

Universidade do Vale do Itajaí
Escola do Mar, Ciência e Tecnologia

Introdução ao Python

LEDS - Laboratory of Embedded and Distributed Systems

Agenda

- Introdução
 - Instalação
 - Tipos de dados
 - Operações
 - Ferramentas de controle de fluxo
 - Numpy
 - OpenCV
-

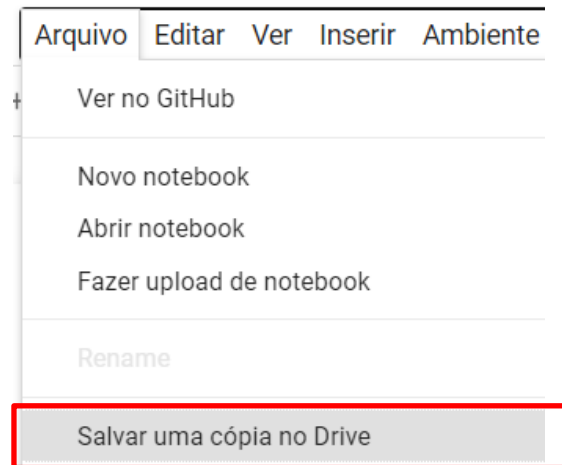
Versão v1.1

- Autor
 - Douglas Almeida Dos Santos
- Revisão
 - Felipe Viel

Introdução


Local com material

- Primeiro acesso <https://colab.research.google.com/>
 - Faça o login com a sua conta do Google (ou crie uma para usar na aula)
- Agora acesse o meu Github, no repositório ColabProjects (<https://github.com/VieIF/ColabProjects/>)
 - Acesse o arquivo Tutorial_Python_Introdução.ipynb
 - Clique no ícone  Open in Colab
 - Com a aba aberta do Colab,
 - clique no menu Arquivo -> Salvar uma cópia no Drive
 - Você irá criar uma cópia e ela já fica salvo no seu Drive



Execução

- Clique em conectar para ganhar um PC na Nuvem

 Compartilhar



Conectar ▼

- Você pode criar células de código e texto
- Para executar as células de código, clique no símbolo de play



História do Python

- Criada pelo holandês **Guido Van Rossum**
- Começou a criar no seu tempo livre em Dezembro de 1989
- Começou como um descendente da linguagem de programação ABC
- Nome baseado na série de comédia *The Monty Python's Flying Circus*
- Primeira versão em 1990



fonte:

https://python.swaroopch.com/about_python.html

Vantagens

- Python é fácil de aprender, altamente legível e fácil de usar
- Linguagem minimalista
- Suporte a avançados paradigmas de engenharia de software
- Códigos menores que os de outras linguagens
- Multiplataforma
- Grande quantidade de bibliotecas
- Integração de componentes de diferentes linguagens

- Desenvolvimento de aplicações WEB
 - Django, Flask, CherryPy e Bottle
- Computação científica e numérica
 - NumPy, SciPy, Pandas, matplotlib, e IPython
- Programação de GUI
 - wxWidgets, PyQt, PySide, Tkinter
- Prototipação de software
 - Pygame

Características básicas

- Case-sensitive
- Variáveis não precisam ser declaradas
- Sem ponto-e-vírgula
- Blocos de código delimitados de acordo com a indentação

Exemplo de código - Ola Mundo

```
print('Ola Mundo')
```

Exemplo de código

```
nome = input("Digite seu nome: ")
if nome == "Calouro":
    print('Olá Calouro')
else:
    print('Seu nome é: ', nome)
```

Executando arquivo

1. Crie um arquivo com a extensão **.py**
2. Execute pelo terminal, com o comando:

`python nome_arquivo.py`

Instalação

Instalação no Windows



Opção B: <https://repl.it/languages/python3>

Instalação no Linux

```
sudo apt install python3 pip3
```


Instalação dos módulos

```
pip3 install numpy opencv-python matplotlib
```

Modo interativo

- Abra o terminal e execute o comando **python**
- Digite comandos com operações básicas

```
douglas@douglas-G3-3579:~$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information
>>> 1+1
2
>>> (5 / 5) + 7
8.0
>>> █
```

Tipos de Dados

Tipos básicos de dados

Tipo do dado	Comando	Criação
Texto	str	<code>var = 'Teste' ou var = "Teste"</code>
Inteiro	int	<code>var = 123</code>
Complexo	complex	<code>var = 1j</code>
Ponto flutuante	float	<code>var = 123. ou var = 123.0 ou var = .123</code>
Lista	list	<code>vet = [] ou vet = list()</code>
Conjunto	set	<code>vet = set()</code>
Dicionário	dict	<code>vet = {} ou vet = dict()</code>

Adaptado dos slides de Vinícius A. dos Santos

Lista

```
list.append(x)  
list.extend(iterable)  
list.insert(i, x)  
list.remove(x)  
list.pop(i)  
list.clear()  
list.index(x[, start[, end]])  
list.count(x)  
list.sort(key=None, reverse=False)  
list.reverse()  
list.copy()
```

Exemplo

```
douglas@douglas-G3-3579:~$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x = [1, 2, 3]
>>> x
[1, 2, 3]
>>> x.append(1)
>>> x
[1, 2, 3, 1]
>>> x = set(x)
>>> x
{1, 2, 3}
>>> 
```

Operações

Operações aritméticas e lógicas

Aritméticas:

```
x = x + 1
```

```
x += 1
```

```
x = x ** 2  
x = x // 2
```

Lógicas:

```
'a' in ['a', 'b', 'c']
```

```
'a' in {'a': 23, 'b': 54}
```

```
a is b
```

```
a is not b
```

```
a == b
```

```
not a == b
```

```
a and b
```

```
a or b
```

Adaptado dos slides de Vinícius A. dos Santos

Ferramentas de controle de fluxo

Desvios condicionais

```
if x is True:  
    print("True")  
elif x is not False:  
    print("Still true")  
else:  
    print("False")
```

Laços de repetição

```
words = ['cat', 'window', 'defenestrate']
```

```
for w in words:  
    print(w)
```

```
for i in range(10):  
    print(i)
```

```
x = 0  
while x < 10:  
    print(x)  
    x += 1
```

Exemplo

```
matrix = [[0,1,2]]*3
```

```
transposed = [[row[i] for row in matrix] for i in range(3)]
```

Exemplo

```
>>> x = 0
>>> while x < 10:
...     print(x)
...     x += 1
...
0
1
2
3
4
5
6
7
8
9
>>> for i in range(10):
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
>>>
```

Definição de funções

```
def fib(n):    # write Fibonacci series up to n
    """Print a Fibonacci series up to n."""
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b

# Now call the function we just defined:
fib(2000)
```

Resultado

```
>>> def fib(n):    # write Fibonacci series up to n
...     """Print a Fibonacci series up to n."""
...     a, b = 0, 1
...     while a < n:
...         print(a, end=' ')
...         a, b = b, a+b
...
>>> fib(2000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 >>>
>>>
>>> 
```

Definição de funções

```
def fib(n: int) -> list:    # write Fibonacci series up to n
    """Print a Fibonacci series up to n."""
    a, b = 0, 1
    result = []
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result

print(fib(n=100))
```


Numpy

Sobre o NumPy

- Pacote fundamental para computação científica com Python
 - Poderoso objeto de arrays N-dimensionais
 - Funções sofisticadas
 - Ferramentas para integração de códigos em C/C++ e Fortran
 - Álgebra linear, Transformada de Fourier e capacidade de gerar números aleatórios

Array Multidimensional

- Principal objeto do NumPy
- Array homogêneo
- Principais atributos
 - ndarray.ndim
 - ndarray.shape
 - ndarray.size
 - ndarray.dtype
 - numpy.int32, numpy.int16, e numpy.float64
 - ndarray.itemsize
 - ndarray.data

Exemplo

```
import numpy as np
```

```
arr = np.array([1, 2, 3])
```

```
print(arr.shape)
```

```
print(arr)
```

```
(3,)  
[1 2 3]
```

Criando um array

```
arr = np.zeros((3, 5)) # Array de zeros com 3 linhas e 5 colunas
```

```
arr = np.ones((3,5)) # Array de um com 3 linhas e 5 colunas
```

```
arr = np.array([10, 20, 30], dtype=np.float64) # Inicializado com elementos, formato forçado para float64
```

```
arr = np.empty((2, 3)) # Array criado sem valores definidos (lixo)
```

```
arr = np.arange(5, 10, 0.5) # Numeros de 5 a 10, com passo de 0.5
```

```
arr = np.linspace( 0, 2, 9 ) # 9 números de 0 a 2
```

```
arr = np.random.random((2,3)) # Numeros aleatórios, com 2 linhas e 3 colunas
```

Operações básicas

- $C = A + B$
 - Soma elementar
- $C = A * B$
 - Multiplicação elementar
- $C = A @ B$
 - Produto matricial

- Faça um contador de valores
 - O algoritmo deve percorrer toda a matriz
 - Deve ser contada a quantidade de vezes que cada número aparece na matriz
 - O resultado para cada número deve ser armazenado num vetor, na sua posição correspondente
 - O maior valor possível é 255

0	4	2
2	4	2
3	0	5



0	1	2	3	4	5
2	0	3	1	2	1

OpenCV

Introdução

- Iniciada em 1999 por Gary Bradsky, disponibilizado a partir de 2000
- Suporta várias linguagens: C++, Python, Java, etc
- Disponível para os sistemas: Windows, Linux, OS X, Android, iOS, etc
- OpenCV-Python é a versão da API do Python do OpenCV.
- OpenCV-Python é a ferramenta apropriada para prototipação rápida de problemas de visão computacional



Carregando imagem

```
import cv2
```

```
# load image
```

```
# cv2.IMREAD_COLOR : Loads a color image
```

```
# cv2.IMREAD_GRAYSCALE : Loads image in grayscale mode
```

```
# cv2.IMREAD_UNCHANGED : Loads image as such including alpha channel
```

```
img = cv2.imread("lena_color.png", cv2.IMREAD_COLOR)
```

```
# show image
```

```
cv2.imshow('lena', img)
```

```
# wait until the key 0 is pressed to destroy all windows
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

Continuação da prática

- Obtenha o histograma da imagem
 - Aplique o algoritmo desenvolvido na prática anterior no array da imagem
 - Plote

```
import numpy as np
from matplotlib import pyplot as plt
import cv2
img = cv2.imread("lena_color.png", cv2.IMREAD_COLOR)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Digite seu código aqui

```
plt.plot(result_hist)
plt.show()
```

Solução

```
import numpy as np
from matplotlib import pyplot as plt
import cv2

img = cv2.imread("lena_color.png", cv2.IMREAD_COLOR)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

result_hist = np.zeros(256, dtype=int)
for line in gray:
    for pix in line:
        result_hist[pix] += 1

plt.plot(result_hist)
plt.show()
```

Descobrimos bordas

- Detector de bordas Canny

```
import cv2
```

```
img = cv2.imread("lena_color.png", cv2.IMREAD_COLOR)
```

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
canny = cv2.Canny(gray, 80, 200)
```

```
cv2.imshow('lena', canny)
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

Canny aplicado em tempo real

```
import cv2
# capture video
# from camera device (device index)
# from file (name of video file)
cap = cv2.VideoCapture(0)
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()
    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    canny = cv2.Canny(gray, 100, 200)
    # Display the resulting frame
    cv2.imshow('frame', canny)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

Operações Bit a Bit

- Operações bit a bit – bitwise
 - “ & ” implementa a operação lógica AND
 - “ | ” implementa a operação lógica OR
 - “ ^ ” implementa a operação lógica XOR
 - “ ~ ” implementa a operação lógica NOT
 - “ << ” implementa a operação de deslocamento a esquerda
 - “ >> ” implementa a operação de deslocamento a direita
- Permite a manipulação de bits em um número
 - Permite a utilização de máscaras para verificação de bits ativos

Exercício

- Em um byte com valor 27, identifique quais bits são 1's utilizando máscara e deslocamento e imprima na tela a quantidade. Repita o processo para 127, 59 e 1005

Utilização com Fourier

- Instalar os módulos matplotlib
 - `python -m pip install -U pip` (caso não esteja instalado)
 - `python -m pip install -U matplotlib`
- Instalar módulo sympy (manipulação algébrica)
 - `python -m pip install -U sympy`

Referências

Wiki do Python

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

Wiki do NumPy

- <https://docs.scipy.org/doc/numpy/dev/>

OpenCV-Python Tutorials

- <https://opencv-python-tutroals.readthedocs.io>

A byte of Python

- <https://python.swaroopch.com/>

Conclusão

- Python é legal



- É possível trabalhar com processamento de imagens com Python
- Adicione a skill ao seu LinkedIn