

PYTHON PARA INICIANTEs

MANOELA KOHLER

www.linkedin.com/in/manoelakohler

APRESENTAÇÃO

Engenheira de Computação – PUC-Rio

Mestre em Engenharia Elétrica (Métodos de Apoio à Decisão) – PUC-Rio

Doutora em Engenharia Elétrica (Métodos de Apoio à Decisão) – PUC-Rio

Professora do CCE – PUC-Rio – Pós-graduação Lato Sensu:

- R e Python
- Inteligência Artificial
- Data Mining
- Machine Learning
- Redes Neurais Artificiais e Deep Learning

Pós-graduação Stricto Sensu

- Computação Evolucionária

Pesquisadora e desenvolvedora no Laboratório de Inteligência Computacional Aplicada

PROGRAMA

First Things First:

- Introdução
- Filosofia
- Ferramentas para análise de dados
- Características
- IDEs

Introdução ao Python

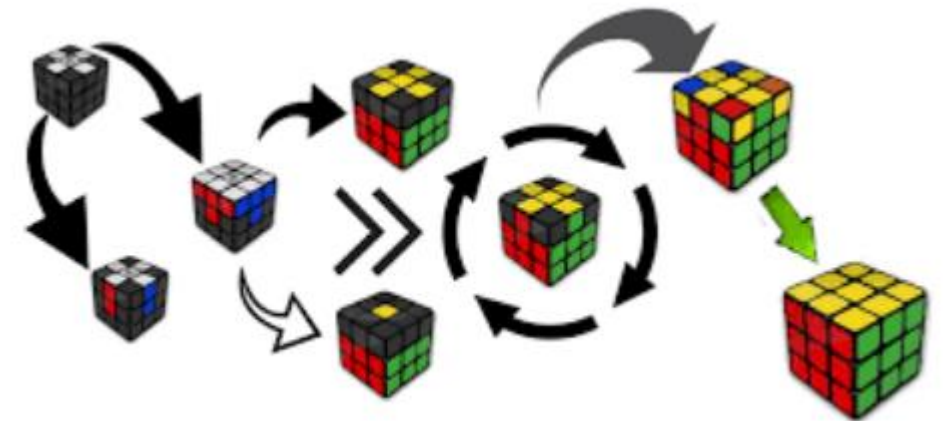
- Conceitos básicos
- Criação de ambientes
- Variáveis
- Tipos de dados
- Operações
- Estruturas condicionais
- Estruturas de repetição
- Bibliotecas: instalação e utilização
- Funções

INTRODUÇÃO

Segundo a filosofia a lógica é a parte da filosofia que trata das formas do pensamento em geral e das operações intelectuais que visam à determinação do que é verdadeiro ou não.

Já quando pensamos em conceitos no universo da tecnologia da informação a lógica é a organização e planejamento das instruções, assertivas etc. em um algoritmo, a fim de viabilizar a implantação de um programa.

Programar nada mais é do que a organização coesa de uma sequência de instruções voltadas à resolução de um problema de forma lógica.



ALGORITMOS

Algoritmos

Algoritmo, pode ser definido como uma sequência de passos a serem seguidos que produzem um resultado esperado para determinada atividade ou tarefa. Ele pode ser usado em diferentes situações:

- Receitas
- Manuais
- Programação
- Entre outros

ALGORITMOS

Por exemplo, vamos criar um algoritmo para trocar uma lâmpada:

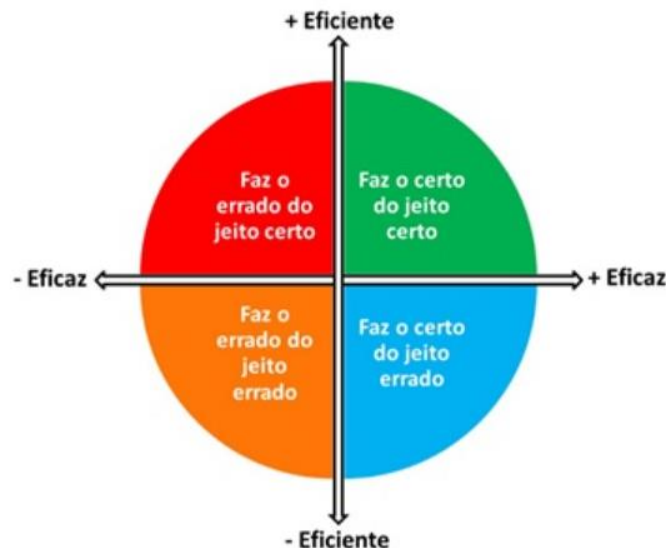
- **Versão 1**

```
Passo 1– Pegar uma lâmpada nova;  
Passo 2 – Pegar a escada;  
Passo 3 – Posicionar a escada embaixo da lâmpada queimada;  
Passo 4 – Subir na escada com a lâmpada nova;  
Passo 5 – Retirar a lâmpada queimada;  
Passo 6 – Colocar a lâmpada nova;  
Passo 7 – Descer da escada;  
Passo 8 – Ligar o interruptor;  
Passo 9 – Guardar a escada;  
Passo 10 – Jogar a lâmpada velha no lixo.
```

ALGORITMOS

Mas quando criamos um algoritmo temos que pensar que ele tem que ser "for dummies". Sendo assim, será que tem como melhorar?

A resposta para essa pergunta sempre vai ser **sim**, sempre tem como melhorar, mas para isso temos que conhecer dois conceitos importantes.



- **Eficácia** — é a qualidade daquilo que cumpre com as metas planejadas, ou seja, uma característica pertencente as pessoas que alcançam os resultados esperados.
- **Eficiência** — é a qualidade daquilo ou de quem é competente, que realiza de maneira correta as suas funções.

ALGORITMOS

Exemplo:

Vamos supor que você esteja desenvolvendo um algoritmo para realizar o lançamento do primeiro foguete brasileiro com destino a lua.

Se seu algoritmo fizer o foguete chegar a lua, ele é um algoritmo eficiente, mas se seu algoritmo fizer o foguete chegar a lua com o menor tempo de processamento, com a menor quantidade de falhas, e assim por diante, esse sim foi um algoritmo eficaz.

ALGORITMOS

Tendo em vista esses conceitos como poderíamos melhorar o nosso algoritmo de trocar a lâmpada?

- **Versão 2**

Passo 1 – Pegar a escada;

Passo 2 – Posicionar a escada embaixo da lâmpada queimada;

Passo 3 – Pegar uma lâmpada nova;

Passo 4 – Subir na escada com a lâmpada nova;

Passo 5 – Retirar a lâmpada queimada;

Passo 6 – Colocar a lâmpada nova;

Passo 7 – Descer da escada;

Passo 8 – Ligar o interruptor;

Passo 9 – Se a lâmpada não acender então volte ao Passo 3;

Passo 10 – Guardar a escada;

Passo 11 – Jogar a(s) lâmpada(s) queimada(s) no lixo.

Nessa versão introduzimos uma etapa de verificação da nova lâmpada, para saber se ela não esta queimada também. Será que podemos continuar melhorando?

ALGORITMOS

- **Versão 3**

```
Passo 1 – Pegar a escada;  
Passo 2 – Posicionar a escada embaixo da lâmpada queimada;  
Passo 3 – Desligar o disjuntor;  
Passo 4 – Subir na escada;  
Passo 5 – Retirar a lâmpada queimada;  
Passo 6 – Descer da escada com a lâmpada queimada;  
Passo 7 – Pegar uma lâmpada nova;  
Passo 8 – Subir na escada;  
Passo 9 – Colocar a lâmpada nova;  
Passo 10 – Descer da escada;  
Passo 11 – Ligar o disjuntor;  
Passo 12 – Ligar o interruptor;  
Passo 13 – Se a lâmpada não acender então volte ao Passo 3;  
Passo 14 – Guardar a escada;  
Passo 15 – Jogar a(s) lâmpada(s) queimada(s) no lixo.
```

Na terceira versão introduzimos algumas etapas de segurança, afinal é importante que tenhamos em mente que um algoritmo deve prever riscos.

FILOSOFIA

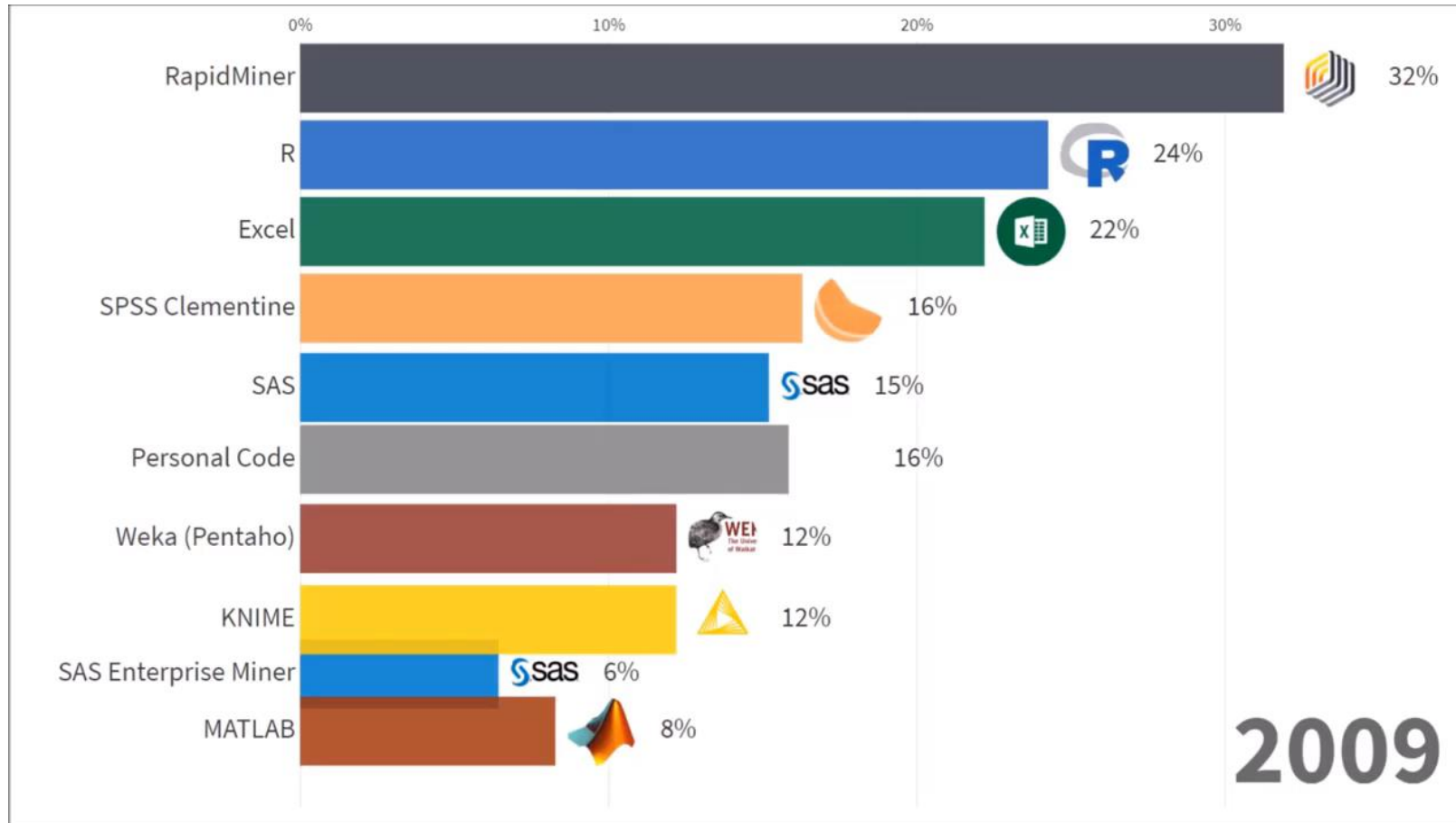
- Totalmente gratuito;
- Usabilidade;
- Orientado à Objetos;
- Poderoso;
- Ferramentas gráficas ponderosas;
- Flexível;
- Vasta comunidade;



FERRAMENTAS DISPONÍVEIS — CIÊNCIA DE DADOS



TOOLS SHARE - KDNUGGETS



Vídeo editado:
2010 - 2019

Vídeo Original:

<https://www.youtube.com/watch?v=pKPaHH7hmv8>

CARACTERÍSTICAS

Python

- Open-Source
- Compatibilidade
- Usado mundialmente pela academia e indústria
- Sempre atualizado e com novas bibliotecas de uso livre
- <https://www.python.org/>

Jupyter

Spyder

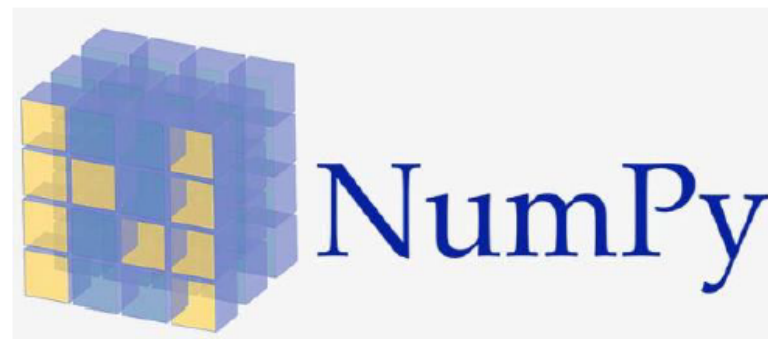
PyCharm

VS Code, etc...

Bibliotecas muito populares em Python



pandas
 $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$





localhost

jupyter DataFrames Last Checkpoint: a day ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Python [default]


Code CellToolbar

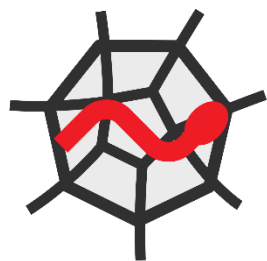
```
In [6]: import pandas as pd
import numpy as np

In [7]: df = pd.DataFrame(data=np.array([[1,2,3], [4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
```

Out[7]:

	A	B	C
0	1	2	3
1	4	5	6





SPYDER

~\MyFirstProject - Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Project explorer

- MyFirstProject
 - Editors.py
 - MyFile.py

Editor - C:\Users\Dora\MyFirstProject\Editors.py

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 mu, sigma = 3, 0.8
5 v = np.random.normal(mu, sigma, 10000)
6
7 plt.hist(v, bins=70, density=2)
8 plt.show()
9
10
11
```

Variable explorer

Name	Type	Size	Value
mu	int	1	3
sigma	float	1	0.8
v	float64	(10000,)	Min: -0.8466367171058615 Max: 6.072040720366579

Variable explorer File explorer Help

IPython console

Console 1/A

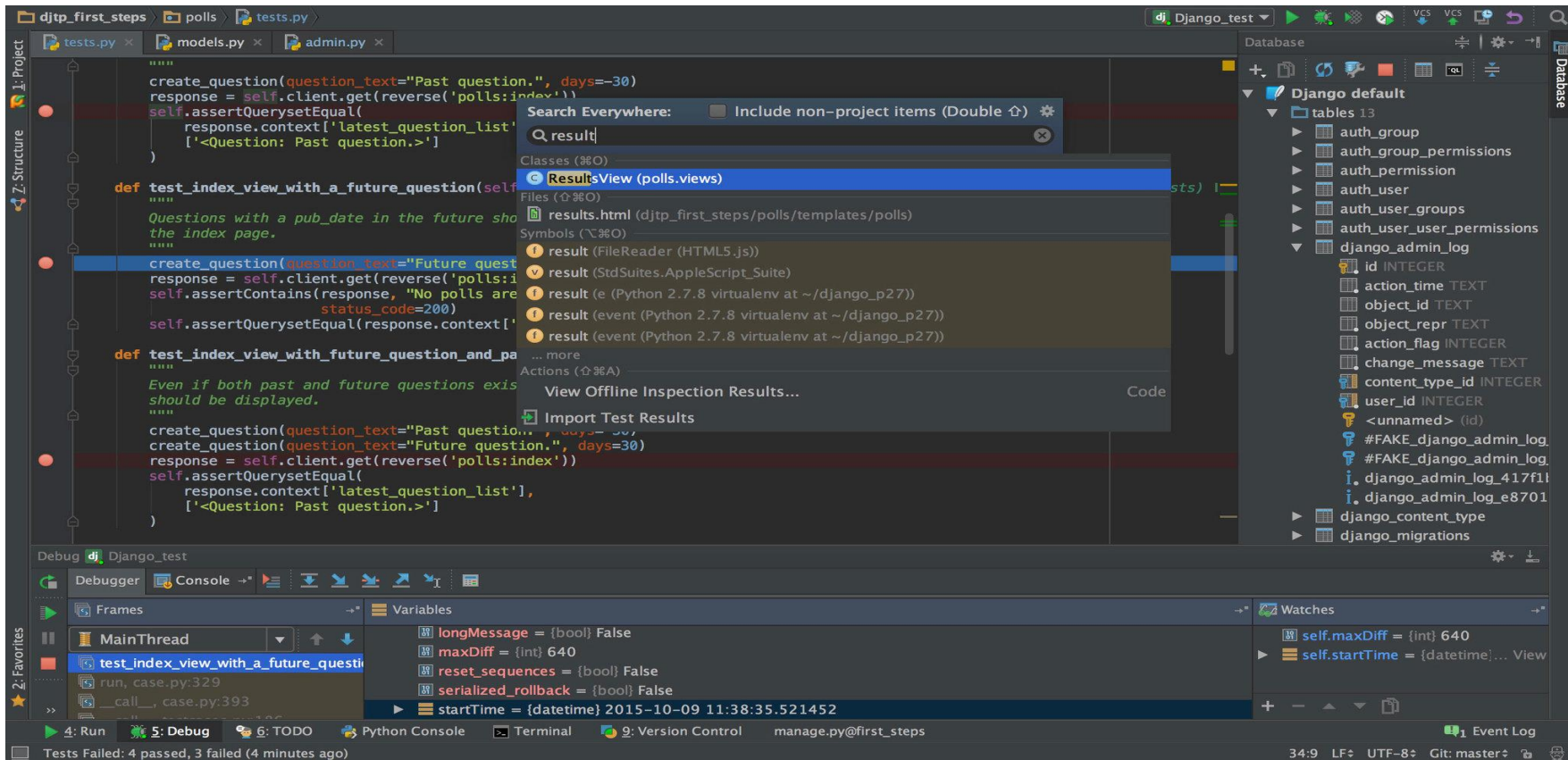
```
In [6]: runfile('C:/Users/Dora/MyFirstProject/Editors.py', wdir='C:/Users/Dora/MyFirstProject')
```

IPython console History log

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 4 Column: 19 Memory: 44 %



PYCHARM





VS CODE

The screenshot displays the Visual Studio Code editor interface. On the left, the Explorer sidebar shows a project structure for 'calculator' with a 'src' directory containing subfolders 'component' and 'logic', and various files including 'App.js', 'App.test.js', 'Button.css', 'Button.js', 'ButtonPanel.css', 'ButtonPanel.js', 'Display.css', 'Display.js', 'calculate.js', 'calculate.test.js', 'isNumber.js', 'operate.js', 'favicon.ico', 'index.css', 'index.js', '.gitignore', 'index.html', 'package.json', and 'README.md'. The main editor window shows the 'App.js' file with the following code:

```
3 import ButtonPanel from './ButtonPanel';
4 import Display from './Display';
5 import React from 'react';
6 import calculate from '../logic/calculate';
7
8 class App extends React.Component {
9   constructor(props) {
10     super(props);
11     this.state = {
12       total: null,
13       next: null,
14       operation: null,
15     };
16   }
17
18   handleClick = (buttonName) => {
19     this.setState(calculate(this.state, buttonName));
20   }
21
22   render() {
23     return (
24       <div className="component-app">
25         <Display value={this.state.next || this.state.total || '0'} />
26         <ButtonPanel clickHandler={this.handleClick} />
27       </div>
28     );
29   }
30 }
```

At the bottom, the Terminal panel shows the output of a 'git status' command:

```
~/Development/calculator master*
> ls
src/ README.md index.html package.json

~/Development/calculator master*
> git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   src/component/App.js
```

The status bar at the bottom indicates the current file is 'App.js' at line 5, column 27, with 2 spaces, UTF-8 encoding, LF line endings, and JavaScript language with ESLint extension.

MELHOR IDE ?

A melhor IDE é aquela que você se sente mais à vontade ao utilizar!



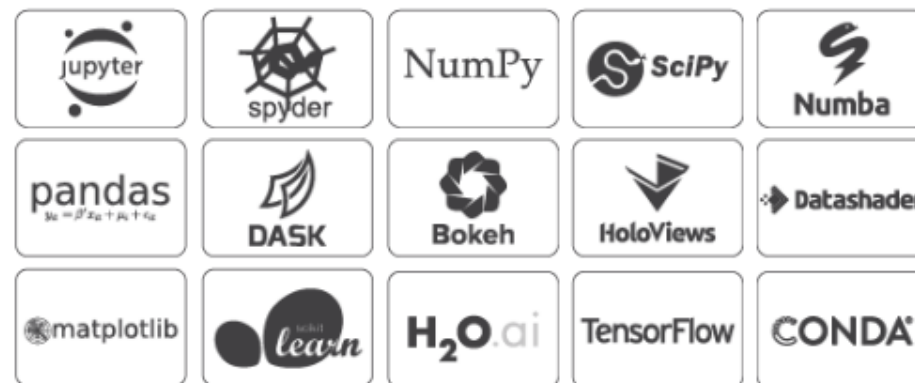
“Anaconda Enterprise gave us everything we needed in a nice, neat package... [Our] data scientists are happy they can concentrate on using the tools rather than maintaining them.”

Will Collins

Analytics Development Leader, National Grid

The open-source [Anaconda Distribution](#) is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 15 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with [Conda](#)
- Develop and train machine learning and deep learning models with [scikit-learn](#), [TensorFlow](#), and [Theano](#)
- Analyze data with scalability and performance with [Dask](#), [NumPy](#), [pandas](#), and [Numba](#)
- Visualize results with [Matplotlib](#), [Bokeh](#), [Datashader](#), and [Holoviews](#)





GOOGLE COLAB

```
AE_digits5.ipynb - Colaboratory x +
colab.research.google.com/drive/1n8gyd9zfRJUK0w6itwCKRoVAGsjuGiG
AE_digits5.ipynb
File Edit View Insert Runtime Tools Help Last edited on May 11, 2019
+ Code + Text
Connect Editing
AutoEncoder

[ ] # Base Digits
from keras.datasets import cifar10
import numpy as np

(x_train, _), (x_test, _) = cifar10.load_data()

[ ] import numpy as np

x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
x_train = np.reshape(x_train, (len(x_train), 32, 32, 3)) # adapt this if using 'channels_first' image data format
x_test = np.reshape(x_test, (len(x_test), 32, 32, 3)) # adapt this if using 'channels_first' image data format

[ ] x_train.shape
(50000, 32, 32, 3)

[ ] from keras.layers import Input, Dense, Conv2D, MaxPooling2D, UpSampling2D
from keras.models import Model
from keras import backend as K

input_img = Input(shape=(32, 32, 3)) # adapt this if using 'channels_first' image data format

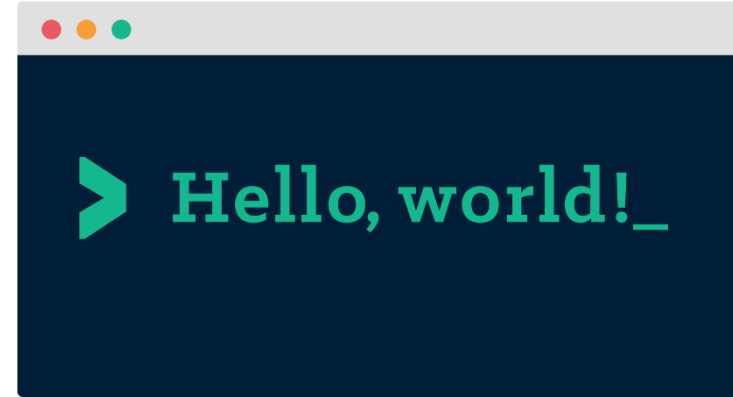
x = Conv2D(16, (3, 3), activation='relu', padding='same')(input_img)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

# at this point the representation is (4, 4, 8) i.e. 128-dimensional

x = Conv2D(8, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
#x = Conv2D(16, (3, 3), activation='relu')(x)
x = UpSampling2D((2, 2))(x)
```


Introdução a Python

CONCEITOS BÁSICOS

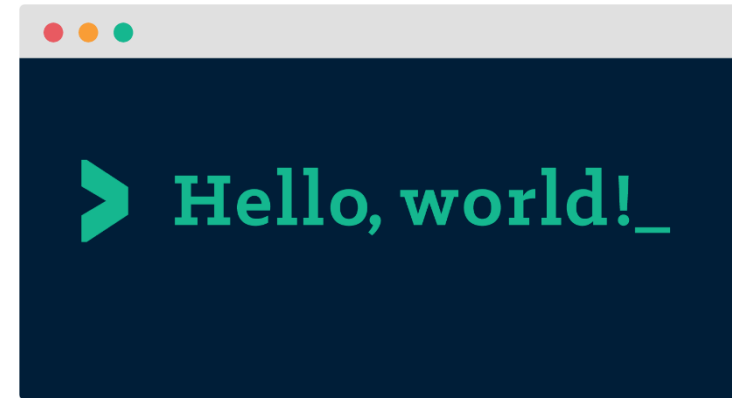


1. Comentários

Quando o programa está ficando maior e mais complicado, fica difícil ler e manter. Por esses motivos, é uma boa prática inserir algumas documentações ou notas em seu código. Essas notas são chamadas de comentários.

O comentário do Python começa com um sinal de *hash* ou tralha (#) e continua até o final da linha. É importante observar que o interpretador Python ignora comentários quando interpreta o código.

CONCEITOS BÁSICOS



1. Comentários

Comentário em bloco:

```
1 # increase price to 5%
2 price = price * 1.05
```

Comentário em linha:

```
1 salary = salary * 1.02 # increase salary 2% for the employee
```

Comentário para documentação de funções, módulos, pacotes, classes, etc:

```
1 def quicksort():
2     """ sort the list using quicksort algorithm """
3     ...
```

```
1 def increase_salary(sal, rating, percentage):
2     """ increase salary base on rating and percentage
3     rating 1 - 2 no increase
4     rating 3 - 4 increase 5%
5     rating 4 - 6 increase 10%
6
7     """
```

CONCEITOS BÁSICOS

2. Indentação

Python usa indentação para blocos, em vez de chaves. Tabs e espaços são suportados. Exemplo:

script.py

```
1 x = 1
2 if x == 1:
3     # indented four spaces
4     print("x is 1.")
5
```

IPython Shell

```
x is 1.
```

CONCEITOS BÁSICOS

3. Variáveis

O nome de uma variável pode possuir letras e números, porém não é permitido iniciar o nome de uma variável com um número. A primeira letra pode ser maiúscula.

Python é *case sensitive* (diferencia letras maiúsculas de minúsculas)

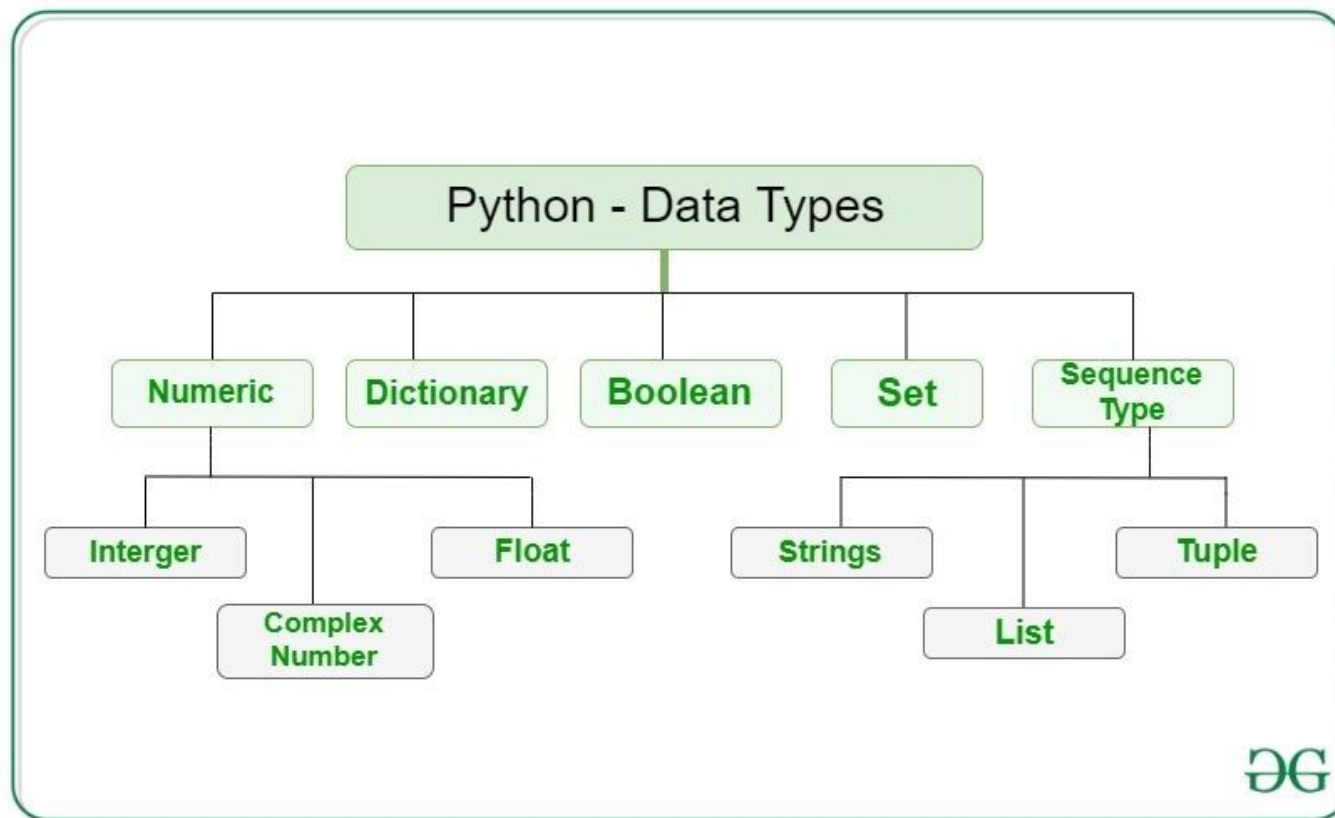
O *underline* _ pode aparecer no nome da variável.

O interpretador Python utiliza **palavras chaves** para reconhecer a estrutura do programa e essas palavras **não** podem ser usadas como nomes de variáveis, elas são:

and, as, assert, break, class, continue,
def, del, elif, else, except, finally, for ,
from, global, if, import, in, is, lambda,
not, or, pass, print, raise, return, try,
while, with, yield

CONCEITOS BÁSICOS

4. Tipos de dados



CONCEITOS BÁSICOS

Tipos de dados: **Numérico**

No Python, o tipo de dado numérico representa os dados que possuem valor numérico. O valor numérico pode ser interger, número flutuante ou mesmo números complexos. Esses valores são definidos como int, float e classe complexa em Python.

Integers - Este valor é representado pela classe int. Ele contém números inteiros positivos ou negativos (sem fração ou decimal).

Float - Este valor é representado pela classe de float. É um número real com representação de ponto flutuante. É especificado por um ponto decimal.

Números complexos - O número complexo é representado por uma classe complexa. É especificado como (parte real) + (parte imaginária) j.

CONCEITOS BÁSICOS

Tipos de dados: **Booleano**

Tipo de dados com um dos dois valores internos: Verdadeiro ou Falso. É indicado pela classe bool.

```
# Python program to  
# demonstrate boolean type  
  
print(type(True))  
print(type(False))
```

Output:

```
<class 'bool'>  
<class 'bool'>
```

CONCEITOS BÁSICOS

Tipos de dados: **Sequências**

Sequência é a coleção ordenada de tipos de dados semelhantes ou diferentes. Sequências permitem armazenar vários valores de forma organizada e eficiente. Existem vários tipos de sequência no Python: String, Lista e Tupla.

Strings - são matrizes de bytes que representam caracteres. Uma string é uma coleção de um ou mais caracteres inseridos em aspas simples, aspas duplas ou triplas. É representado pela classe str.

Acessando elementos dentro de uma string

G	E	E	K	S	F	O	R	G	E	E	K	S
0	1	2	3	4	5	6	7	8	9	10	11	12
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

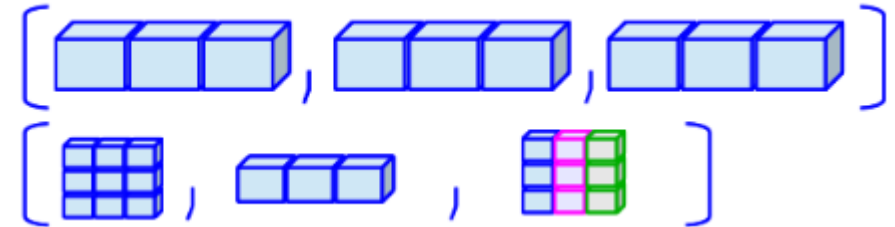
CONCEITOS BÁSICOS

Tipos de dados: **Sequências**

Sequência é a coleção ordenada de tipos de dados semelhantes ou diferentes. Sequências permitem armazenar vários valores de forma organizada e eficiente. Existem vários tipos de sequência no Python: String, Lista e Tupla.

Lista - As listas são como as matrizes, declaradas em outros idiomas. As listas não precisam ser sempre homogêneas. Uma única lista pode conter DataTypes como Inteiros, Strings e Objetos. As listas são mutáveis e, portanto, podem ser alteradas mesmo após a criação. A lista em Python é ordenada e tem uma contagem definida. É representada pela classe da list.

Lists



CONCEITOS BÁSICOS

Tipos de dados: **Sequências**

Sequência é a coleção ordenada de tipos de dados semelhantes ou diferentes. Sequências permitem armazenar vários valores de forma organizada e eficiente. Existem vários tipos de sequência no Python: String, Lista e Tupla.

Tupla - é uma coleção ordenada de objetos Python, semelhante a uma lista. A sequência de valores armazenados em uma tupla pode ser de qualquer tipo e são indexados por números inteiros.

A diferença importante entre uma lista e uma tupla é que as tuplas são imutáveis. É representado pela classe tuple.

CONCEITOS BÁSICOS

Tipos de dados: **Set**

Set é uma coleção não ordenada de tipos de dados que é iterável, mutável e sem elementos duplicados. A ordem dos elementos em um conjunto é indefinida, embora possa consistir em vários elementos.

A principal vantagem de usar um conjunto, em oposição a uma lista, é que ele possui um método altamente otimizado para verificar se um elemento específico está contido no conjunto.

CONCEITOS BÁSICOS

Tipos de dados: **Dicionário**

Dicionário em Python é uma coleção não ordenada de valores de dados, usada para armazenar valores de dados como um mapa, que, diferentemente de outros tipos de dados que mantêm apenas um valor único como elemento, o Dicionário mantém um par ***chave:valor***.

Cada par chave-valor é separado por dois pontos, enquanto cada chave é separada por uma "vírgula".

CONCEITOS BÁSICOS

5. Operadores

Quando mais de um operador ($*$ $+$ $-$ $/$ $**$) aparece em uma mesma expressão, a ordem de avaliação depende das regras de precedente. Para operadores matemáticos, Python segue a convenção matemática. O acrônimo **PEMDAS** é um caminho útil para lembrarmos da convenção.

1. $P \rightarrow ()$
2. $E \rightarrow **$
3. $M \text{ e } D \rightarrow * \text{ e } /$
4. $A \text{ e } S \rightarrow + \text{ e } -$

CONCEITOS BÁSICOS

5. Type Casting

Alterando uma variável para um número inteiro

int()

Alterando uma variável para um número real

float()

Alterando uma variável para uma string

str()

EXERCÍCIOS

1. Criar uma variável com valor 6 (tipo inteiro). Criar outra variável com o valor 2 (tipo inteiro). Dividir a primeira pela segunda (salvar em uma nova variável). Ver o tipo do resultado da divisão.

2. Ao final de um ano, dois amigos querem dividir o lucro obtido através de um site por onde prestam serviço de consultoria para projetos de IA. Digamos que o lucro mensal seja de 8k. O amigo1 tem direito a 30% e o restante pertence ao amigo2. Calcule o lucro total do ano e o lucro de cada amigo.

3. Faça a conta de cabeça e diga qual será o resultado da expressão:

$$5 + 3 * 10 / 3 == 15$$

Confirme pelo Python!

EXERCÍCIOS

- Use os dados fornecidos e faça um script para atender as necessidades descritas abaixo:

```
movieName = "The Shining"
actors=["Jack Nicholson", "Shelley Duvall", "Danny Lloyd", "Scatman Crothers", "Barry Nelson"]
scores = [4.5,4.0,5.0]
comments = ["Best Horror Film I Have Ever Seen", "A truly brilliant and scary film from Stanley Kubrick", "A masterpiece of psychological horror"]
```

1. Crie uma lista com os 4 elementos carregados: nome do filme, atores, avaliações e comentários; Faça todas as questões seguintes usando esta lista!
2. Imprima o vetor com o nome do primeiro ator;
3. Imprima a melhor avaliação do filme (score e comentário)

Dica1: `max(lista)` retorna o máximo valor de uma lista

Dica2: `lista.index(valor da lista)` retorna o índice do valor na lista

CONCEITOS BÁSICOS

6. Interação com usuário

Interagindo com o usuário

input() → pergunta diretamente ao usuário e a resposta é uma variável do tipo string

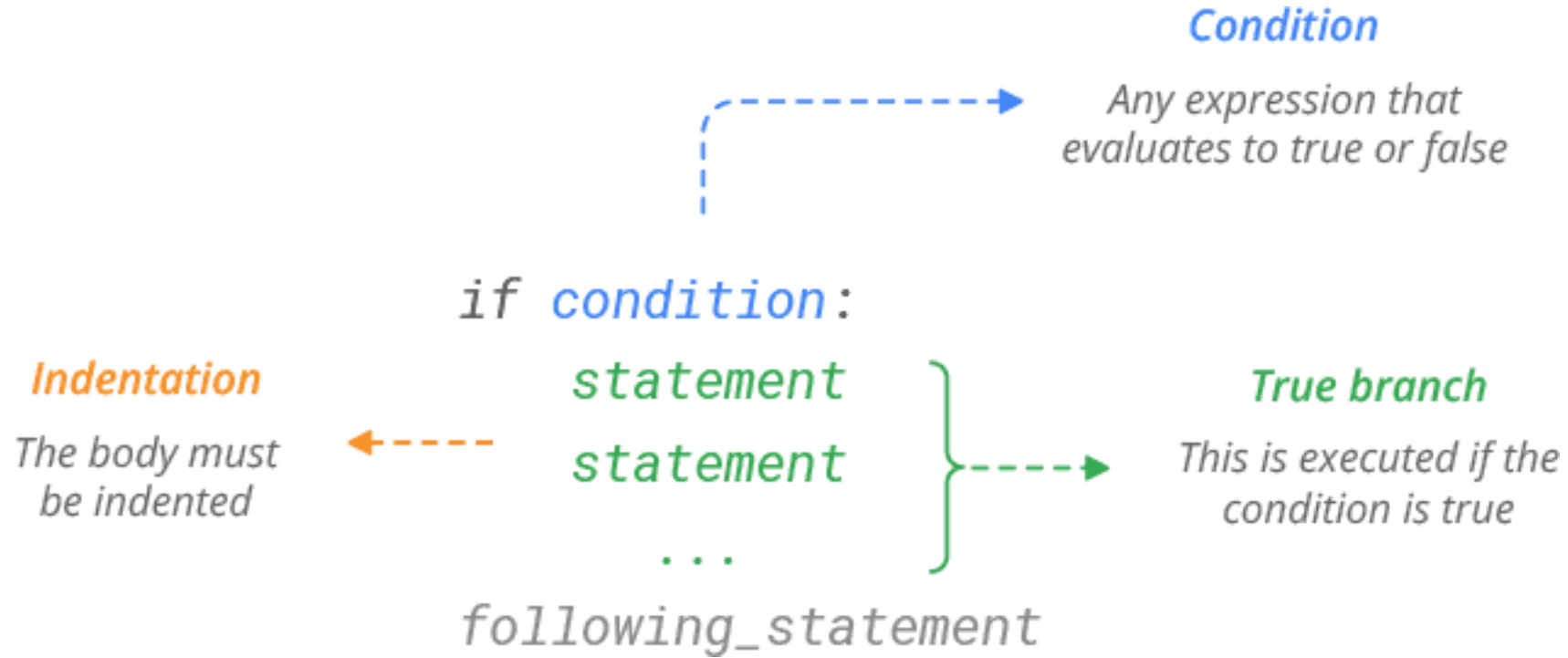
Checando o tipo de variável

type(variable) → fornece o tipo da variável entre ()

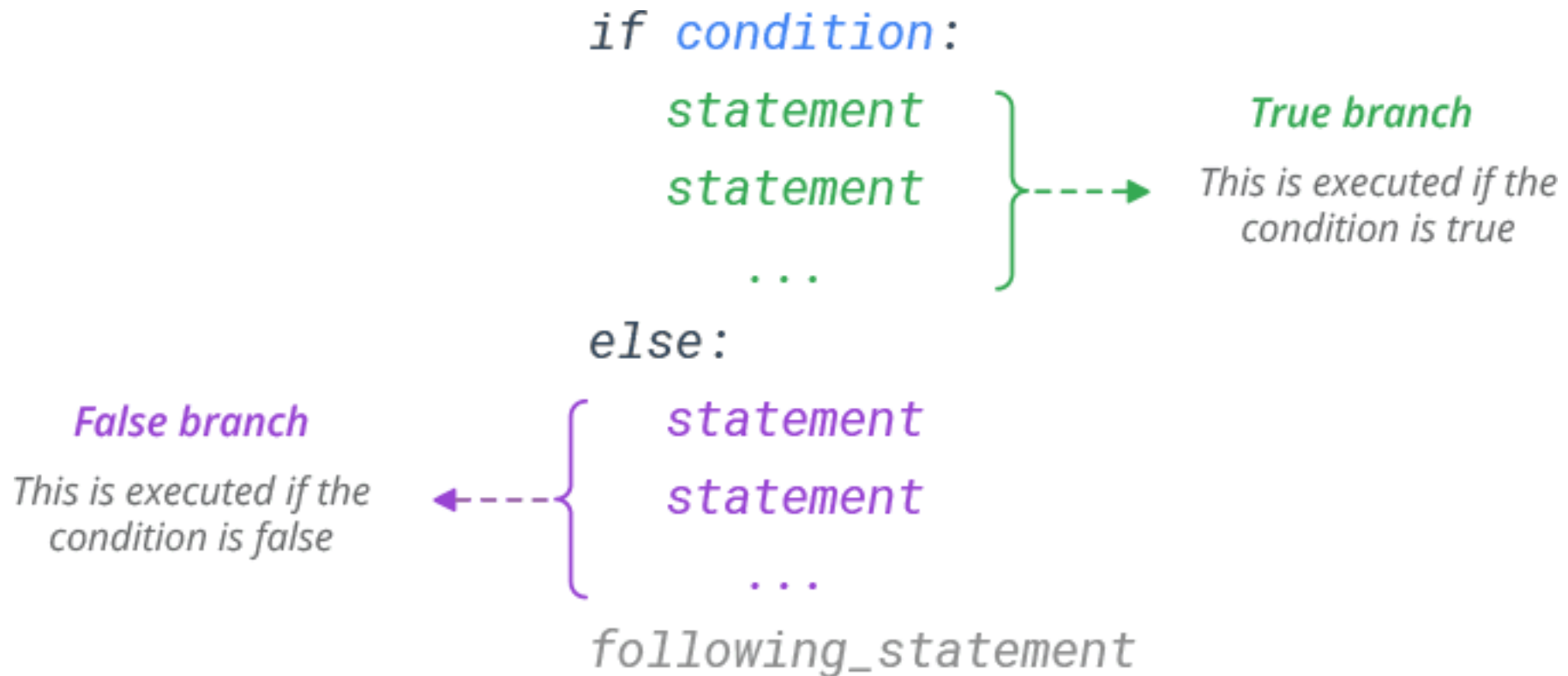
EXERCÍCIO

Exercício: crie um script que pergunte quantos anos o usuário tem e espere a resposta do mesmo. Em seguida imprima a idade, o tipo, converta para numérico e imprima novamente o tipo.

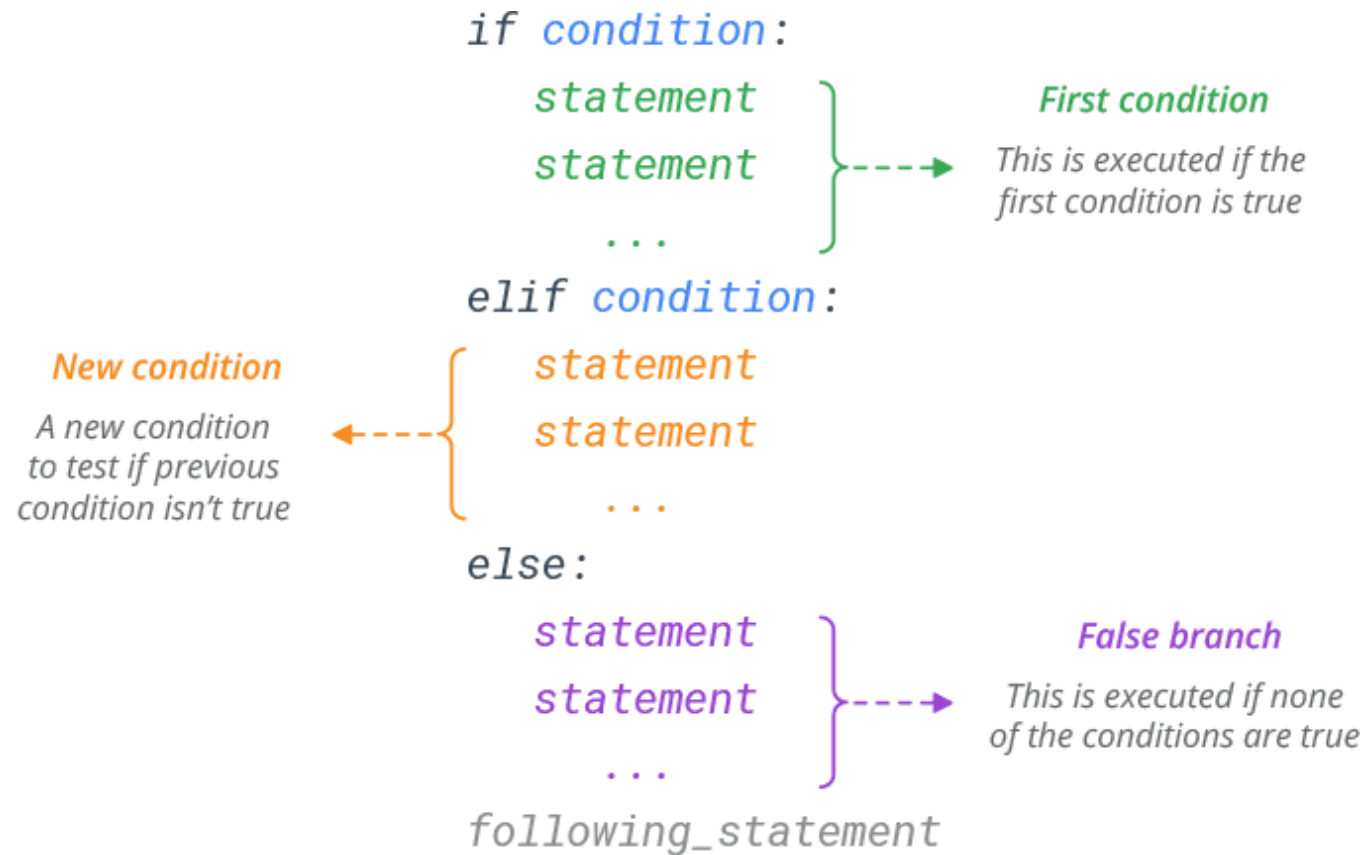
ESTRUTURAS CONDICIONAIS



ESTRUTURAS CONDICIONAIS

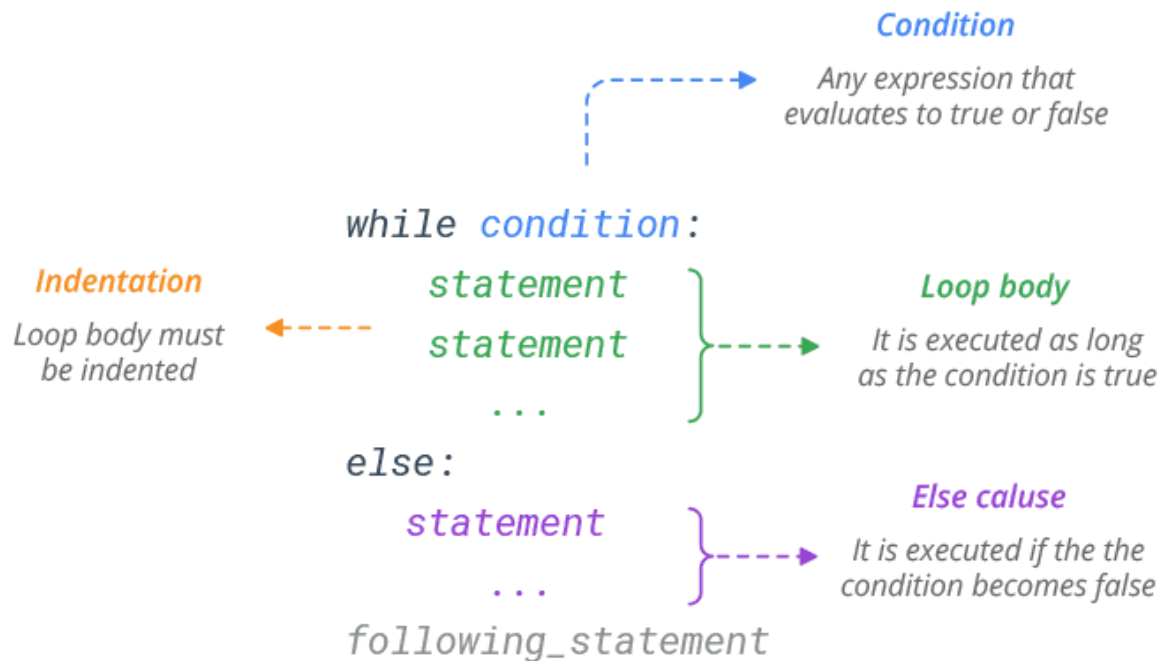


ESTRUTURAS CONDICIONAIS

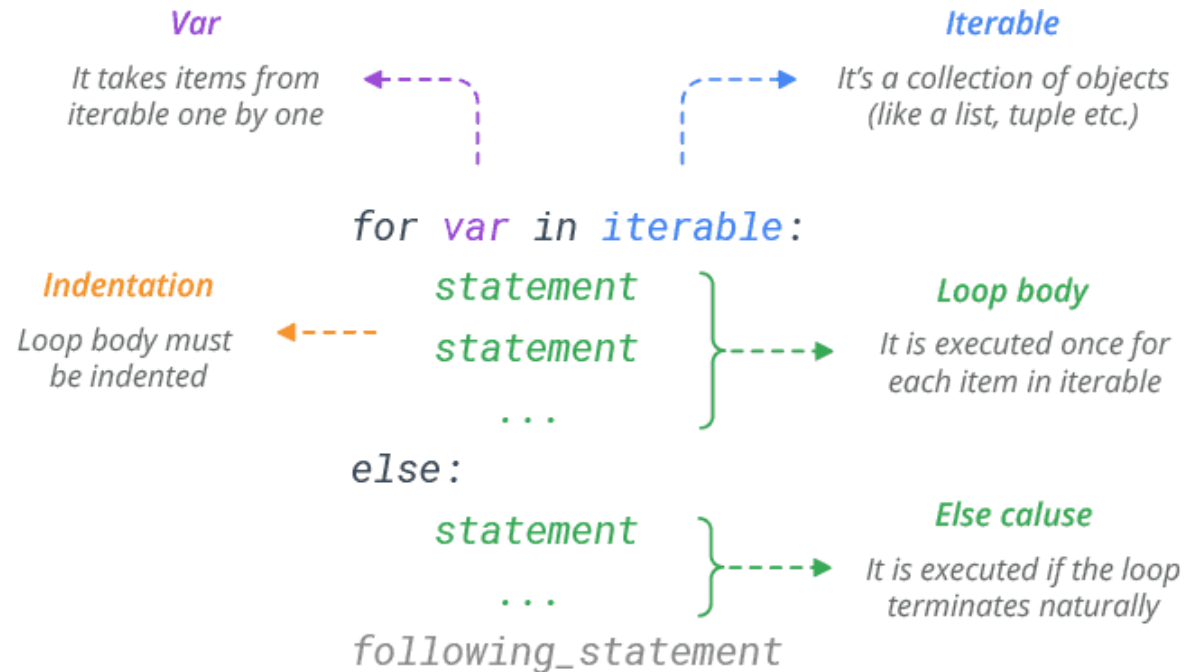


ESTRUTURAS DE REPETIÇÃO

WHILE LOOP



FOR LOOP



EXERCÍCIOS

1. Imprima uma sequência de 25 números consecutivos.
2. Imprima a sequência acima 2 vezes. Dica: for dentro de for ou while dentro de while, etc...
3. Na sequência de 0 a 30, imprima somente os números pares. Dica: operador para pegar o resto de uma divisão: %
4. Mega Sena: 6 números entre 1 e 60 (1 sorteio). (usando uma estrutura de repetição e sem usar estruturas de repetição).

Dica sem usar estrutura de repetição: função `np.random.randint()` (import numpy as np)

Pergunta: estrutura de repetição e `np.random.randint` sem nenhum tratamento pode gerar algum problema?

Dica: `np.random.choice()`
5. Simule o resultado de um dado de 6 faces jogado 7 vezes. (usando uma estrutura de repetição e sem usar estruturas de repetição). Dica: função `np.random.randint()`

Exemplo de uso da função:

`np.random.randint(1, 60, 1) ==>` sorteio de 1 número entre 1 e 60

EXERCÍCIOS

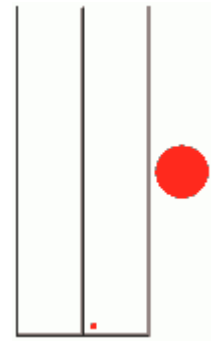
6. Sabendo que o fatorial de um número é definido como:

$$x! = x \cdot (x - 1) \cdot (x - 2) \cdot \dots \cdot 1$$

Crie um script que calcule o fatorial de um número qualquer.

EXERCÍCIO

LEI DOS GRANDES NÚMEROS



$$\overline{X_n} \rightarrow E(X) \text{ quando } n \rightarrow \infty$$

EXERCÍCIO

LEI DOS GRANDES NÚMEROS

10: 7/3

70% / 30%

100: 52/48

52% / 48%

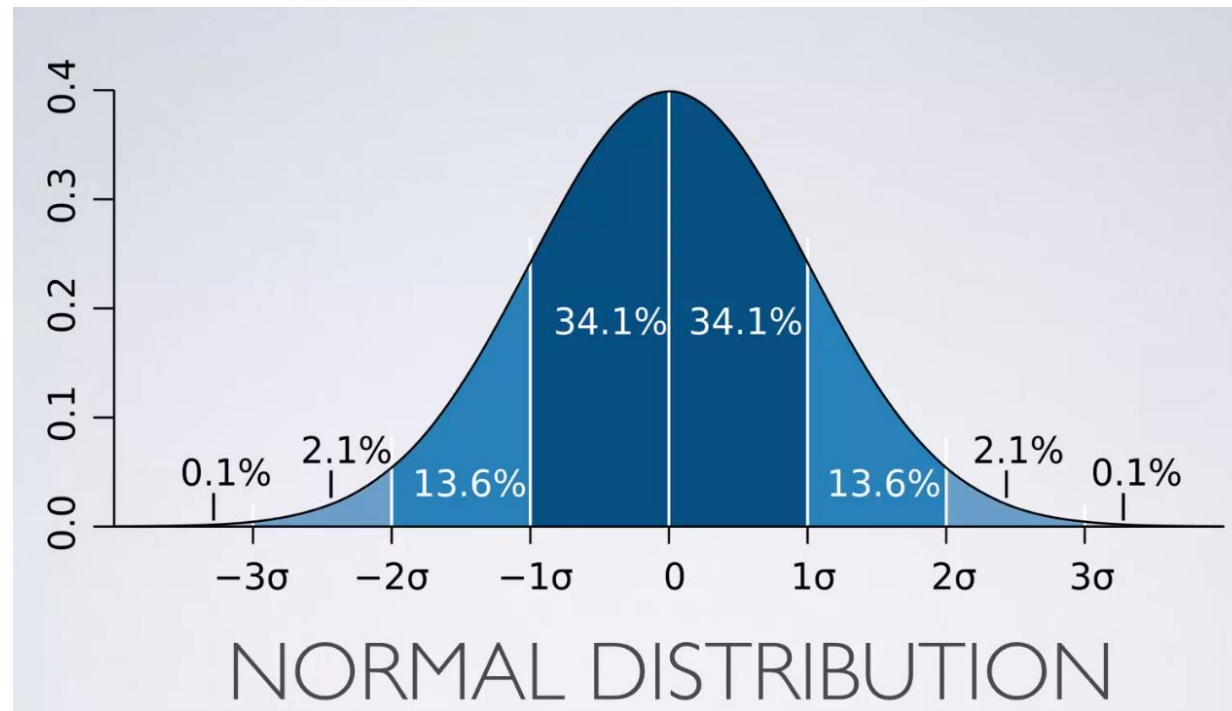
1000: 502/498

50.2% / 49.8%

...

EXERCÍCIO

LEI DOS GRANDES NÚMEROS



EXERCÍCIO

LEI DOS GRANDES NÚMEROS

Teste a lei dos grande números para N números aleatórios gerados com distribuição normal (média = 0 e desvio padrão = 1).

Crie um script que conte quantos desses números caem entre -1 e 1 e divida por N .

Você sabe que $E(X) = 68.2\%$

Verifique que a média(X_n) tende a $E(X)$ conforme você roda o script para valores maiores de N .

Dica:

```
import numpy as np
```

```
np.random.normal(1)
```

Funções

FUNÇÕES



Retorno

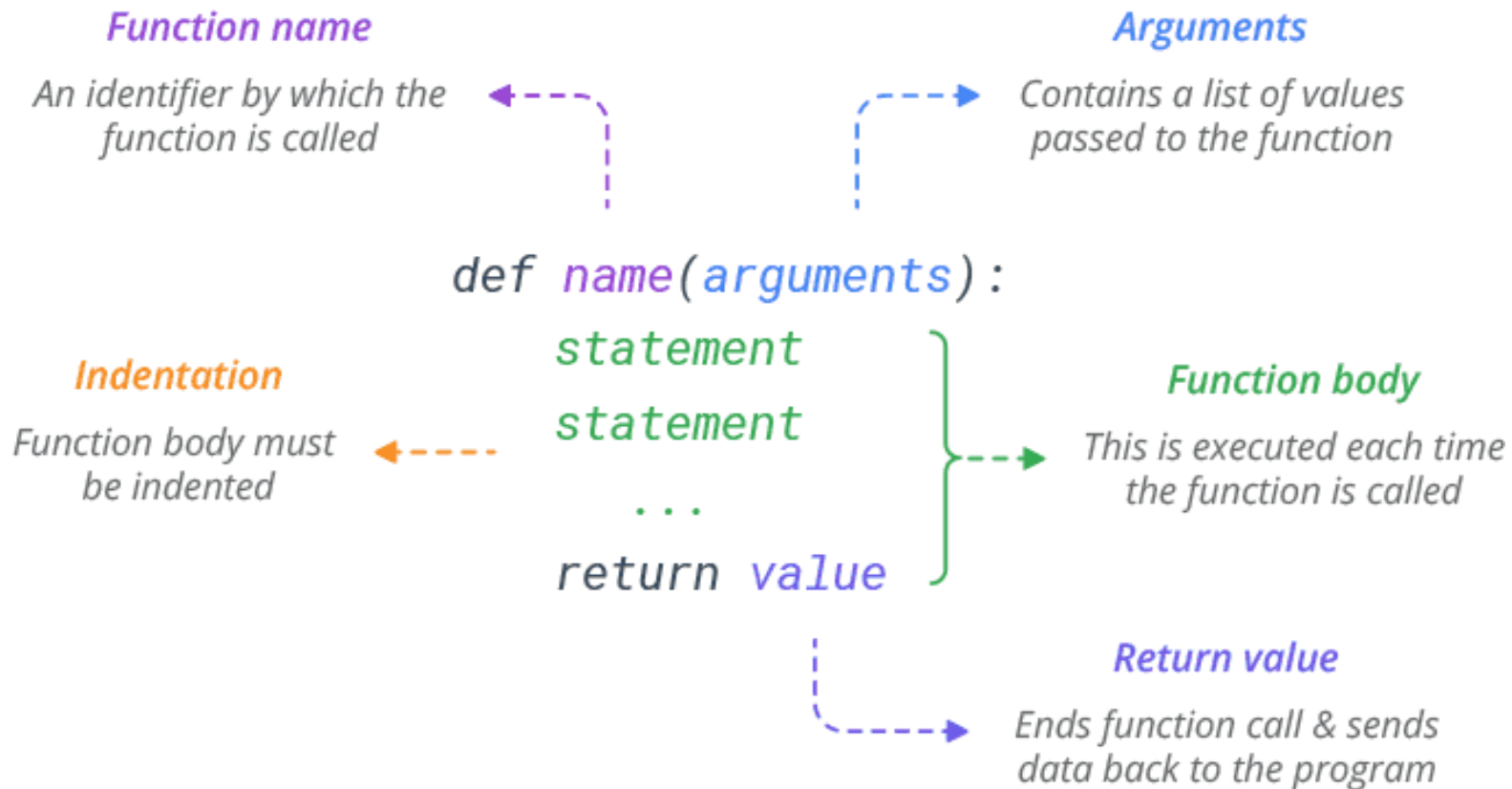


Função



**Parâmetros
de entrada**

FUNÇÕES



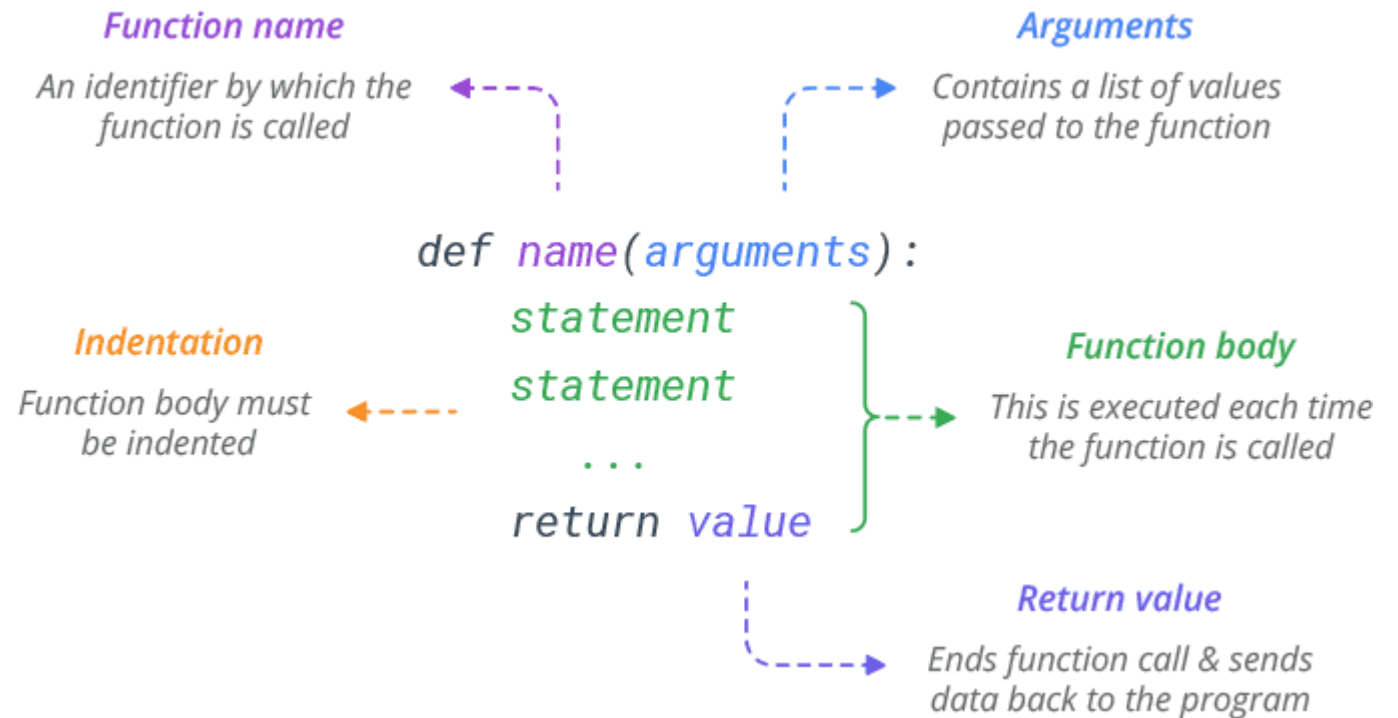
A função deve ser declarada antes de sua chamada!

CRIANDO FUNÇÕES

As funções são o primeiro passo para a reutilização de código.

Eles permitem definir um bloco de código reutilizável que pode ser usado repetidamente em um programa.

O Python fornece várias funções internas, como `print ()` e `len ()`, mas você também pode definir suas próprias funções para usar em seus programas.



EXERCÍCIOS

1. Sabendo que o fatorial de um número é definido como:

$$x! = x \cdot (x - 1) \cdot (x - 2) \cdot \dots \cdot 1$$

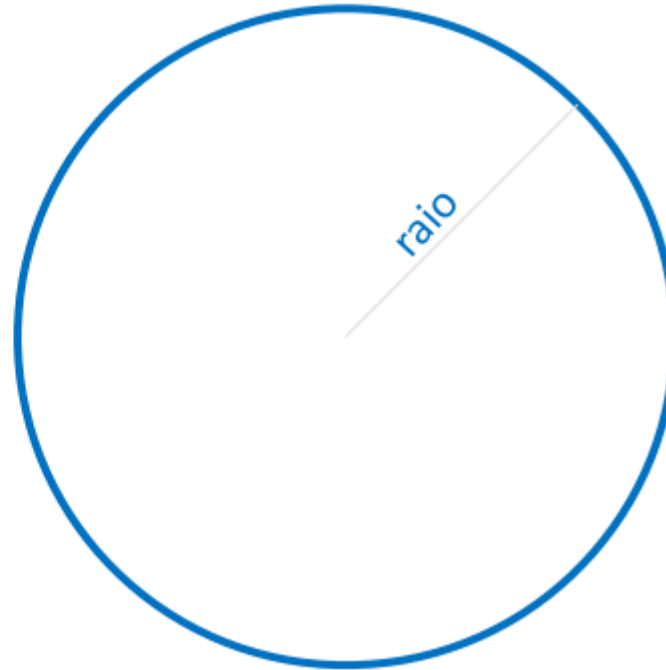
Crie **uma função** que calcule o fatorial de um número qualquer.

2. Agora tente usar a função factorial do pacote math para calcular o fatorial de um número.

EXERCÍCIOS

3. Crie uma função que receba o valor do raio de um círculo e retorne o valor da área do mesmo. Utilize o pacote ***math***.

$$S_{círculo} = \pi r^2$$

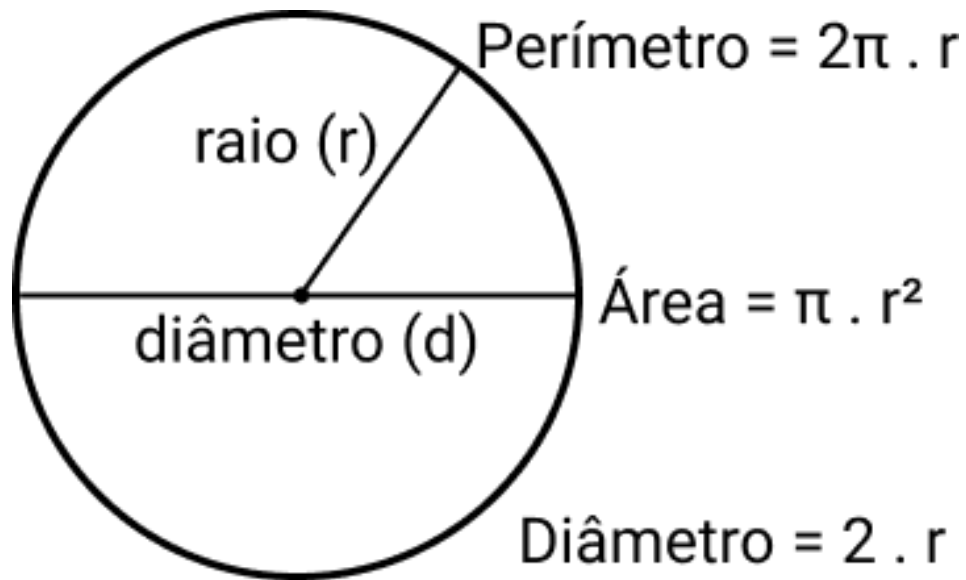


Dica: `math.pi`

Potência: `**`

EXERCÍCIOS

4. Altere a função criada anteriormente para que ela retorne, além do valor da área do círculo, retorne também o diâmetro e o perímetro.



Agora altere a função para que o raio tenha um valor padrão de 5.

EXERCÍCIO

Crie uma função para validar a lei dos grandes números para o problema proposto anteriormente.

A função deve receber como parâmetro o número de experimentos.

Após validação da função, inclua um parâmetro opcional de intervalo a ser validado. O valor *default* deve ser -1 e 1.

Teste outros intervalos e verifique a lei dos grandes números.

