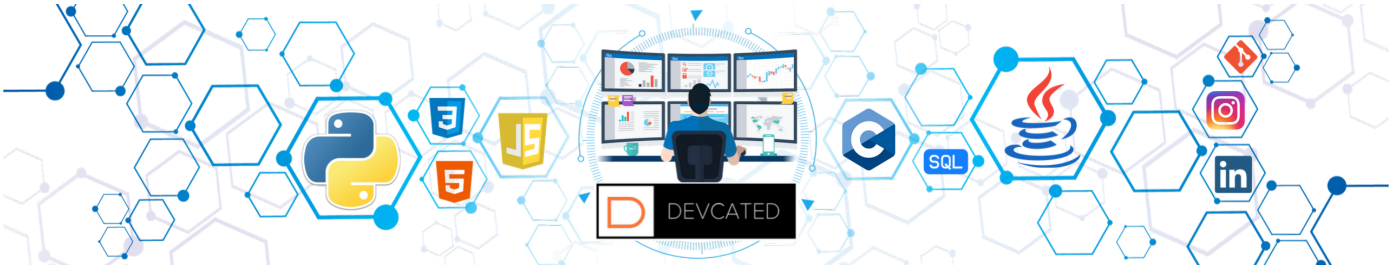


Curso Python Aula 01 - História da Linguagem Python

[(https://www.linkedin.com/in/marcosmapl)](https://www.linkedin.com/in/marcosmapl) [(https://www.youtube.com/)](https://www.youtube.com/)

Marcos Avner P. de Lima

marcos.lima@icomp.ufam.edu.br (mailto:marcos.lima@icomp.ufam.edu.br)

Roteiro

História da Linguagem Python

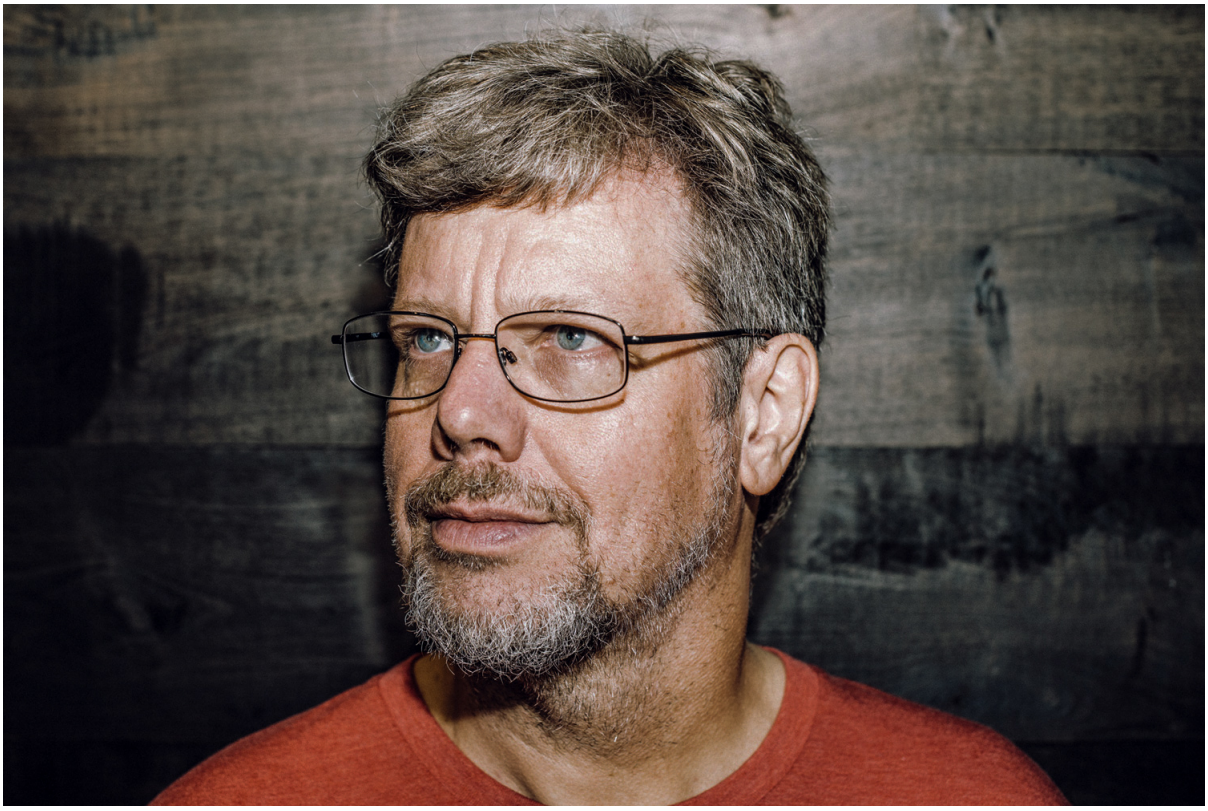
Python é uma linguagem de programação de *alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte*.

- *Alto nível* - É quando uma linguagem tem um nível de abstração relativamente elevado, longe do código de máquina e mais próximo à linguagem humana.
- *Interpretada* - É quando uma linguagem de programação em que o código fonte não é executado diretamente pelo sistema operacioanal e sim por um programa de computador chamado interpretador.
- *Scripts* - São programas escritos para um sistema de tempo de execução especial que automatiza a execução de tarefas que poderiam alternativamente ser executadas por um operador humano.
- *Paradigma Imperativo* - É um paradigma de programação que descreve a computação como ações, enunciados ou comandos que mudam o estado (variáveis) de um programa.
- *Paradigma Orientado a Objetos* - É um paradigma de programação baseado na composição e interação entre diversas unidades chamadas de *objetos*.

- *Funcional* - É um paradigma de programação que trata a computação como uma avaliação de funções matemáticas e que evita estados ou dados mutáveis.
- *Tipagem* - Podemos categoriza as linguagens de programação, quanto ao forma de tipagem:

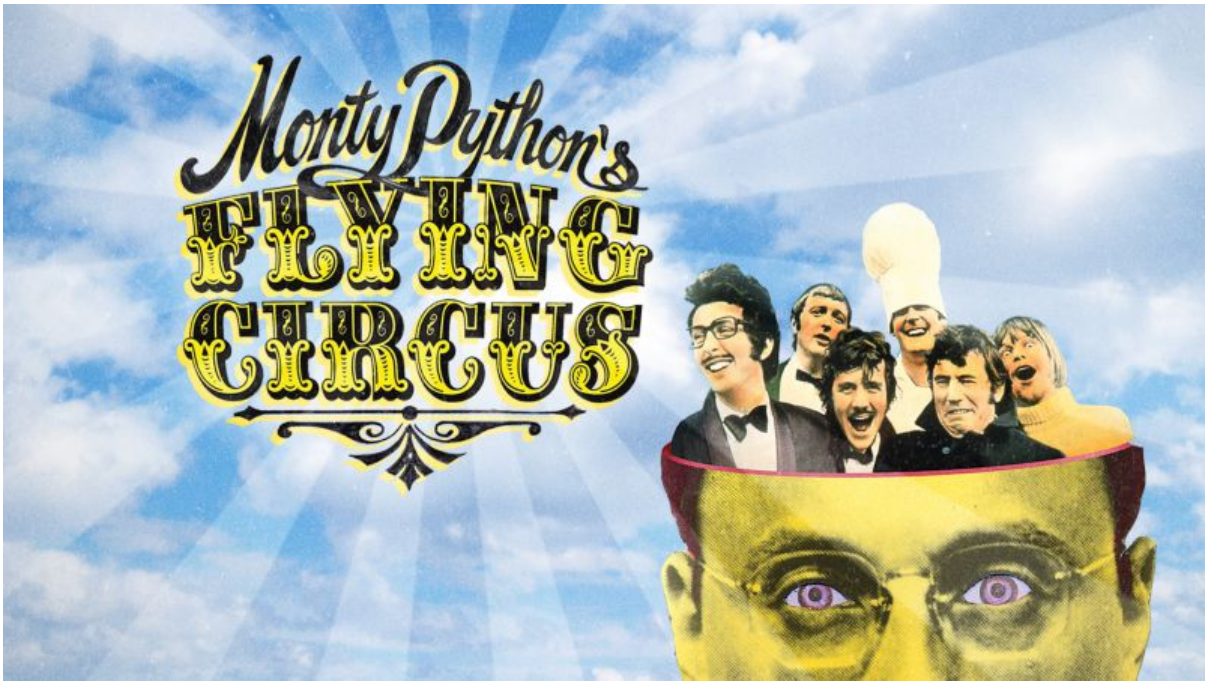
    1. Forte: Há verificação dos tipos de dados utilizados em tempo de execução ou compilação. Ex: Python e Java
    2. Fraca: Não há verificação dos tipos de dados utilizados em tempo de execução ou compilação. Ex: JavaScript e PHP
    3. Estática: A linguagem exige que sejam declarados os tipos de dados utilizados, e uma vez definido não são possíveis alterações em tempo de execução. Ex: C# e Java
    4. Dinâmica: Não a necessidade de declarar os tipos de dados utilizados e temos a possibilidade de alterar o tipo da nossa variável em tempo de execução. Ex: JavaScript e Python.

Foi lançada por Guido van Rossum em 1991. Foi projetada com a filosofia de prioriza a **legibilidade do código** sobre a velocidade ou expressividade.



Devido às suas características, ela é principalmente utilizada para processamento de textos, dados científicos e criação de CGIs para páginas dinâmicas para a web. Foi considerada pelo público a **4º Tecnologia mais popular**, a **2ª linguagem "mais amada"** e a **1ª linguagem "mais buscada"**, de acordo com uma pesquisa conduzida pelo site Stack Overflow em 2019 (https://insights.stackoverflow.com/survey/2019#most-loved-dreaded-and-wanted), além e está entre as **5 linguagens mais populares**, de acordo com uma pesquisa conduzida pela RedMonk em Junho de 2019 (https://redmonk.com/sogrady/2019/07/18/language-rankings-6-19/).

O nome Python teve a sua origem no grupo humorístico britânico **Monty Python**, criador do programa **Monty Python's Flying Circus**.



Bonûs - História da Linguagem Python por Guido Van Rossum

Link: https://www.youtube.com/watch?v=J0Aq44Pze-w (https://www.youtube.com/watch?v=J0Aq44Pze-w)

Preparando o ambiente

Antes de começar a programar em Python, você precisara ter acesso ao **Interpretador** Python. Há divesas opções para isso:

- **Interpretadores Online:**
  - Python.org Online Console (https://www.python.org/shell).
  - Repl.it (https://repl.it/languages/python3).
  - Online GBD (https://www.onlinegbd.com/online_python_compiler).
  - Trinket.io (https://trinket.io/python).
- **Instalando o Interpretador:**
  - **Windows** - Normalmente não possui Python3 instalado, felizmente o processo de instalação é muito simples e envolve apenas baixar o instalador (adequado 32-bit/64-bit) no site da Python Software Foundation (https://www.python.org), executá-lo.
    - Para confirmar a instalação, abra uma **Janela de Comando** (`Windows + R` , digite `cmd` e tecle `ENTER`).
    - Na **Janela de Comando** digite `python3` e tecle `ENTER` .
  - **Linux** - Há uma grande chance do seu sistema já possuir Python instalado, mas provavelmente não será a versão mais atual.
    - Para verificar qual versão instalada basta abrir uma **Janela Terminal** (`Ctrl+T`) e digite `python --version` e tecle `ENTER` .
    - Se a versão exibida for diferente de **Python 3.7.x** você não possui a versão mais atual lançada até a data em que escrevo esta aula e terá que atualizá-la.
    - Para atualizar sua versão recomendo este tutorial Instalando Python - RealPython (https://realpython.com/installing-python).
  - **Mac OS** - Uma abordagem bastante recomendada pela comunidade é a instalação utilizando o gerenciador de pacotes **Homebrew** conforme o tutorial Instalando Python3 no Mac OS X - Docs Python (https://docs.python-guide.org/starting/install3/osx/).

- **Plataforma Anaconda** - Plataforma de Data Science e Machine Learning em Python/R. Disponível para Linux, Windows e Mac OS X. Download e instalação pelo site da plataforma Anaconda (https://www.anaconda.com/distribuition/).
  - Traz diverso pacotes pré-instalados, além do **Jupyter Notebook** e **Spyder IDE**.
  - Gerenciamento e download de módulos/bibliotecas.
  - Desenvolvimento e treino de modelos de **Machine Learning** utilizando `scikit-learn`, `TensorFlow e theano`.
  - **Análise de dados, escalabilidade e performance** utilizando `dask`, `numPy`, `pandas e numba`.
  - **Visualização** de resultados com `matplotlib`, `bokeh`, `datashader e holoviews`.

Outras Ferramentas

- **Editor de Texto** - Ferramenta apenas para edição de código, em geral bom para aprendizagem e scripts.
  - `Sublime Text 3`.
  - `Atom`.
  - `GNU Emacs`.
  - `Vi/Vim`.
- **Ambiente Integrado de Desenvolvimento (IDE)** - Fornece um conjunto de ferramentas para codificação, compilação, depuração, execução, controle de versão. Podem possuir outras funções como autocomplete, code snipets, syntax highlighting, drag-drop GUI layout, auto formatting, refactoring, database, browsers, etc.
  - `PyCharm`.
  - `Spyder`.
  - `Visual Studio Code`.
  - `PyDev`.
  - `Eclipse`.
  - `IDLE`.
  - `Wing`.
  - `Rodeo`.
  - `Thonny`.

Qual a melhor opção **Editor de Texto** ou **IDE**? Minha recomendação:

- Para Iniciantes, testem as soluções com o menor número de customizações possíveis. "Menos é mais!"
- Se você já usa Editores de Texto para outras tarefas (desenvolvimento web por exemplo), uma das opções dentre os Editores de Texto será mais adequada.
- Se você já é um desenvolvedor de software e é familiarizado com alguma IDE compatível, traga o Python para dentro do seu mundo, assim vai tirar proveito de todo ferramental que você domina.
- Link: 12 Melhores Editores e IDEs para Python (https://www.softwaretestinghelp.com/python-ide-code-editors/amp/)

Executando o Interpretador

Para iniciar o **Interpretador** digite `python3`, na janela do terminal, e aperte ENTER:

```
$ python3.7.3
Python 3.7.3 (default, Mar 27 201, 22:11:17)
[GCC 7.3.0] Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Com isso você instalou o **Interpretador** e já pode começar a trabalhar com Python3. Para finalizar o **Interpretador** digite `quit()` e aperte `ENTER`.

The Zen of Python

The Zen of Python é uma lista de 19 (talvez 20 ~Easter Egg) princípios gerais para desenvolvimento.

In [1]:

```
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!
```

Vamos entender tais princípios:

1. **Beautiful is better than ugly** *(Bonito é melhor que feio)*
    - Em Python, quando existem várias formas para chegar numa solução, escolhemos a **visualmente mais agradável**.
2. **Explicit is better than implicit** *(Explicito é melhor que implicito)*
    - Em Python, sempre preferimos a opção mais **legível**, ou seja, aquela que apresenta todos os elementos para se fazer entender.
3. **Simple is better than complex** *(Simples é melhor que complexo)*
    - Ao desenvolver um código Python, não busque o caminho mais complexo! Tente manter **simples**!
4. **Complex is better than complicated** *(Complexo é melhor que complicado)*
    - Quando a alternativa simples não é possível, fuja da solução complexa.
5. **Flat is better than nested** *(Linear é melhor do que aninhado)*
    - Evite criar estruturas dentro de estruturas ... Opte pela alternativa **linear**, sem aninhá-las.
6. **Sparde is better than dense** *(Esparso é melhor que denso)*
    - Não tente amontoar código!
7. **Readability counts** *(Legibilidade conta)*

- Ao terminar de desenvolver, verifique a **legibilidade** do seu código.

8. **Special cases aren't special enough to break the rules. Athough praticality beats purity** *(Casos especiais não são especiais o bastante para quebrar as regras. Ainda que praticidade vença a pureza)*

   - **Nenhum caso** é tão especial a ponto de passar por cima das regras!

9. **Erros should never pass silently. Unless expliciy silenced** *(Erros nunca devem passar silenciosamente. A menos que sejam explicitamente silenciados)*

   - Silenciar exceções é um erro grave de manuntenção de código, pois pode esconder um erro crítico!

10. **In the face of ambiguity, refuse the temptation to guess** *(Diante da ambiguidade, recuse a tentação de adivinhar)*

    - Mantenha seu código específico, limpo, visualmente agradável e sem ambiguidade.

11. **There should be one - and preferably only one - obvious way to do it. Although that way may note be obvious at first unless you're Dutch** *(Deveria haver um - e preferencialmente apenas um - modo óbvio para fazer algo. Embora esse modo possa não ser óbvio a princípio a menos que você seja "Holandês")*

    - Python defende que **deve** existir apenas uma maneira de se chegar a uma solução do problema. É isso que torna Python **tão simples**.
    - A segunda parte é apenas uma referência ao criador do Python, Guido van Rossum - que é holandês. E por isso para ele aprender algo relacionado a linguagem seria muito mais fácil.

12. **Now is better than never. Although neve is often better than right now** *(Agora é melhor que nunca. Embora nunca frequentemente seja melhor que já)*

    - Primeiramente, não gaste seu tempo demasiadamente com planejamento, as vezes é melhor fazer uma primeira versão e iterar adicionando pequenas melhorias até ter uma versão final.
    - Mas mesmo assim, nunca saia fazendo *qualquer coisa* , apenas pelo fato de se ter algo.

13. **If the implementation is hard to explain, it's a bad idea** *(Se a implementação é difícil de explicar, é uma má ideia)*

    - Se a sua solução é de difícil entendimento, revise-a!

14. **If the implementation is easy to explain, it may be a good idea** *(Se a implementação é fácil de explicar, pode ser uma boa ideia)*

    - Agora, se a solução é simples de ser explicada, ela **pode ser** uma boa ideia.

15. **Namespaces are one honking great idea -- let's do more of those!** *(Namespaces são uma grande ideia - vamos ter mais dessas)*

Fontes:

- What do different aphorisms in The Zen of Python means? (https://www.quora.com/What-do-different-aphorisms-in-The-Zen-of-Python-mean)
- A Brief Analysis of "The Zen of Python" (https://medium.com/@Pythonidaer/a-brief-analysis-of-the-zen-of-python-2bfd3b76edbf)