



Prof. Adriano Silva
adrianovss@gmail.com

adrianovss@gmail.com

Ementa

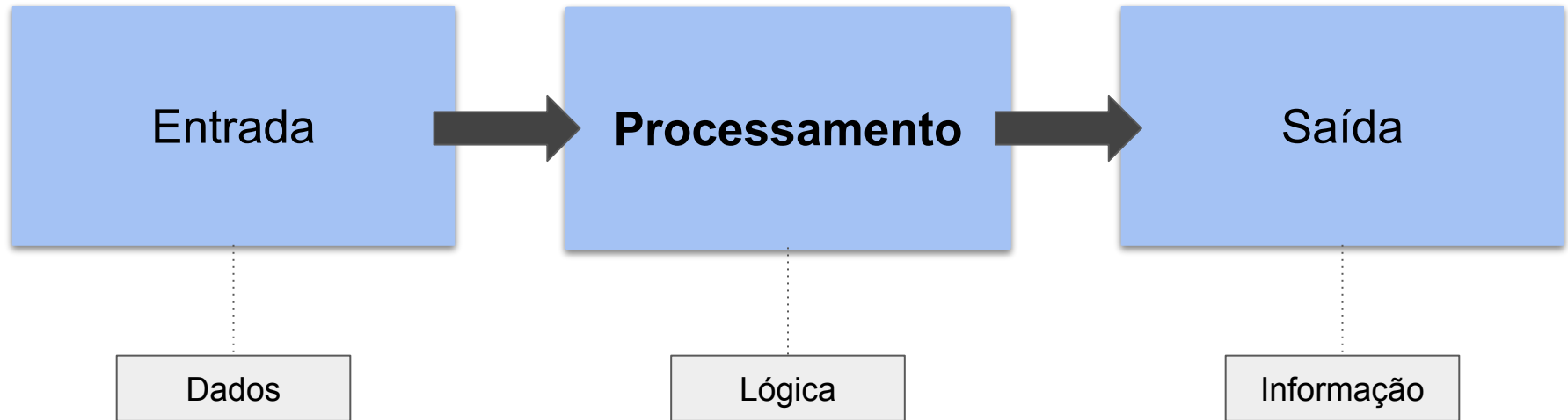
- **Iniciando com Programação:**
 - Algoritmos e Linguagens de Programação
 - Fluxo de um Programa
 - Boas Práticas
- **Iniciando com o Python:**
 - Histórico
 - Instalação
 - Como o código é executado?
 - IDEs / Plataformas
 - Hello World (funções para entrada e saída)

Ementa (continuação...)

- Variáveis
- Tipos de Dados
- Concatenação de Textos
- *Casting*
- Operadores
- Estruturas de Fluxo (Condicionais)



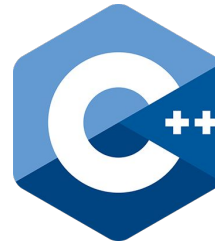
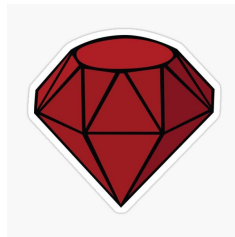
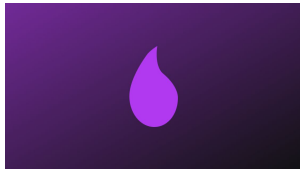
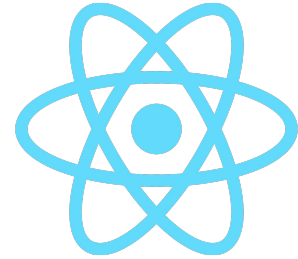
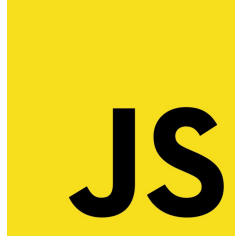
Iniciando com Programação



Algoritmo

- Algoritmo é uma sequência finita de instruções bem definidas e não ambíguas, cada uma das quais devendo ser executadas mecânica ou eletronicamente em um intervalo de tempo finito e com uma quantidade de esforço finita. (Wikipedia)
- Simplificadamente um algoritmo é uma receita, um conjunto de instruções bem definidas para solucionar um problema conhecido.

Linguagens de Programação (+ Soluções)



Boas Práticas

- Code Conventions:
 - <https://peps.python.org/pep-0008/>
- Clean Code
- Nomes Significativos
- Documentação
- Comentários
- Code Review / Tests

Practice





Iniciando com Python

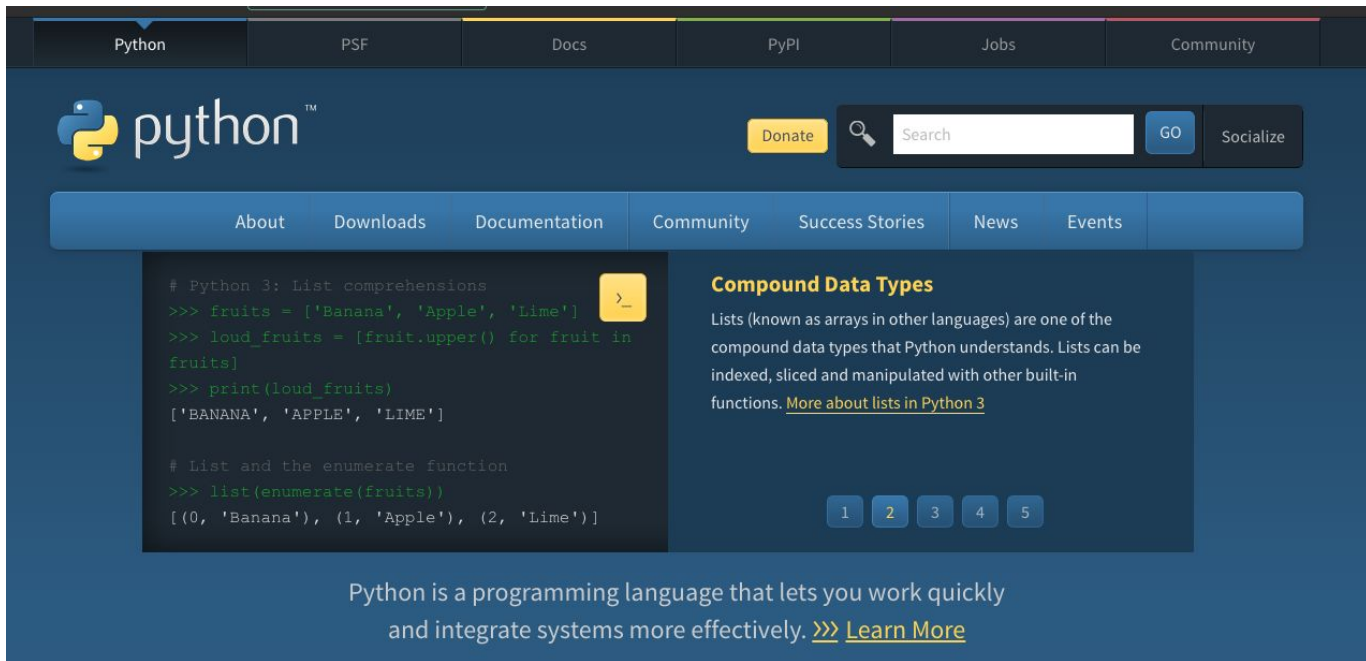
Histórico

- Desenvolvida por Guido van Rossum (década de 90).
- Características:
 - Programação de alto nível.
 - Interpretada e de código fonte aberto.
 - Interativa.
 - Multi-Plataforma / Multi-Paradigma.
 - Sintaxe simples, fácil de aprender e manter.
 - Tipagem forte e dinâmica.
 - Tudo em Python é um objeto: variáveis, funções, etc. Cada objeto possui um ID, tipo e valor.
- Homenagem do grupo de comédia *Monty Python*!



Instalação

- <https://www.python.org>



The screenshot shows the Python.org homepage with a dark blue header and navigation bar. The main content area features a code editor on the left and a text block on the right. The code editor displays Python 3 list comprehensions and the enumerate function. The text block is titled 'Compound Data Types' and explains that lists are one of the compound data types that Python understands. At the bottom, a footer states: 'Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)'.

Python

PSF

Docs

PyPI

Jobs

Community

python™

Donate

Search

GO

Socialize

About

Downloads

Documentation

Community

Success Stories

News

Events

```
# Python 3: List comprehensions
>>> fruits = ['Banana', 'Apple', 'Lime']
>>> loud_fruits = [fruit.upper() for fruit in fruits]
>>> print(loud_fruits)
['BANANA', 'APPLE', 'LIME']

# List and the enumerate function
>>> list(enumerate(fruits))
[(0, 'Banana'), (1, 'Apple'), (2, 'Lime')]
```

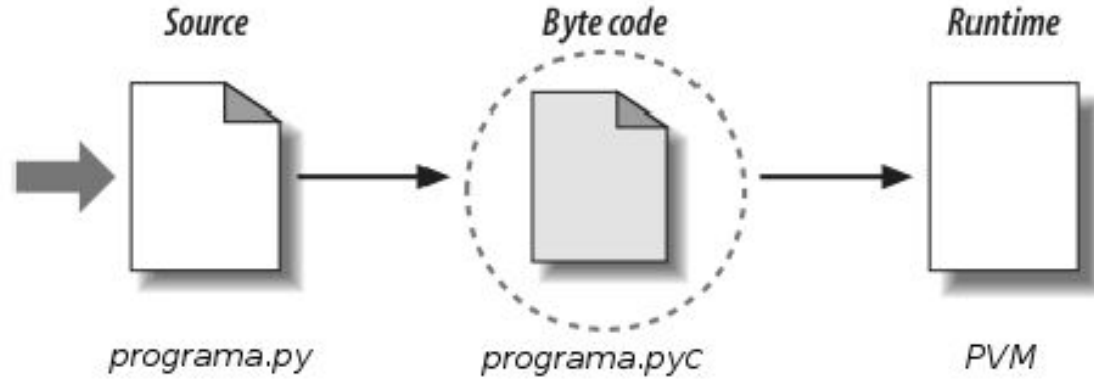
Compound Data Types

Lists (known as arrays in other languages) are one of the compound data types that Python understands. Lists can be indexed, sliced and manipulated with other built-in functions. [More about lists in Python 3](#)

1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)

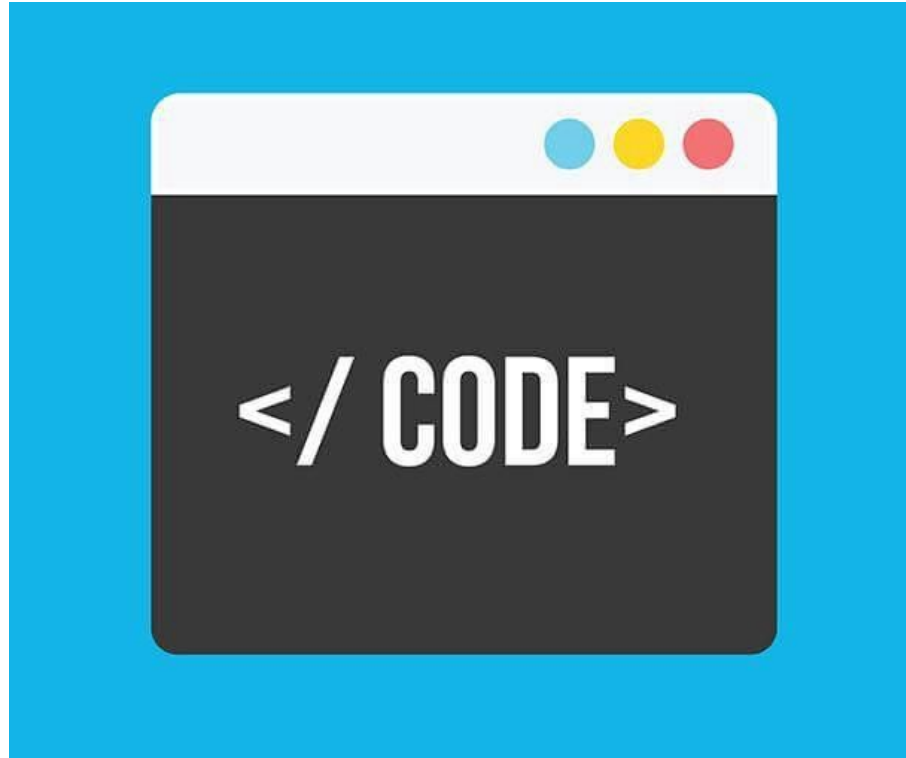
Como é executado?



IDEs / Plataformas



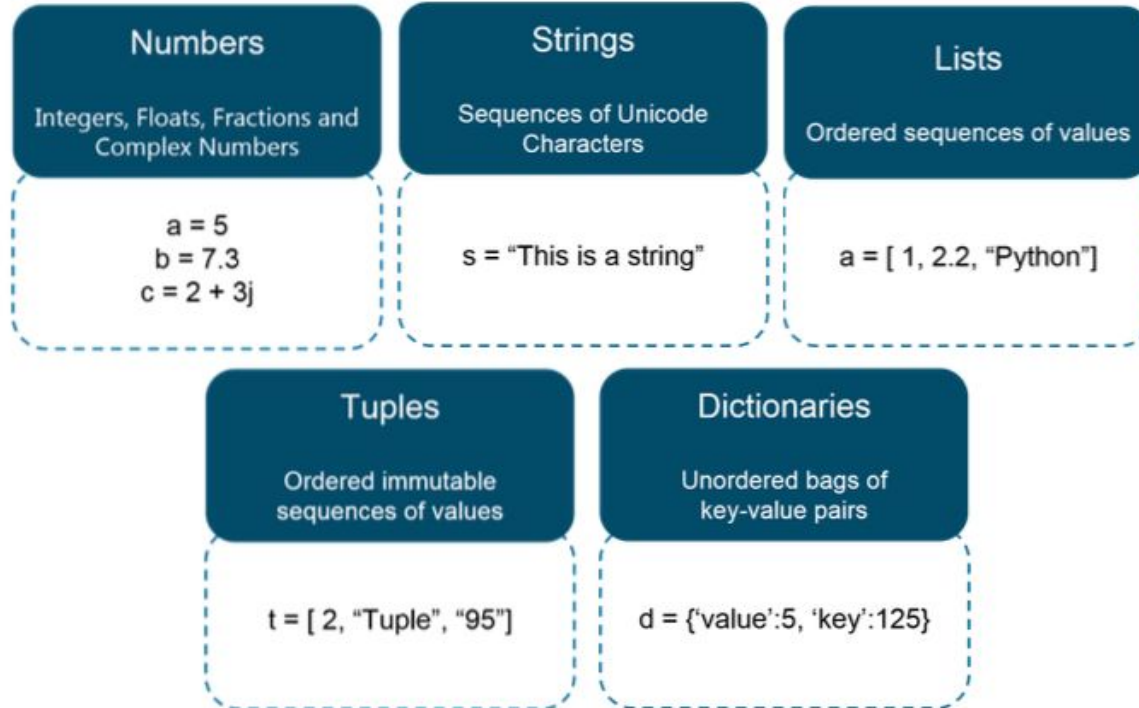
Google Colab + Hello World



Variáveis

- Um objeto / espaço na memória do computador destinado a um dado que é alterado durante a execução de um algoritmo. Definidas por um nome, tipo e valor.
- No Python, as variáveis são dinamicamente tipadas.
- Podem mudar de tipo, simplesmente atribuindo novos valores.
- É possível atribuir um mesmo valor para várias variáveis ao mesmo tempo, assim como múltiplos valores para múltiplas variáveis.

Tipos de Dados



Tipos Numéricos

- **int** (*signed integer*):
 - Também chamado apenas de **integer**, comporta números positivos e negativos sem casas decimais
- **float** (*floating point*):
 - Representa números reais e são escritos com ponto decimal, dividindo um inteiro em partes fracionárias, também podendo ser escrito em notação científica com “e” indicando potência de 10 (ex: 2.5e2)
- **complex**:
 - São escritos por dois valores reais, a parte real e a parte imaginária na forma (real + IMAG J). Neste caso o número i ($\sqrt{-1}$) é designado pela letra j - não são muito utilizados.

Funções mais comuns com tipos numéricos

Função	Descrição
<code>int (x)</code>	Converte x em um inteiro
<code>float (x)</code>	Converte x em um ponto-flutuante
<code>abs (x)</code>	Retorna o valor absoluto de x
<code>exp (x)</code>	Retorna o exponencial de x (e^x)
<code>log (x)</code>	Retorna o logaritmo natural de x (inverso da função exponencial)
<code>pow (x, y)</code>	Retorna o valor de x elevado à potência y
<code>sqrt (x)</code>	Retorna a raiz quadrada de x
<code>round (x, y)</code>	Retorna x arredondado em y casas decimais

Strings

- Python não suporta tipos “char”. Um char em é considerado uma String de tamanho 1.
- Uma String pode ser atribuída de várias formas diferentes.
- Strings iniciam sempre com índice 0.
- Pode se usar operações como slicing ([], [:]), concatenação (+), repetição (*) e *membership* (in) .

Funções e métodos mais comuns com o tipo *string*

Função	Descrição
<code>str (num)</code>	Converte um número em String
<code>len (str)</code>	Retorna o tamanho de uma String
<code>str.count (s)</code>	Retorna a quantidade de conjuntos <i>s</i> presentes na string
<code>str.isalpha ()</code>	Retorna False se a string contiver algum caracter que não seja letras
<code>str.isdigit ()</code>	Retorna False se a string contiver algum caracter que não seja número
<code>str.lower ()</code>	Retorna a string transformada em minúsculos
<code>str.upper ()</code>	Retorna a string transformada em maiúsculos
<code>str.replace (old, new)</code>	Substitui uma porção da string por outro conteúdo
<code>str.strip ()</code>	Retira espaços em branco no começo e no fim da string
<code>str.title ()</code>	Retorna a string capitalizada (iniciais em maiúscula)
<code>str.split (delimiter)</code>	Separa uma string conforme um delimitador. É o inverso do <code>join()</code>
<code>str.join (sequence)</code>	Junta cada item da string com um delimitador especificado. É o inverso do <code>split()</code> .

Além disso...

- Boolean (***bool***): verdadeiro / falso.
 - Na lógica computacional, podem ser considerados como 0 ou 1.
- Utilize a função ***type()*** para identificar o tipo de uma variável (de acordo com o seu conteúdo).

Concatenação de Textos / *Casting*

- +
- ,
- Interpolação

Operadores Aritméticos

Operador	Descrição	Exemplo
+	Adição	$10 + 15 = 25$
-	Subtração	$25 - 15 = 10$
*	Multiplicação	$10 * 3 = 30$
/	Divisão	$30 / 4 = 7.5$
//	Parte inteira da divisão	$30 // 4 = 7$
%	Módulo (Resto da divisão)	$30 \% 4 = 2$
**	Exponenciação (Potência)	$3^{**4} = 3 * 3 * 3 * 3 = 81$

Exercícios



Operadores Relacionais

Operador	Descrição	Exemplo
<code>==</code>	Igual	<code>10 > 15</code> é falso; <code>'Python' == 'python'</code> é falso
<code>!=</code>	Diferente	<code>5 != 7</code> é verdadeiro, <code>'Python' != 'python'</code> é verdadeiro
<code>></code>	Maior	<code>5 > 5</code> é falso; <code>5 > 2</code> é verdadeiro
<code><</code>	Menor	<code>7 < 12</code> é verdadeiro; <code>7 < 5</code> é falso
<code>>=</code>	Maior ou igual	<code>5 >= 5</code> é verdadeiro; <code>5 >= 6</code> é falso
<code><=</code>	Menor ou igual	<code>5 <= 5</code> é verdadeiro; <code>5 < 7</code> é verdadeiro

Operadores Lógicos

Operador	Descrição
and	Retorna True se os dois operandos forem True
or	Retorna True se um dos operandos for True
not	Negativa ou reverte a estado do operando

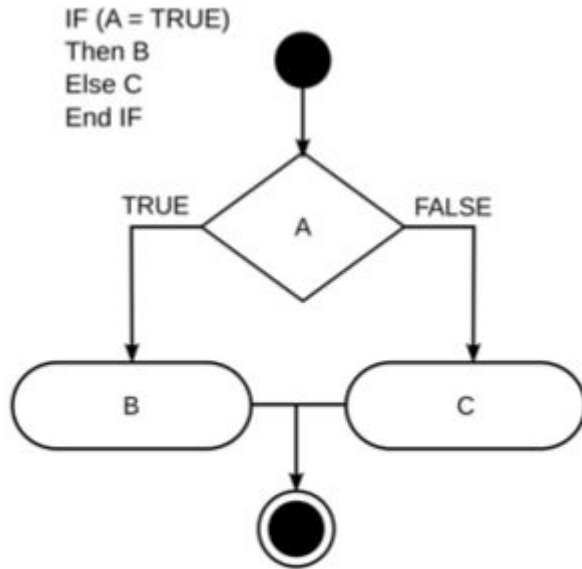
Operadores de Membro

Operador	Descrição
in	True se encontra a expressão pesquisada

Operadores de Identidade

Operador	Descrição
is	True se os operadores apontarem para o mesmo endereço de memória

Estruturas de Fluxo (Condicionais)



- Estruturas Condicionais são utilizadas para decidir qual fluxo de execução deverá ser tomado pelo programa, dada uma determinada condição ou um conjunto de condições.
- São representadas em Python, assim como em outras linguagens pelas instruções:
 - *if*
 - *elif*
 - *else*

Exercícios





Facens

AQUI TEM ENGENHARIA