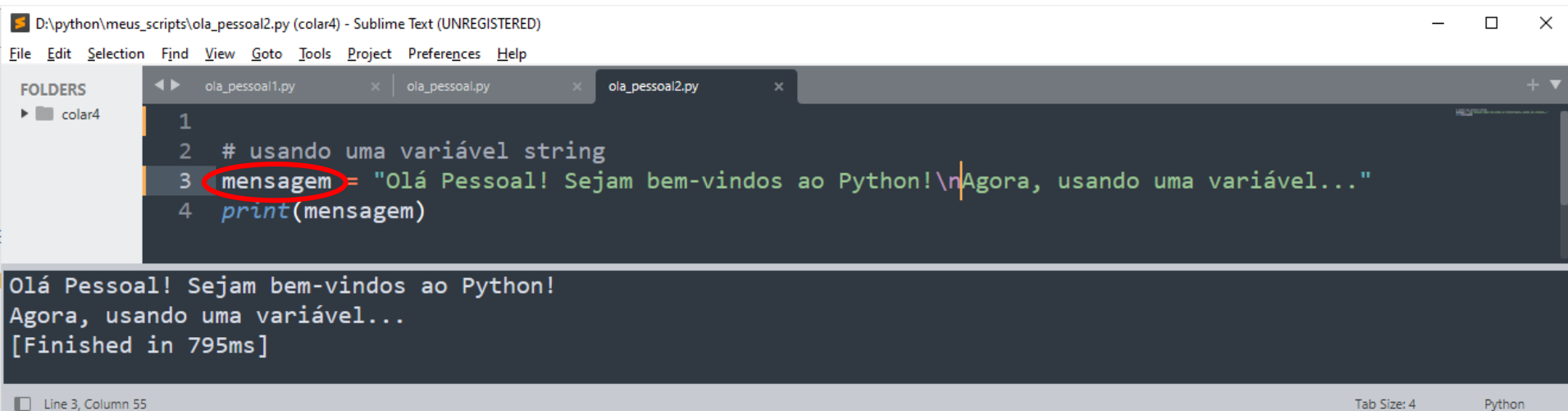
Conteúdo

1. Variáveis
2. Ampliando o Programa “Hello World!”
3. Denominação de Variáveis
4. Strings
 - a. Manipulando Strings
 - b. Usando Variáveis em Strings
 - c. Usando Caracteres de Controle em Strings
 - d. Retirando Espaços em Branco de Strings
5. Números
6. Resumo



Variáveis

- Usando uma variável string denominada 'mensagem' no programa ola_pessoal.py
- Cada variável é conectada a um **valor** via comando de atribuição: **var = valor**
- Esse **valor** é a informação associada à variável e que é recuperada quando a variável é referenciada no programa



The screenshot shows a Sublime Text editor window titled "D:\python\meus_scripts\ola_pessoal2.py (colar4) - Sublime Text (UNREGISTERED)". The editor has three tabs open: "ola_pessoal1.py", "ola_pessoal.py", and "ola_pessoal2.py". The "ola_pessoal2.py" tab is active, showing the following code:

```
1
2 # usando uma variável string
3 mensagem = "Olá Pessoal! Sejam bem-vindos ao Python!\nAgora, usando uma variável..."
4 print(mensagem)
```

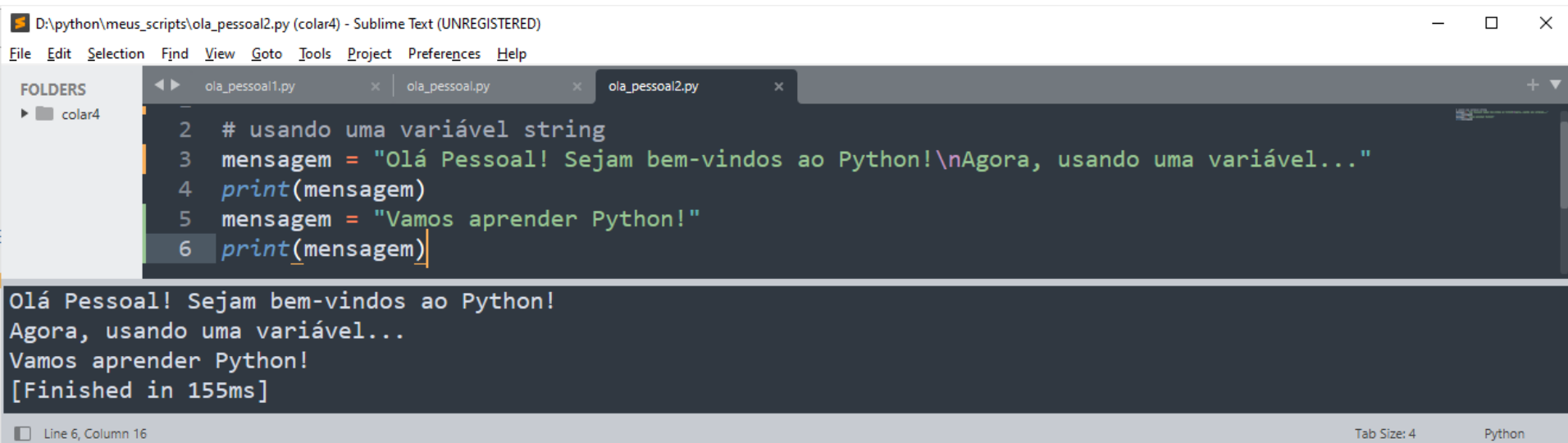
The word "mensagem" in line 3 is circled in red. Below the editor, the output of the program is displayed in a dark console area:

```
Olá Pessoal! Sejam bem-vindos ao Python!
Agora, usando uma variável...
[Finished in 795ms]
```

The status bar at the bottom indicates "Line 3, Column 55", "Tab Size: 4", and "Python".

Ampliando o programa “Hello World!”

- Vamos acrescentar uma segunda mensagem ao programa



The screenshot shows a Sublime Text editor window titled "D:\python\meus_scripts\ola_pessoal2.py (colar4) - Sublime Text (UNREGISTERED)". The editor has three tabs open: "ola_pessoal1.py", "ola_pessoal.py", and "ola_pessoal2.py". The "ola_pessoal2.py" tab is active, showing the following Python code:

```
2 # usando uma variável string
3 mensagem = "Olá Pessoal! Sejam bem-vindos ao Python!\nAgora, usando uma variável..."
4 print(mensagem)
5 mensagem = "Vamos aprender Python!"
6 print(mensagem)
```

Below the editor, a terminal window displays the output of the script:

```
Olá Pessoal! Sejam bem-vindos ao Python!
Agora, usando uma variável...
Vamos aprender Python!
[Finished in 155ms]
```

The status bar at the bottom indicates "Line 6, Column 16", "Tab Size: 4", and "Python".

Denominação de Variáveis

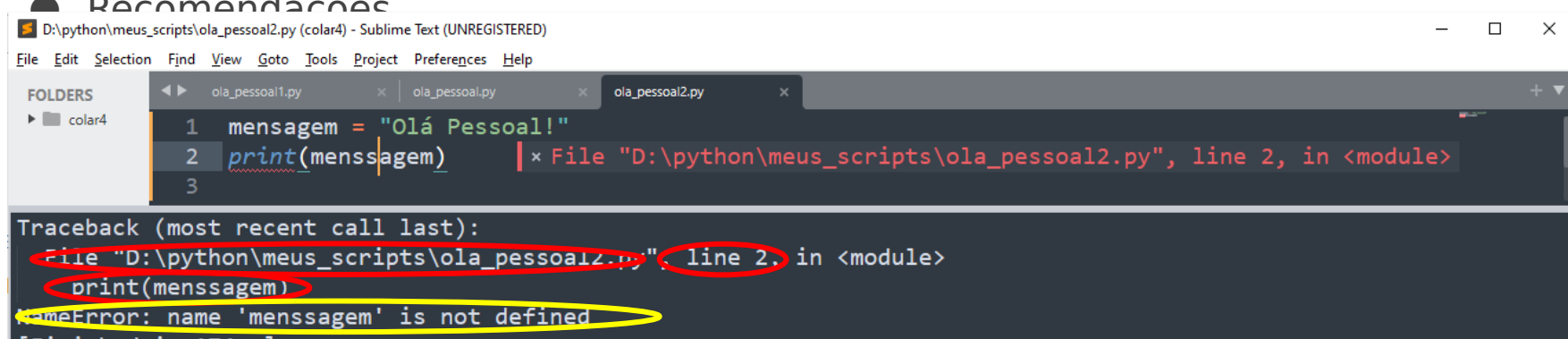
as variáveis Python que usamos até agora foram escritas em letras minúsculas. Não serão gerados erros, se usarmos letras maiúsculas, mas letras maiúsculas em nomes de variáveis têm significados especiais que vamos discutir em capítulos mais adiante.

● Regras

- nomes de variáveis podem conter letras, números e sublinhados
- devem começar com letra ou sublinhado
- espaço em branco e caracteres especiais não são permitidos
- não podem ser palavras chaves da linguagem: and, break, continue, for, if,

...

● Recomendações



The screenshot shows a Sublime Text window titled "D:\python\meus_scripts\ola_pessoal2.py (colar4) - Sublime Text (UNREGISTERED)". The editor has three tabs open, all named "ola_pessoal2.py". The code in the editor is:

```
1 mensagem = "Olá Pessoal!"
2 print(menssagem)
3
```

Below the code, the output area shows a traceback for a NameError:


```
Traceback (most recent call last):
  File "D:\python\meus_scripts\ola_pessoal2.py", line 2, in <module>
    print(menssagem)
NameError: name 'menssagem' is not defined
```

In the error message, "File 'D:\python\meus_scripts\ola_pessoal2.py'", "line 2", and "print(menssagem)" are circled in red. The full error message "NameError: name 'menssagem' is not defined" is circled in yellow.

Exercícios

Escreva um programa para um dos exercícios seguintes. Use o padrão e convenções do Python nos nomes dos arquivos dos programas, usando letras minúsculas e caracteres sublinhado, tais como

`mensagem_simples.py` e **`mensagens_simples.py`**

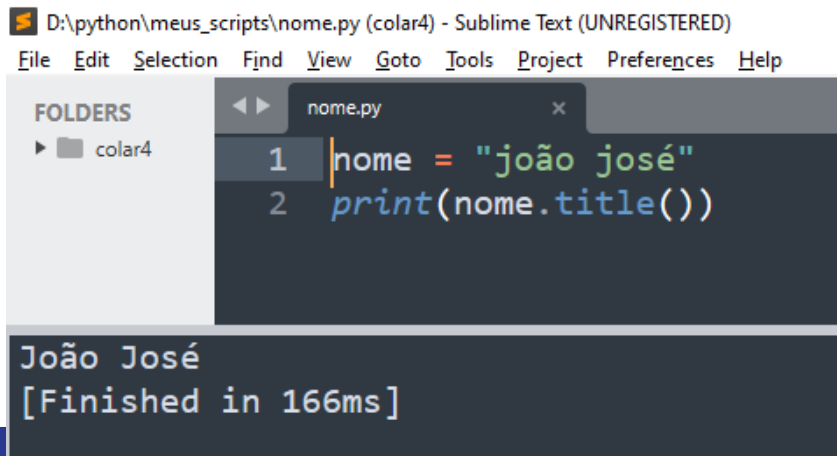
1. Mensagem simples: Atribuir uma mensagem a uma variável e, em seguida, faça a impressão da variável.
 2. Mensagens simples: Atribuir uma mensagem a uma variável, e imprimir a mensagem. Em seguida, altere o valor da variável para uma nova mensagem e imprimir a nova mensagem.
- 

Strings

- String é definida em programação como sendo uma série de caracteres
- Qualquer coisa delimitada por aspas é considerada uma string em Python
- Pode-se usar aspas duplas, “caractere(s)”, ou aspas simples (apóstrofes), ‘caractere(s)’
- Esse é um tipo de dado muito usual
- Exemplos
 - `'Eu disse ao meu amigo, "Python é minha linguagem favorita!"'`
 - `"O nome 'Python' é uma homenagem ao grupo humorístico Monty Python, e não à cobra"`

Manipulando Strings

- Mudando a caixa das letras em uma string
`nome = "joão josé"`
`print(nome.title())`
- Salve o *script*: nome.py
- Execute o *script*



The screenshot shows a Sublime Text editor window titled "D:\python\meus_scripts\nome.py (colar4) - Sublime Text (UNREGISTERED)". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. On the left, a sidebar shows "FOLDERS" with a folder named "colar4". The main editor area displays a file named "nome.py" with two lines of code: `1 nome = "joão josé"` and `2 print(nome.title())`. Below the editor, a console window shows the output: `João José` followed by `[Finished in 166ms]`.

O método **title()** aparece depois da variável na chamada à função **print()**.

Um **método** é uma ação que o Python pode executar num pedaço de dados.

O ponto '.' após a variável (objeto string) em **nome.title()** informa ao Python que o método **title()** deve agir no dado da variável **nome**.

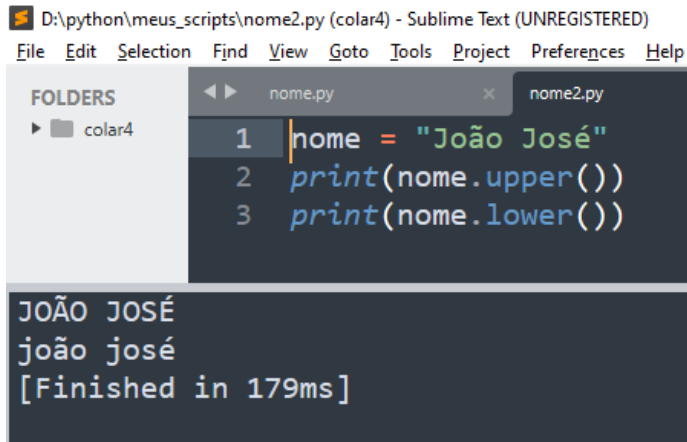
Todo método é seguido por um par de parênteses.

O método **title()** muda a primeira letra de cada palavra na string para maiúscula.

Manipulando Strings

- Mudando a caixa das letras em uma string

```
nome = "joão josé"  
print(nome.upper())  
print(nome.lower())
```



The screenshot shows a Sublime Text editor window titled "D:\python\meus_scripts\nome2.py (colar4) - Sublime Text (UNREGISTERED)". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. On the left, a sidebar shows "FOLDERS" with a folder named "colar4". The main editor area displays a file named "nome2.py" with the following code:

```
1 nome = "João José"  
2 print(nome.upper())  
3 print(nome.lower())
```

Below the editor, the output of the script is shown:

```
JOÃO JOSÉ  
joão josé  
[Finished in 179ms]
```

Vários outros métodos estão disponíveis para lidar com a caixa das letras.

Por exemplo, podemos alterar uma string de caracteres para todas as letras em maiúsculas ou minúsculas.

Usando Variáveis em Strings

- Execute o seguinte snippet

```
prenome    = "joão"  
sobrenome  = "josé"
```

D:\python\meus_scripts\nome_completo2.py (colar4) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

▶ colar4

```
1 prenome    = "joão"  
2 sobrenome  = "josé"  
3 nome       = f"{prenome} {sobrenome}"  
4 mensagem   = f"Olá {nome.title()}! Tudo bem?"  
5 print(mensagem)  
6  
7 # para versões anteriores a 3.6 use o método .format():  
8 mensagem   = "Olá {}! Tudo bem?".format(nome.title())  
9 print(mensagem)
```

```
Olá João José! Tudo bem?  
Olá João José! Tudo bem?  
[Finished in 248ms]
```

Para inserir o valor de uma variável em uma string, coloque a letra **f** antes da abertura de aspas. Colocar chaves em volta do nome(s) de qualquer variável que se queira usar na string.

Estas strings são chamados de strings-f. **F** é para o formato, porque Python formata a string de caracteres, substituindo o nome de qualquer variável entre chaves com o valor da variável.

Usando Caracteres de Controle em Strings

- Execute o seguinte snippet

```
print("Python")
```

D:\python\meus_scripts\caracteres_controle2.py (colar4) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

► colar4

caracteres_controle2.py x

```
1 print("Linguagens:\nPython\nC\nJavaScript")
2 print()
3 print("Linguagens:\n\tPython\n\tC\n\tJavaScript")
4
```

Linguagens:

Python

C

JavaScript

Linguagens:

Python

C

JavaScript

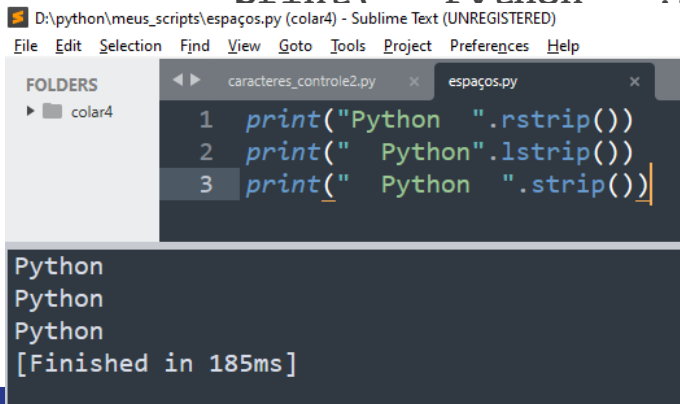
[Finished in 183ms]

Na programação, caracteres de controle referem-se a qualquer caractere não-imprimível, tais como símbolos de retro espaço, tabulação e nova-linha.

Retirando Espaços em Branco de Strings

- Retirando espaços em branco à:

- Direita: `rstrip()`
 - Esquerda: `lstrip()`
 - Direita e Esquerda: `strip()`
- ```
print("Python ".rstrip())
print(" Python".lstrip())
print(" Ppython ".strip())
```



The screenshot shows a Sublime Text editor window with the file path `D:\python\meus_scripts\espaços.py (colar4) - Sublime Text (UNREGISTERED)`. The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The left sidebar shows a 'FOLDERS' panel with 'colar4' expanded. The editor has two tabs: 'caracteres\_controle2.py' and 'espaços.py'. The 'espaços.py' tab is active and contains three lines of Python code:

```
1 print("Python ".rstrip())
2 print(" Python".lstrip())
3 print(" Ppython ".strip())
```

The output console at the bottom shows the results of these commands:

```
Python
Python
Python
[Finished in 185ms]
```

Espaços em branco extra pode ser confuso nos programas. Para os programadores '**python**' e '**python**' parecem ser as mesmas coisas.

Mas para um programa, eles são duas diferentes seqüências de caracteres...

O recurso de **realce de sintaxe** do editor deve ajudá-lo a identificar erros de sintaxe rapidamente enquanto se escreve seus programas...

# Exercícios

Salve cada um dos exercícios a seguir num arquivo separado com nome semelhante a nome\_caixa.py.

- 1. Mensagem pessoal:** Use uma variável para representar o nome de uma pessoa e imprimir uma mensagem para essa pessoa. A sua mensagem deve ser simples, tal como: 'Olá José, você gostaria de aprender um pouco sobre Python hoje?'
- 2. Caixa de Nomes:** Use uma variável para representar o nome de uma pessoa e, em seguida, imprimir o nome da pessoa em letras minúsculas, letras maiúsculas e somente as primeiras letras das palavras em maiúsculas.
- 3. Frase famosa:** Encontrar uma citação de uma pessoa famosa que você admira. Imprimir o citação e o nome do autor. Sua saída deve ser algo como (incluindo as aspas): "Albert Einstein disse uma vez, 'Uma pessoa que nunca fez uma erro nunca tentou nada de novo.'"
- 4. Frase famosa 2:** Repita o Exercício 2 a 5, mas desta vez, represente o nome da pessoa Frase famosa usando uma variável chamada pessoa\_famosa. Em seguida, componha sua mensagem e o represente com uma nova variável chamada mensagem. Imprima sua mensagem.
- 5. Tirando espaços em branco de Nomes:** Use uma variável para representar o nome de uma pessoa, e inclua alguns caracteres de espaço em branco no início e no final do nome. Certifique-se de você usa cada combinação de caracteres, '\t' e '\n', pelo menos uma vez. Imprimir o nome uma vez, de modo que, os espaços em branco à volta do nome sejam mostrados. Em seguida, imprimir o nome usando cada um das três funções de retirada de espaços em branco: lstrip(), rstrip(), e strip().

# Números

- Inteiro (***int***): números sem ponto decimal
- Real (***float***): números com ponto decimal
  - Erros de representação acontecem em todas as linguagens e são de pouca importância, em geral. Python tenta representar o resultado de forma tão precisa quanto possível, mas às vezes é difícil, dada a forma como os computadores representam números internamente
- Complexo (***complex***): números com parte imaginária e/ou parte real

```
D:\python\meus_scripts\espaços.py (colar4) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS
▶ colar4

caracteres_controle2.py x espaços.py x
1 print(0.2 + 0.1, 3 * 0.1)

0.30000000000000004 0.30000000000000004
[Finished in 170ms]
```

## Operadores Aritméticos:

costumeiros: + - \* /

potenciação: \*\*

divisão inteira: //

Python sempre usa float como padrão para armazenar resultados de operações que envolvam pelo menos um operador float, mesmo que o resultado seja um inteiro.

Sublinhados em números são desprezados no armazenamento:

```
idade_universo = 14_000_000_000
```

# Números

- Atribuição Múltipla
- Constantes

Python não tem um tipo predefinido para constantes, mas os programadores Python usam letras maiúsculas para indicar que uma variável deve ser tratada como uma constante, e portanto, seu valor nunca deve ser alterado:

```
CONEXOES_MAX = 140
```

D:\python\meus\_scripts\aceleracoes.py (colar4) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

▶ colar4

```
caracteres_controle2.py x | espacos.py x | aceleracoes.py x
1 acX, acY, acZ = 10, 20, 30.5
2 print("Acelerações: X =", acX, " Y =", acY, " Z =", acZ)
```

```
Acelerações: X = 10 Y = 20 Z = 30.5
[Finished in 168ms]
```

# Zen do Python

- Coleção de 19 princípios orientadores, na forma de poema, com uma série de aforismos, para se escrever programas de computador que influenciam o design da linguagem de programação Python
- Na programação, as pessoas resolvem problemas
- Os programadores sempre respeitam soluções bem projetadas, eficientes e até mesmo bonitas
- Se você tem uma escolha entre uma solução simples e uma complexa, e ambas funcionam, então use a solução simples
- Mesmo quando o código é complexo, torne-o legível

**Aforismo** é um gênero textual, ou uma obra deste gênero, caracterizado por frases breves que definem um preceito moral ou prático.

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```



# Resumo

- Variáveis devem ter nomes curtos e descritivos
  - Regras para denominação de variáveis
  - Strings e seus métodos para mudar caixa e retirar espaços em branco
  - Strings-f para formatação de strings com variáveis
  - Caracteres de controle em Strings
  - Números: inteiro, real e complexo
  - Zen do Python
- 