

Introdução ao Python

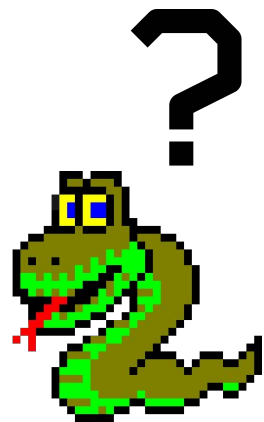


Prof. Francisco Carlos
fcarlosmonteiro@gmail.com

Minicurso
FLISoL - 2018

Por que Python?

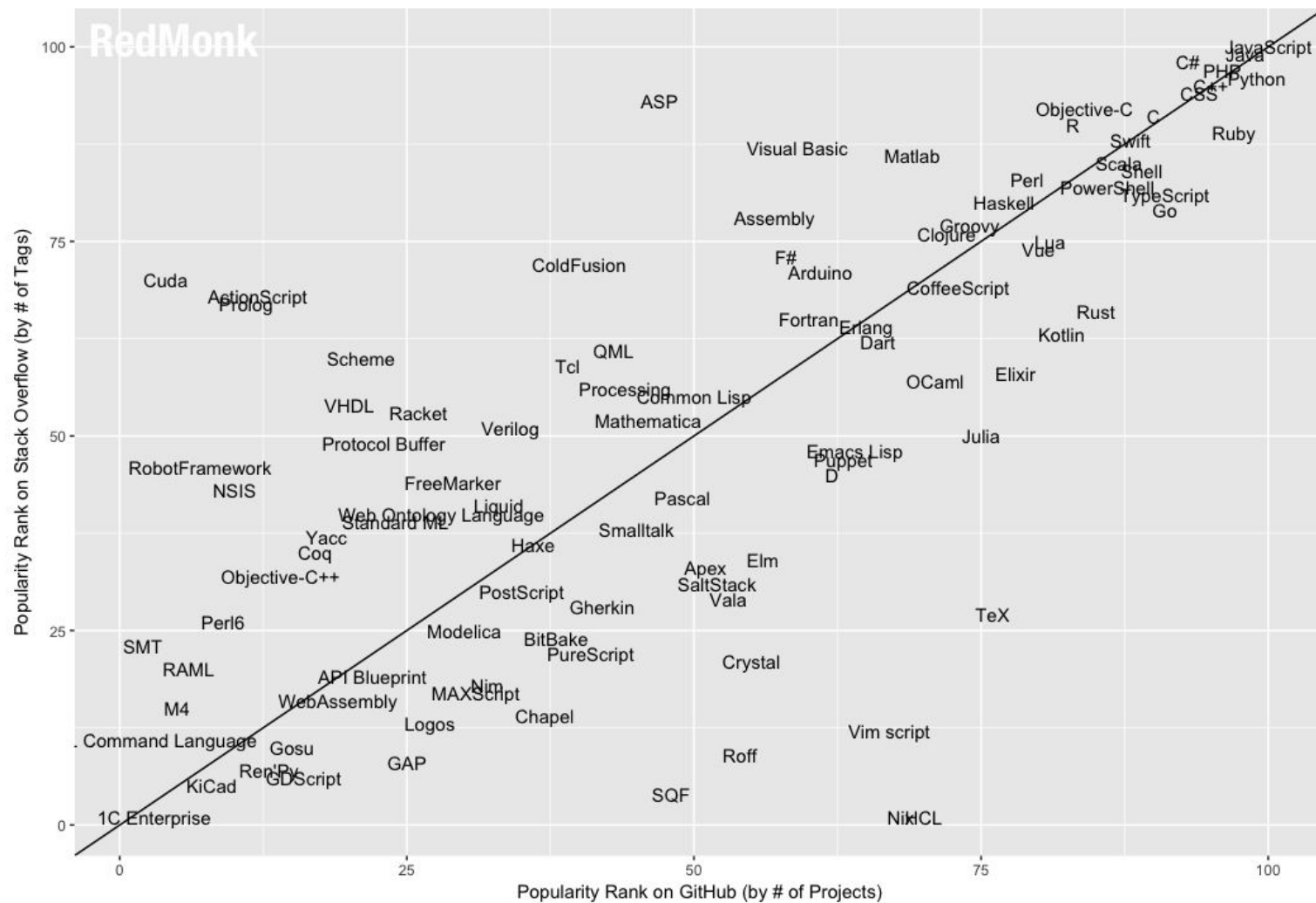
- Sintaxe limpa
- Fácil entendimento
- Comunidade forte
- Linguagem de alto nível
- Orientada à objetos
- Diversidade de bibliotecas
- Código Aberto



A Linguagem Python

- Criada em 1989 pelo holandês **Guido van Rossum**
- Possui tipagem **dinâmica**
- Não é necessário **declarar tipos de dado** das variáveis
- Linguagem **interpretada**
- Delimitação de bloco por **indentação**
- Leitura do código **mais amigável**
- IDE e Frameworks

RedMonk Q118 Programming Language Rankings



Quem usa Python?

Google

Spotify®

blender™

YAHOO!®

Dropbox

redhat®

mozilla
Firefox®

globo.com

Pinterest

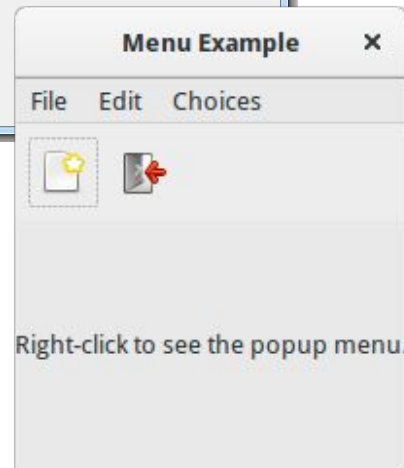
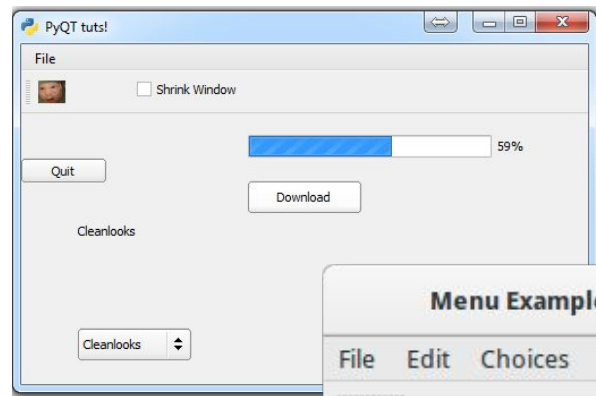
YouTube

Instagram

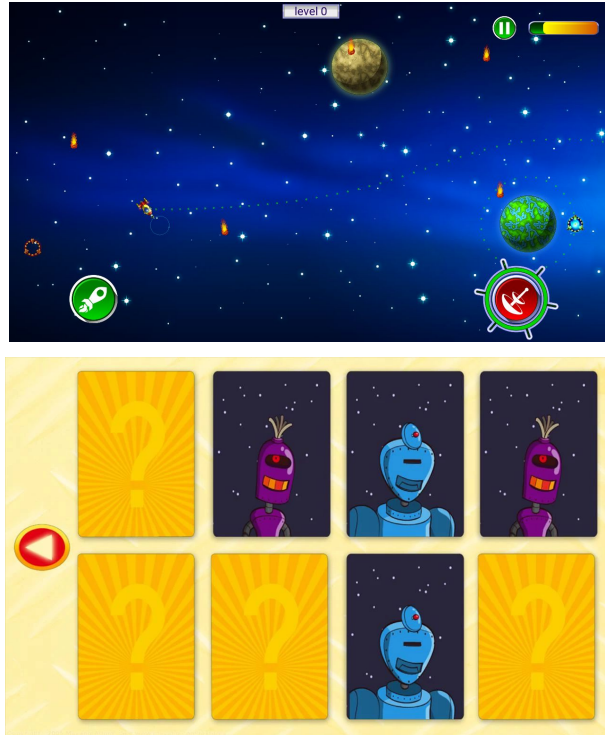


ubuntu®

O que pode se fazer com Python?

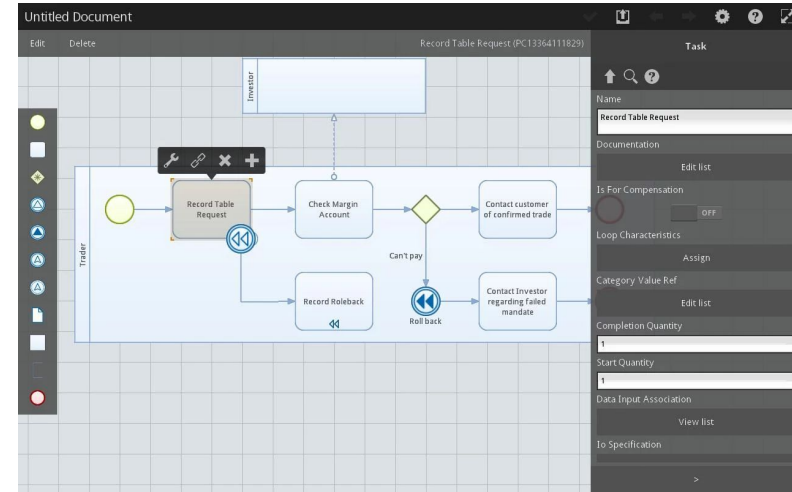
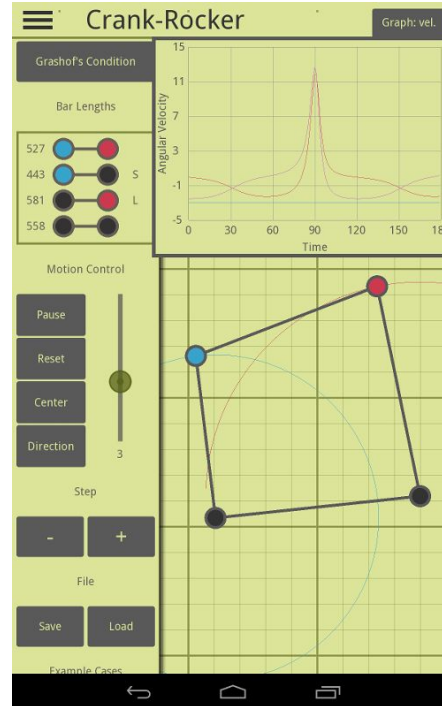
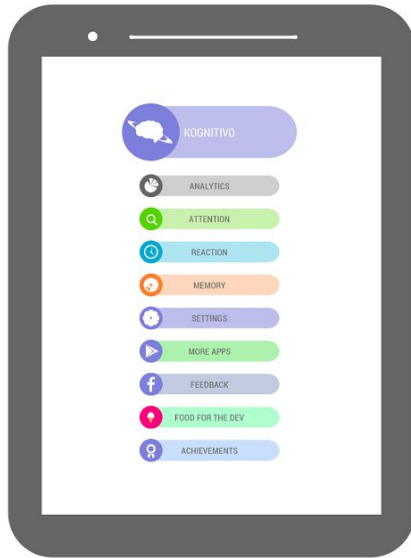


Projetos - Games



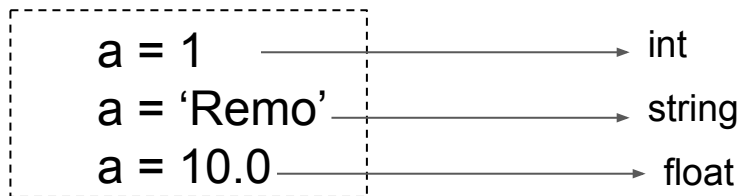
Projetos - Apps

4D
BRAIN TRAINER



Python - Noções Básicas

- Python possui tipagem **dinâmica**
- A tipagem **pode mudar** a cada nova entrada de dados em uma variável



- **Comentários** é representado por #
 - #Isso é um comentário em python

Python - Noções Básicas

- Executando um programa em python
 - Escrever em um arquivo com a **extensão .py** o seguinte código:
 - `print "Hello World"`
- Abrir o **terminal** no diretório do código:
 - `cd /caminho`
- **Executar** código:
 - `python programa.py`



Python - Noções Básicas

- Expressões **Aritméticas**

- $+$, $-$, $*$, $/$

- Expressões de **comparação**

- $<$, $>$, $<=$, $>=$, $==$, $!=$


- Expressões **lógicas**

- and , or , not

Soma	$a + b$
Subtração	$a - b$
Multiplicação	$a * b$
Divisão	a / b
Módulo (resto)	$a \% b$
Potenciação	$a ** b$

Python - Noções Básicas

- Não existe um **delimitador** específico para blocos de código
- A delimitação é feita pela **indentação**



```
1 a = 1
2
3 for i in range (0,10):
4     if a <= 10:
5         print "Testando identacao!"
6         a = a+2
7     else:
8         print "Flws"
```

Python - Estruturas de Controle IF

- A estrutura condicional **if** usa a seguinte sintaxe:

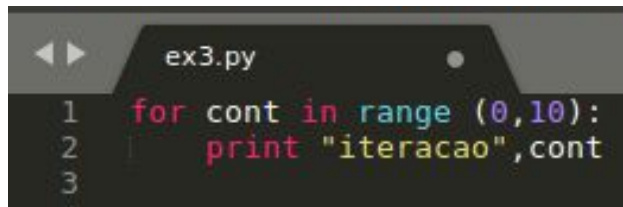
```
if condição:  
    comandos  
    ...  
elif condição:  
    comandos  
    ...  
else:  
    comandos  
    ...
```

```
1  a = 5  
2  b = 8  
3  
4  if a > b:  
5      print "valor de a=",a  
6      print "a maior que b"  
7  
8  elif a == b:  
9      print "valor de b=",b  
10     print "b maior que a"  
11  
12 else:  
13     print "valor de a=",a  
14     print "a menor que b"
```

Python - Estruturas de Controle FOR

- A estrutura **FOR** usa a seguinte sintaxe:

```
for cont in range (numInicial, numFinal):  
    comandos  
    ...
```

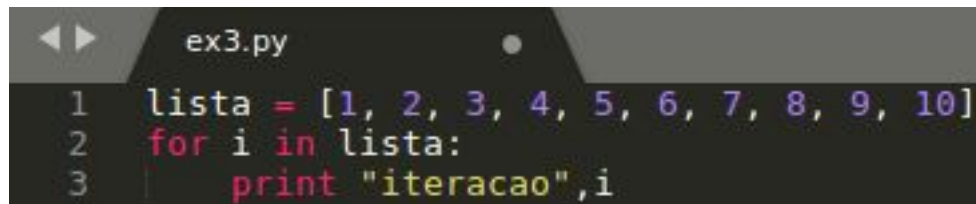
A screenshot of a code editor window titled 'ex3.py'. The editor has a dark background with light-colored text. It shows a Python for loop that iterates from 0 to 9, printing the word 'iteracao' followed by the current value of 'cont'. The code is as follows:

```
1  for cont in range (0,10):  
2      print "iteracao",cont  
3
```

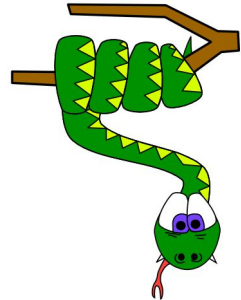
Python - Estruturas de Controle FOR

- Ou ele pode percorrer uma sequência de elementos:

```
lista = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for i in lista:
    comandos
    ...
```



```
ex3.py
1 lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2 for i in lista:
3     print "iteracao",i
```



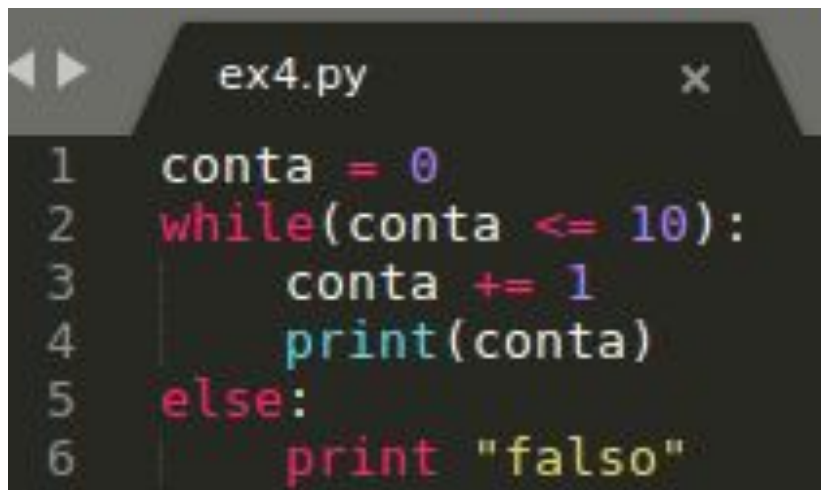
Exemplo

- Faça um programa que exiba todos números pares entre 1 a 10.
- Faça um programa que receba um número do usuário e responda se o número é par ou ímpar
 - exemplo: `i = input("Digite um numero")`
- Faça um programa que exiba todos números pares em uma lista de 10 números.

Python - Estruturas de Controle WHILE

- A estrutura **WHILE** usa a seguinte sintaxe:

```
while condição:  
    comandos  
    ...
```



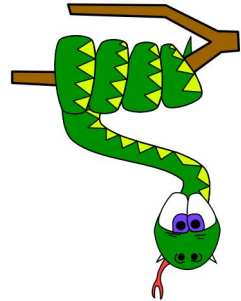
The screenshot shows a code editor window titled 'ex4.py'. It contains a Python script with a while loop that counts from 0 to 10. The code is as follows:

```
1  conta = 0  
2  while(conta <= 10):  
3      conta += 1  
4      print(conta)  
5  else:  
6      print "falso"
```

Python - Strings

- **String** é uma sequência imutável com o propósito de armazenar cadeias de caracteres.
 - `str = 'abcdef'`
- **Posição** de um carácter:
 - `str[0]`
- **Tamanho** de uma string:
 - `len(str)`
- **Concatenar** strings:
 - `str + str2`
- Uppercase e Lowecase:
 - `str.upper()`, `str.lower()`

Exemplo2



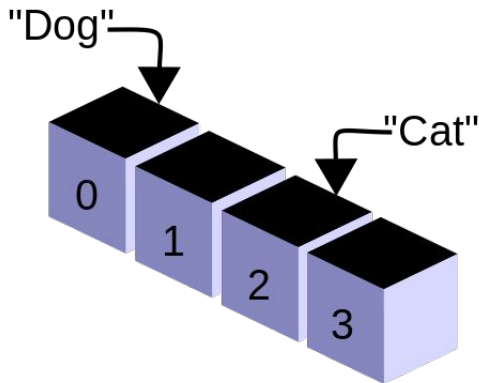
- Faça um programa que concatene duas strings e verifique quantas vezes a letra A aparece na string.

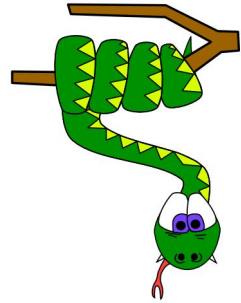
Python - Listas

- **Lista** é uma **sequência** de valores indexadas por um inteiro
- Uma lista pode conter **qualquer tipo** de valor:
 - `numeros = [1, 2, 3, 4, 5]`
 - `nomes = ['fora', 'lula', 'dilma', 'temer']`
 - `misto = [1, 2, 3.0, 'fora', 'temer']`
- O **tamanho** da lista pode ser verificado com o comando:
 - `len(numeros)`
- Os elementos da lista podem ser **acessados por meio de índices** que vão de 0 até o comprimento da lista-1:
 - `print numeros [-1]`

Python - Listas

- **Adicionando** um novo valor:
 - `numeros.append(0)`
- **Removendo** um valor:
 - `numeros.remove(0)`
- **Invertendo** a ordem dos valores:
 - `numeros.reverse()`
- **Obtendo** a posição de um valor:
 - `numeros.index(4)`





Exemplo3

- Faça um programa que verifique quantas vezes o número 10 aparece em uma lista
 - `lista = [1,2,10,3,4,10,5,6,10,7,8,10,9,10]`
- Faça um programa que remova todos números 10 da lista

Concatenação de Listas

```
lista1 = [1,2,3]
lista2 = ["a","b","c"]
print "Lista 1: ", lista1
print "Lista 2: ", lista2

for i in range(len(lista2)):
    lista1.append(lista2[i])
print(lista1)
```

Tem um maneira mais simples

```
lista1 = [1,2,3]
lista2 = ["a","b","c"]

print "Lista 1: ", lista1
print "Lista 2: ", lista2

#Basta somar as duas listas
lista3=lista1+lista2
print "Lista 3: ", lista3
```



Manipulação de Arquivos

- Ler e manipular arquivos são necessidades comuns no dia dia de programadores
 - Exemplo: Armazenamento de dados, criação de logs, obtenção de informações
- O Python possui uma classe e métodos muito eficientes para esse contexto
- Principais funcionalidades:
 - Criar
 - Ler
 - Escrever



Criar e Abrir Arquivos

- Existem diversas formas para abrir um arquivo de texto:
 - As mais comuns são de Leitura com o parâmetro 'r' ou com permissão de escrita 'w'

```
# leitura
```

```
f = open('nome-do-arquivo', 'r')
```

```
# escrita
```

```
f = open('nome-do-arquivo', 'w')
```

- Criar arquivo vazio

```
arquivo = open('novo-arquivo.txt', 'w')  
arquivo.close()
```

Ler Arquivos

- Para ler, existem 3 métodos:
 - `read()`, `readline()`, `readlines()`

```
arquivo = open ('file.txt')  
f = arquivo.read()  
print f  
arquivo.close()
```

```
arquivo = open ('file.txt')  
f = arquivo.readlines()  
print f  
arquivo.close()
```

```
arquivo = open ('file.txt')  
f1 = arquivo.readline()  
print f1  
f2 = arquivo.readline()  
print f2  
arquivo.close()
```

```
arquivo = open('file.txt', 'r')  
for linha in arquivo:  
    print(linha)  
arquivo.close()
```

Escrever em um arquivo

- Método `write()` para escrita
 - Se o arquivo conter algum texto o método irá sobrescrever

```
arquivo = open('file1.txt', 'w')
arquivo.write('Escrevendo em um arquivo')
arquivo.close()
```

```
arquivo = open('file.txt', 'r')
conteudo = arquivo.readlines()
conteudo.append('No FliSol\n') # insira seu conteúdo

arquivo = open('file.txt', 'w') # Abre novamente o arquivo (escrita)
arquivo.writelines(conteudo)    # escreva o conteúdo criado anteriormente nele.

arquivo.close()
```

Funções em Python

- Formato geral:

```
def nome (arg):  
    comando  
  
nome()
```

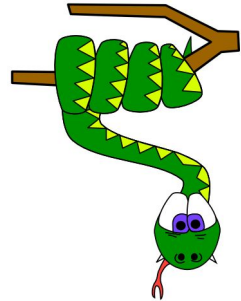
- onde:
 - nome = nome da função
 - arg = são os parâmetros da função
 - comandos = são as instruções a serem executadas
 - nome () = é a chamada da função
 - Caso necessário existe a função *return* para retornar a saída da função

Funções em Python

- Exemplo função imprimirNome

```
def imprimirNome (nome):  
    print nome  
  
imprimirNome("Rick Grimes")
```

Exemplo4



- Escrever duas funções que recebem dois argumentos
 - função Somar, para somar dois números
 - função Subtrair, para subtrair dois números

Classes e bibliotecas úteis

- `os.system`: é possível interagir com o sistema operacional

```
import os

cmd = "python --version"

saida = os.system(cmd)
```

- `math()`: funções matemáticas

```
from math import *

x = 4
print sin(x)
print cos(x)
```

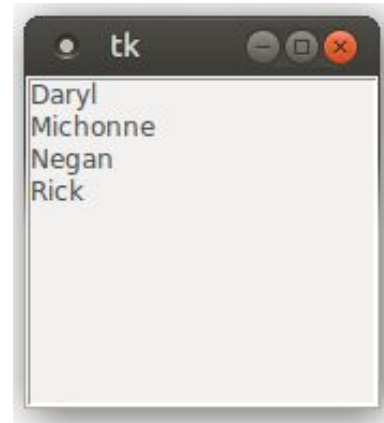

Classes e bibliotecas úteis

- Tkinter: biblioteca de GUI padrão

```
from Tkinter import *
window = Tk()

lista = 'Rick Negan Michonne Daryl'.split()
listb = Listbox(window)
for item in lista:
    listb.insert(0,item)

listb.pack()
window.mainloop()
```



Classes e bibliotecas úteis

- Matplotlib: permite a criação de diferentes tipos de gráficos

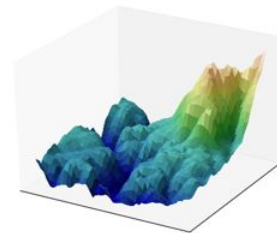
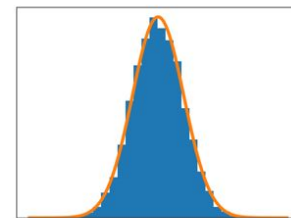
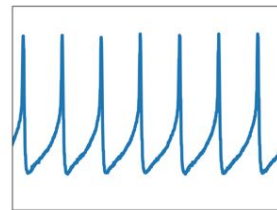
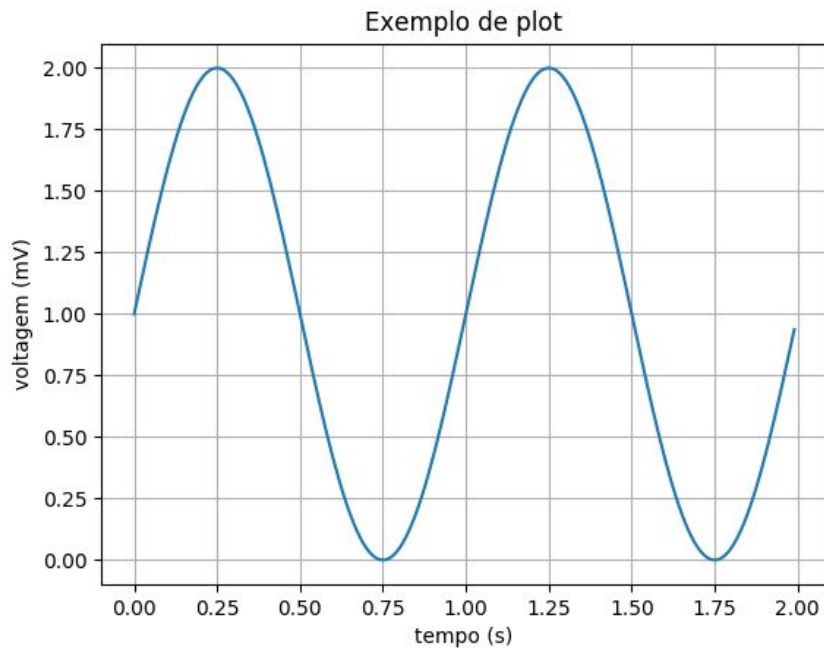
```
import matplotlib.pyplot as plt
import numpy as np

t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2*np.pi*t)
plt.plot(t, s)

plt.xlabel('tempo (s)')
plt.ylabel('voltagem (mV)')
plt.title('Exemplo de plot')
plt.grid(True)
plt.savefig("test.png")
plt.show()
```

Classes e bibliotecas úteis

- Matplotlib:



Classes e bibliotecas úteis

- Aprendizagem de máquina

scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

TensorFlow™

Google.org

An open-source machine learning framework for everyone



Welcome to Spotipy!

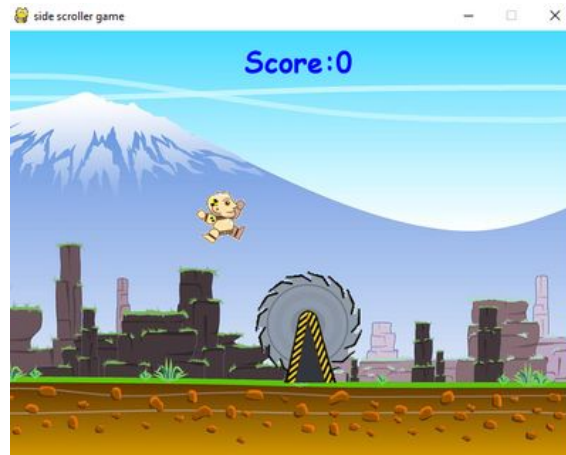
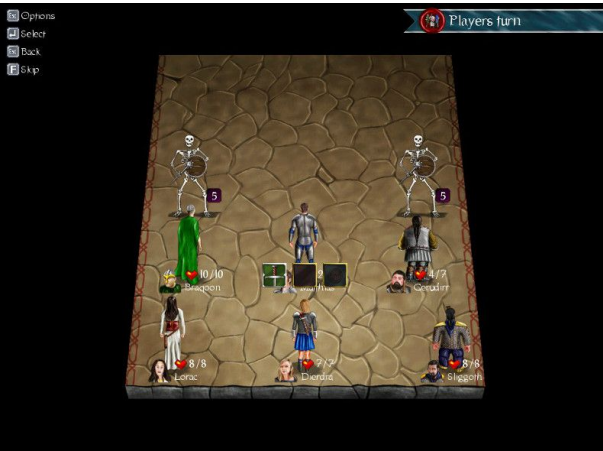
Spotipy is a lightweight Python library for the [Spotify Web API](#). With Spotipy you get full access to all of the music data provided by the Spotify platform.

Tweepy

An easy-to-use Python library for accessing the Twitter API.

Classes e bibliotecas úteis

- Pygame



Referências

- www.python.org
- www.pygame.org
- www.kivy.org
- www.matplotlib.org
- www.scikit-learn.org
- www.tensorflow.org
- www.spotipy.readthedocs.io
- www.tweepy.org

Obrigado e bom estudo!



Prof. Francisco Carlos
fcarlosmonteiro@gmail.com

