Python 101 para Engenheiros

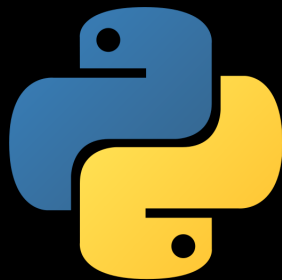
Laís Brum Menezes

Marcos Alves dos Santos

14/08/19 e 21/08/19

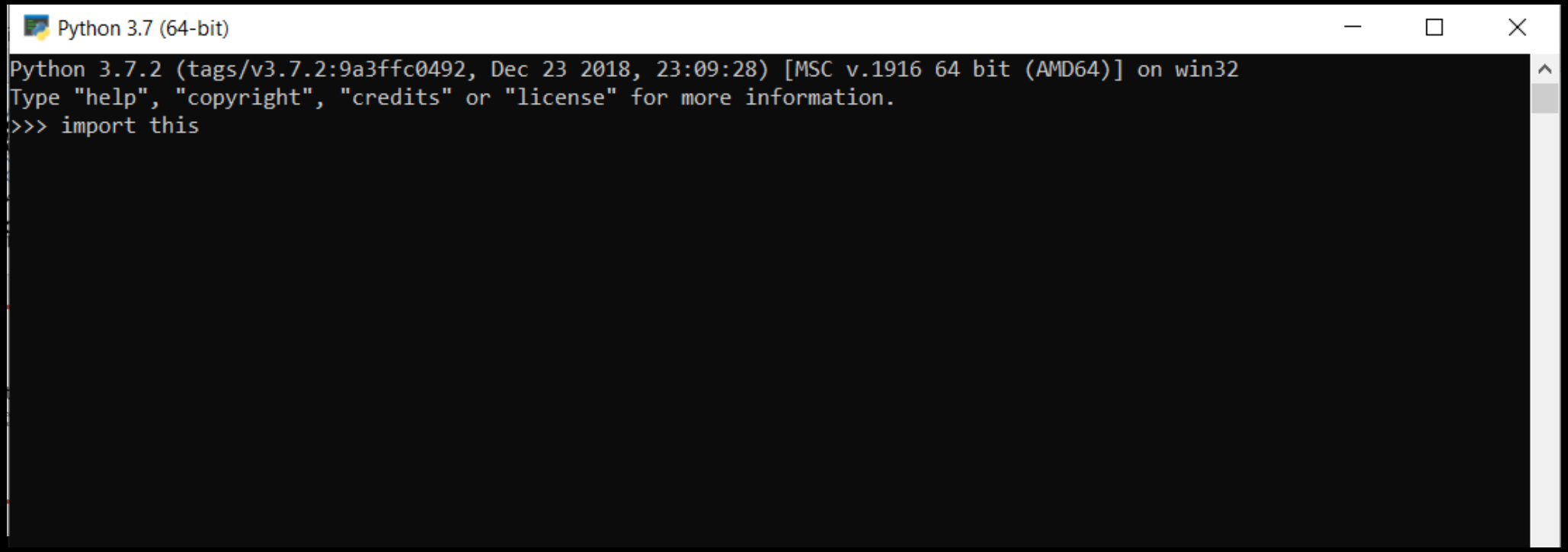
Resumo do minicurso

- UNIDADE 1 – Introdução ao Python:
  1. História
  2. Aplicações
  3. CompilaçãoIDE
  4. Por que Python?
  5. Vantagens
  6. Sintaxe da Linguagem
- UNIDADE 2 – Download e Instalação:
  1. Download e Instalação do interpretador
  2. Download e Instalação do Pycharm
- UNIDADE 3 – Conceitos Básicos:
  1. Hello World
  2. Variáveis
  3. Expressões Aritméticas
  4. Expressões Lógicas
  5. Entrada e Saída de dados
  6. Atribuição
  7. Bibliotecas
- UNIDADE 4 – Desvios Condicionais:
  1. If, Elif e Else
- UNIDADE 5 – Laços de Repetição:
  1. For
  2. While
- UNIDADE 6 – Strings e Conjuntos de Dados:
  1. Strings
  2. Listas, Tuplas e Dicionários
  3. Matrizes
- UNIDADE 7 – Funções:
  1. Funções
- UNIDADE 8 – Programação Orientada a Objetos:
  1. Abstração
  2. Encapsulamento
  3. Herança
  4. Polimorfismo



“import this”

Metas do Python resumidas por Tim Peters em um texto chamado “Zen of Python”.

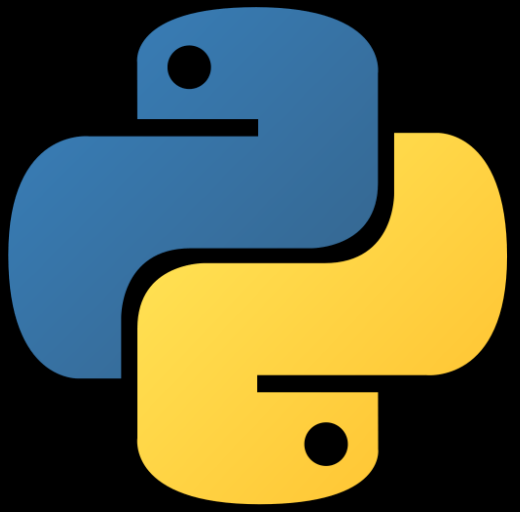

Python 3.7 (64-bit)
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import this

Python 101

UNIDADE 1 – Introdução ao Python:

1. História
2. Aplicações
3. Compilação
4. IDE
5. Por que Python?
6. Vantagens
7. Sintaxe da Linguagem

História



História

CWI
Distributed and Interactive Systems

abc LANGUAGES

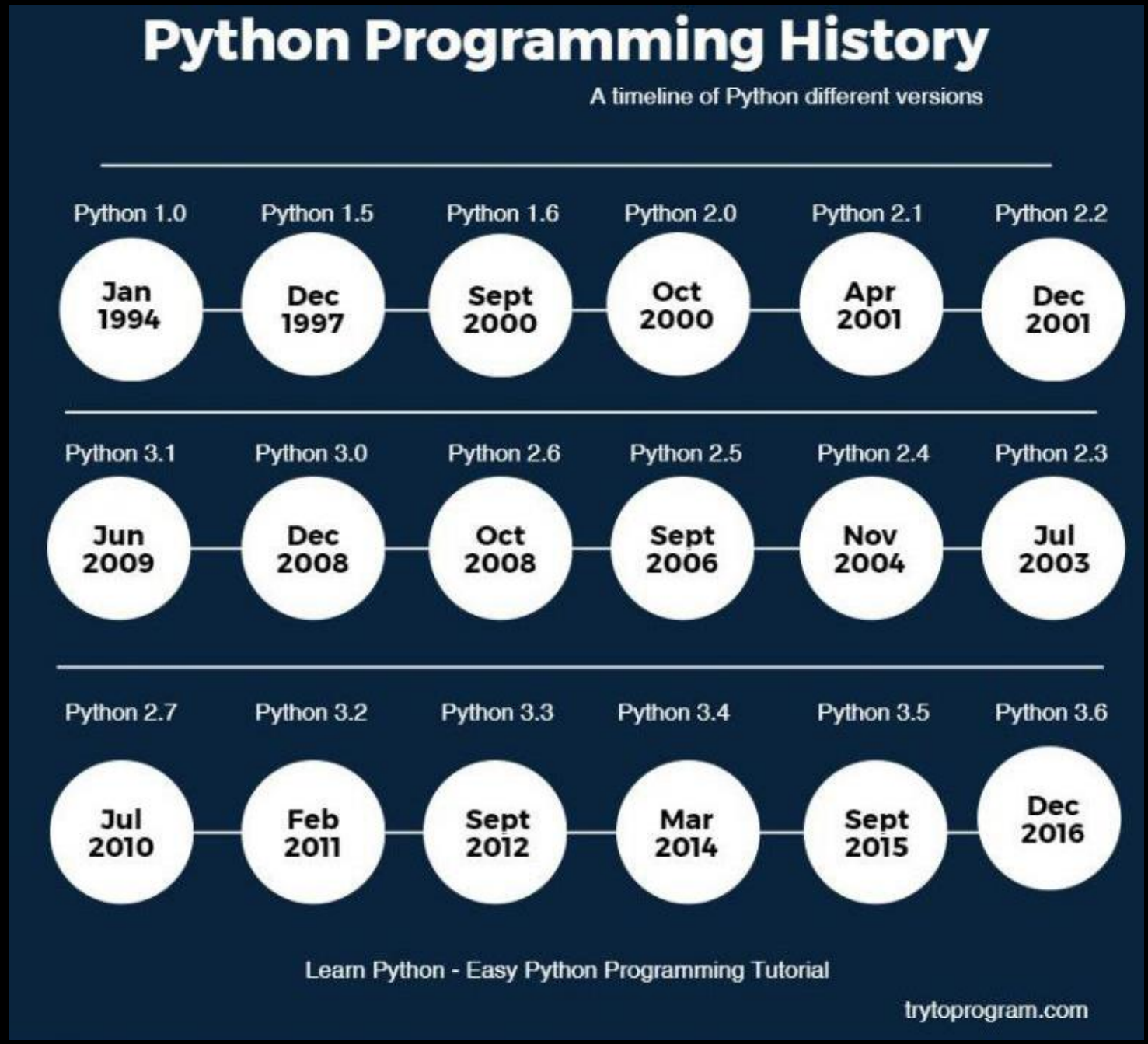
Década de 1980, Van Rossum usava o CWI (lab) para trabalhar em uma linguagem chamada "ABC", juntamente com o S.O. "AMOEBA". Nesse tempo, Rossum começou a pensar em uma nova linguagem que possuísse uma sintaxe semelhante à da "ABC" e com o acesso semelhante ao do "AMOEBA".

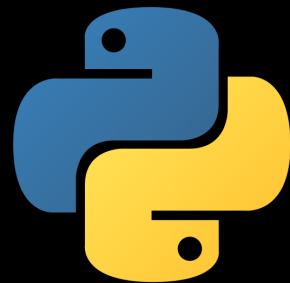
Portanto, iniciando-se na década de 1980,e concluindo em 1991, foi lançada a primeira versão da linguagem "Python" 1.0.

CURIOSIDADE

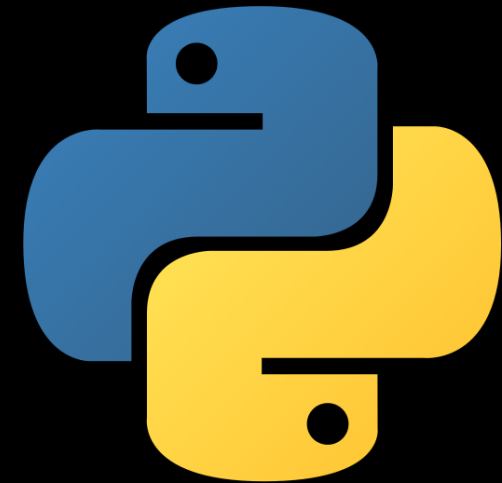
Segundo o site "Trytoprogram", na década de 70 existia um famoso programa de comédia na BBC, chamado "Monty Python's Flying Circus", do qual Rossum era fã. Então, quando o projeto estava concluído, esse foi nomeado "Python".

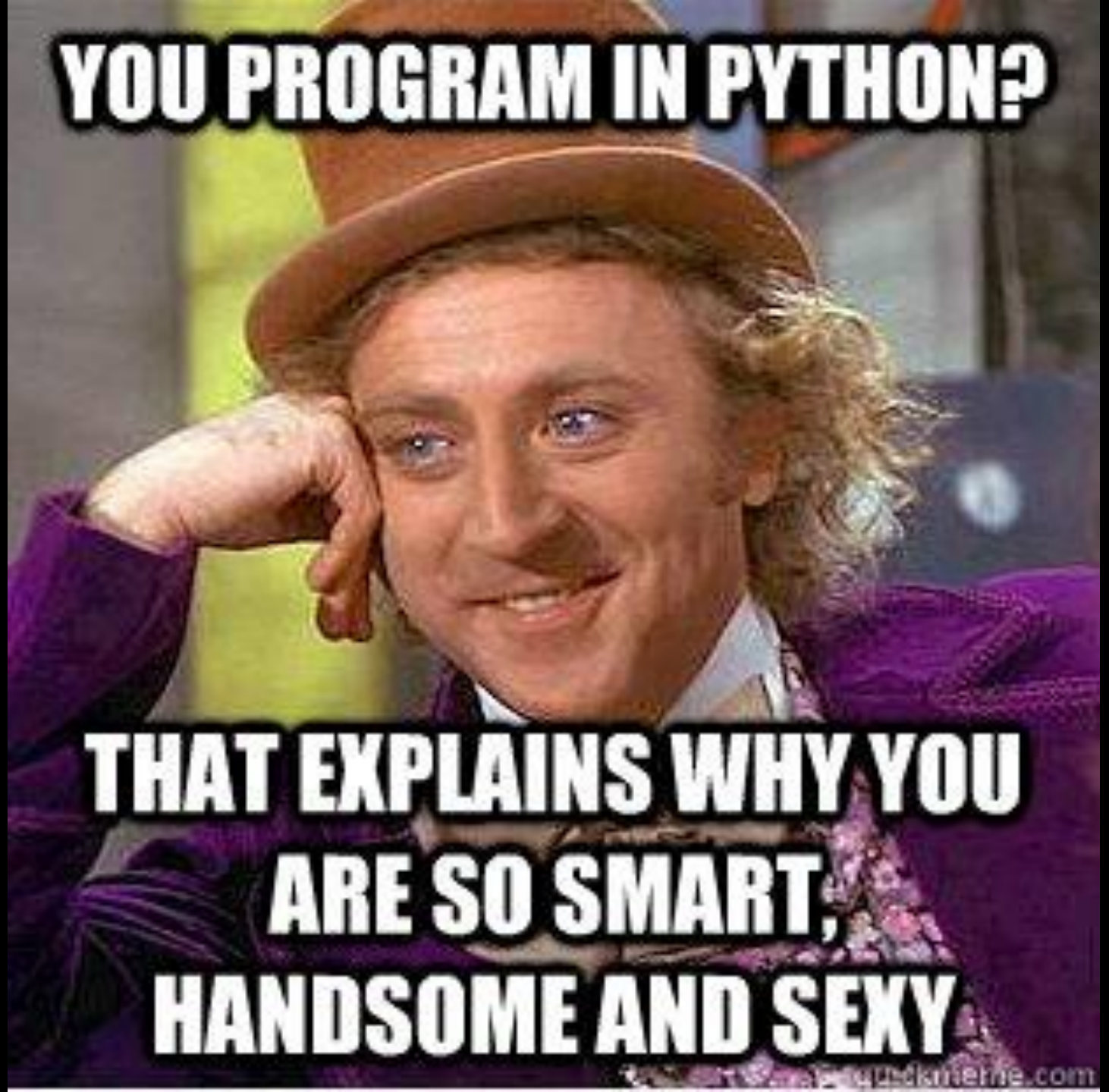
História


Python Programming History
A timeline of Python different versions
Python 1.0
Jan 1994
Python 1.5
Dec 1997
Python 1.6
Sept 2000
Python 2.0
Oct 2000
Python 2.1
Apr 2001
Python 2.2
Dec 2001
Python 3.1
Jun 2009
Python 3.0
Dec 2008
Python 2.6
Oct 2008
Python 2.5
Sept 2006
Python 2.4
Nov 2004
Python 2.3
Jul 2003
Python 2.7
Jul 2010
Python 3.2
Feb 2011
Python 3.3
Sept 2012
Python 3.4
Mar 2014
Python 3.5
Sept 2015
Python 3.6
Dec 2016
Learn Python - Easy Python Programming Tutorial
trytoprogram.com

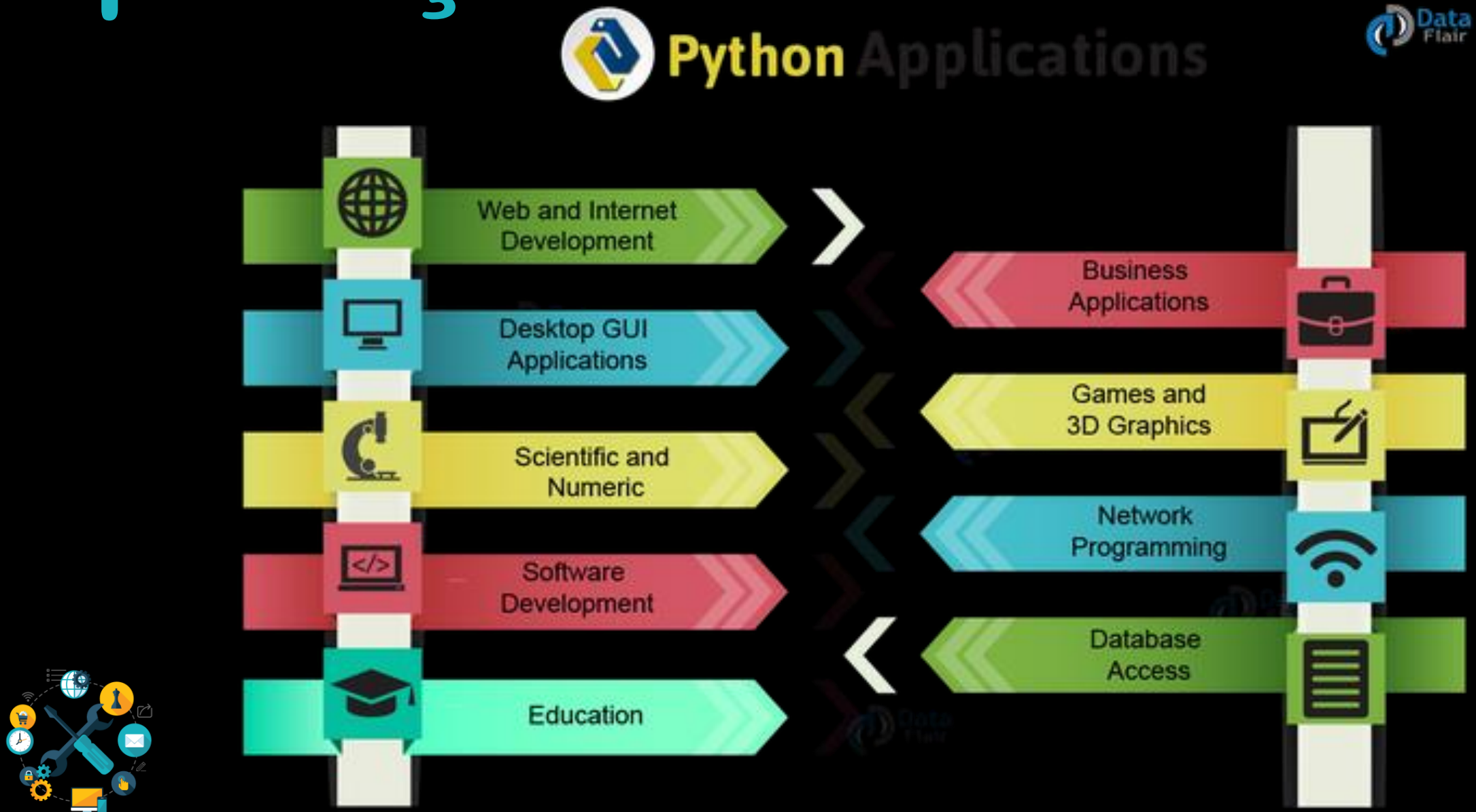


Aplicação




YOU PROGRAM IN PYTHON?
THAT EXPLAINS WHY YOU ARE SO SMART, HANDSOME AND SEXY

Aplicação


Python Applications
Data Flair
Web and Internet Development
Desktop GUI Applications
Scientific and Numeric
Software Development
Education
Business Applications
Games and 3D Graphics
Network Programming
Database Access

Aplicação

Desenvolvimento Web e Internet

O Python oferece muitas opções para desenvolvimento na web :

- Frameworks como Django e Pyramid .
- Micro-estruturas como Flask e Bottle .
- Sistemas avançados de gerenciamento de conteúdo, como o Plone e o django CMS .

A biblioteca padrão do Python suporta muitos protocolos da Internet:

- HTML e XML
- JSON
- E-mail processing.
- Suporte para FTP , IMAP e outros protocolos da Internet .
- Interface de soquete fácil de usar .

E o Índice de Pacotes ainda tem mais bibliotecas:

- Solicitações , uma poderosa biblioteca cliente HTTP.
- BeautifulSoup , um analisador de HTML que pode lidar com todos os tipos de HTML estranho.
- Feedparser para analisar feeds RSS / Atom.
- Paramiko , implementando o protocolo SSH2.
- Twisted Python , um framework para programação de rede assíncrona.

Científico e Numérico

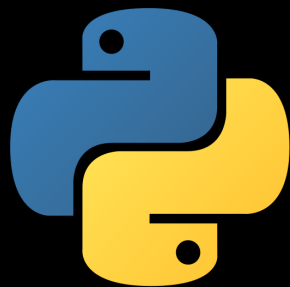
Python é amplamente utilizado em computação científica e numérica :

- SciPy é uma coleção de pacotes para matemática, ciências e engenharia.
- Pandas é uma biblioteca de análise e modelagem de dados.
- O IPython é um poderoso shell interativo que apresenta fácil edição e gravação de uma sessão de trabalho e suporta visualizações e computação paralela.
- O Curso de Carpintaria de Software ensina habilidades básicas para computação científica, executando bootcamps e fornecendo materiais de ensino de acesso aberto.

Educação

O Python é uma linguagem excelente para o ensino de programação, tanto no nível introdutório quanto em cursos mais avançados.

- Livros como Como Pensar como um Cientista da Computação , Programação em Python: Uma Introdução à Ciência da Computação e Programação Prática .
- O Grupo de Interesse Especial em Educação é um bom lugar para discutir questões de ensino.



Aplicação

GUIs de desktop

A biblioteca Tk GUI está incluída na maioria das distribuições binárias do Python.

Alguns kits de ferramentas que podem ser usados em várias plataformas estão disponíveis separadamente:

- wxWidgets
- Kivy , para escrever aplicativos multitouch.
- Qt via pyqt ou pyside

Toolkits específicos da plataforma também estão disponíveis:

- GTK +
- Microsoft Foundation Classes através das extensões do win32

Desenvolvimento de software

O Python é frequentemente usado como uma linguagem de suporte para desenvolvedores de software, para controle e gerenciamento de construções, testes e de muitas outras maneiras.

- SCons para controle de compilação.
- Buildbot e Apache Gump para compilação e testes contínuos automatizados.
- Roundup ou Trac para rastreamento de bugs e gerenciamento de projetos.

Aplicativos de negócios

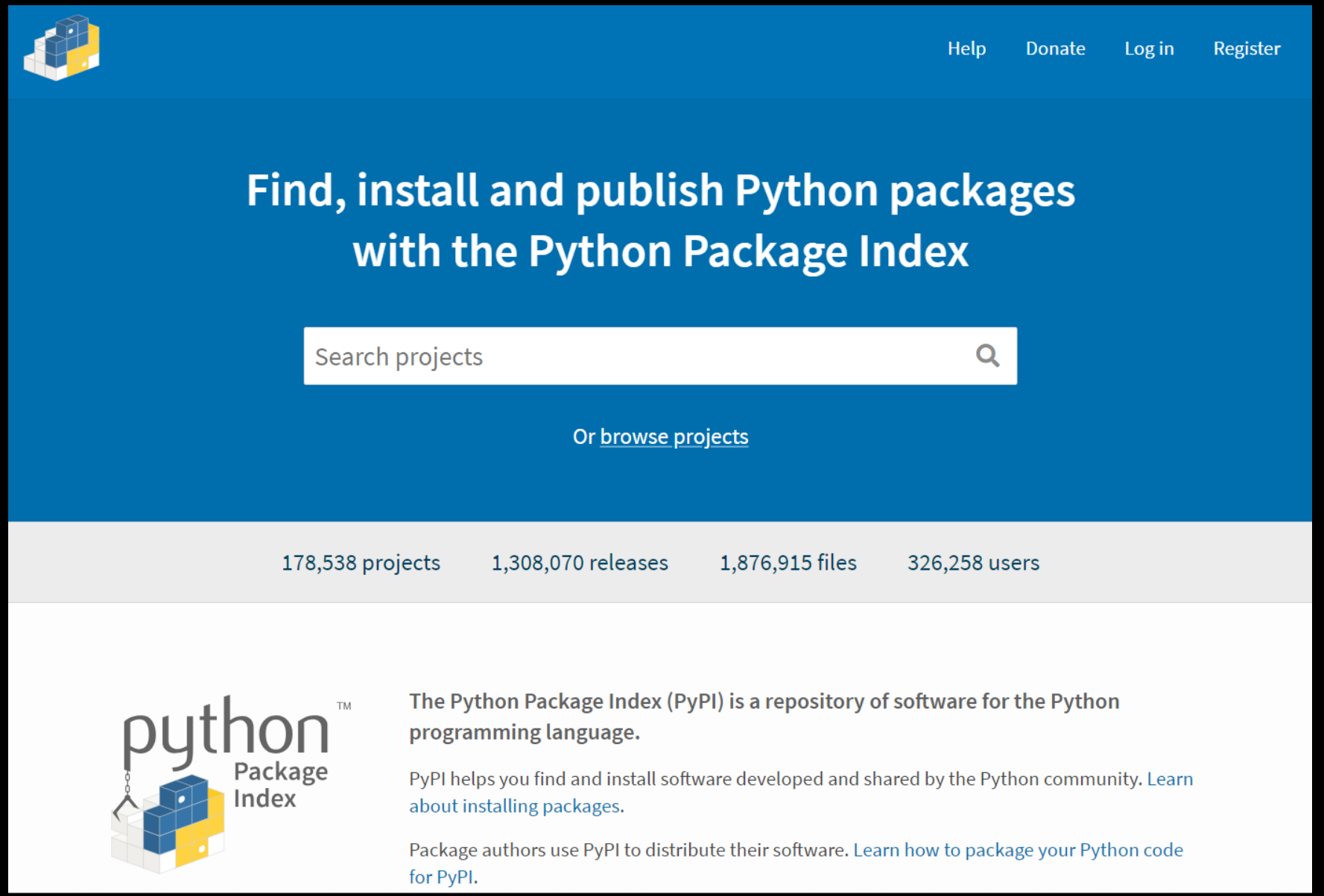
O Python também é usado para construir sistemas de ERP e e-commerce:

- Odoo é um software de gerenciamento completo que oferece uma variedade de aplicativos de negócios que formam um conjunto completo de aplicativos de gerenciamento corporativo.
- O Tryton é uma plataforma de aplicativos de uso geral de alto nível e três níveis.

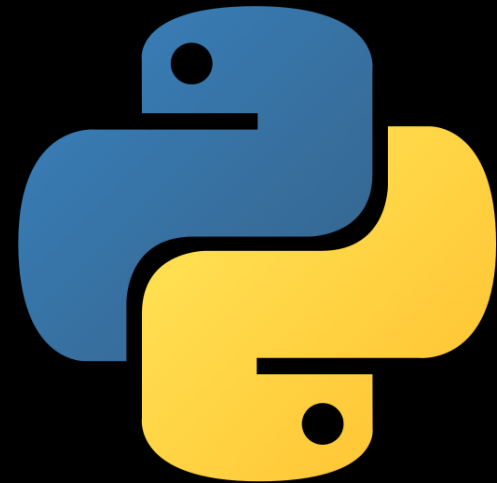
Aplicação

Python é usado em muitos “applications domains".

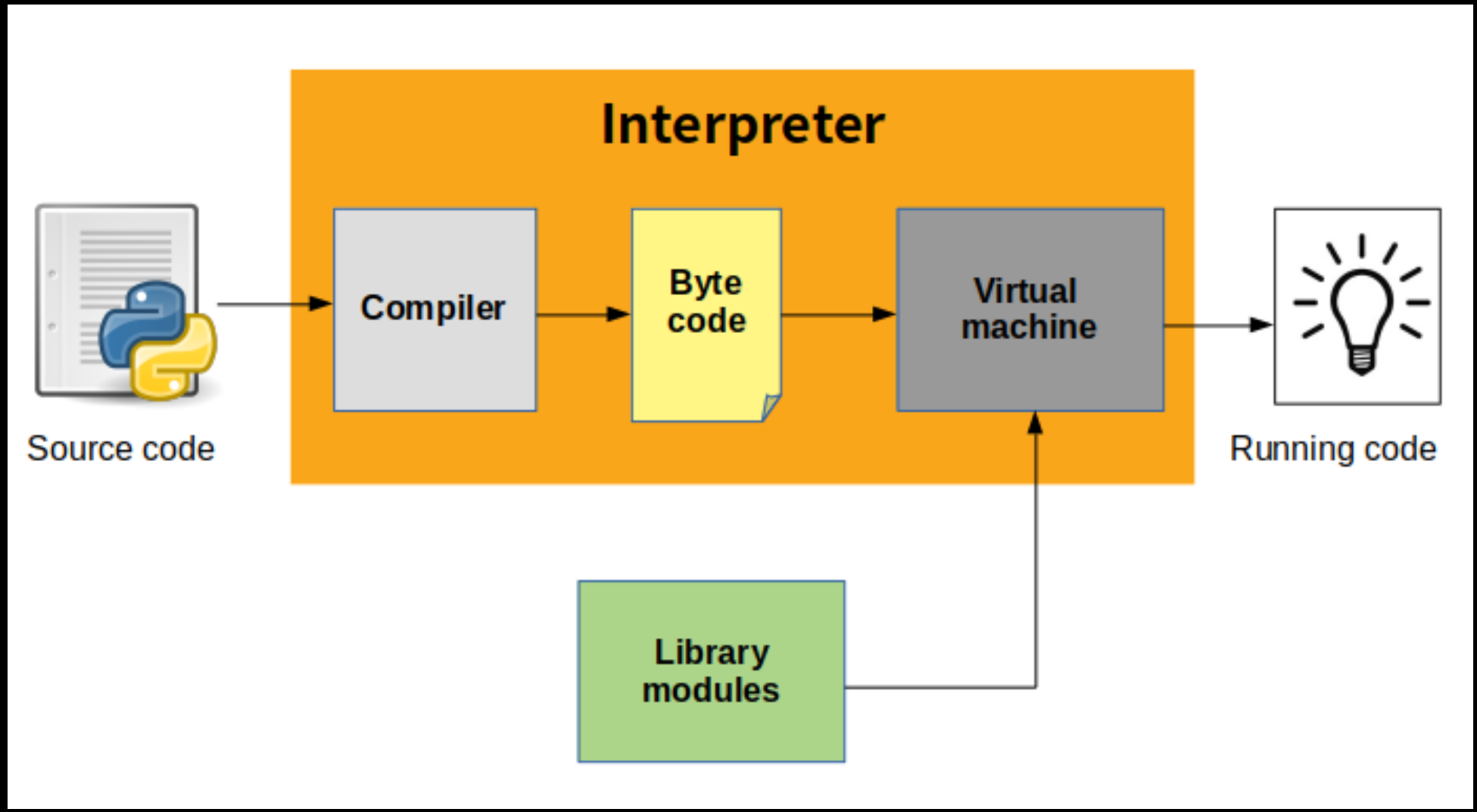
The Python Package Index lists thousands of third party modules for Python.

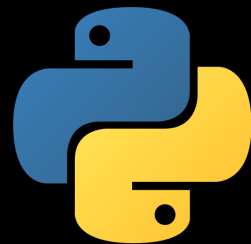


Compilação



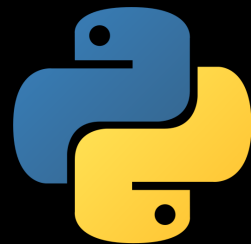
Compilação


Interpreter
Compiler
Byte code
Virtual machine
Source code
Running code
Library modules

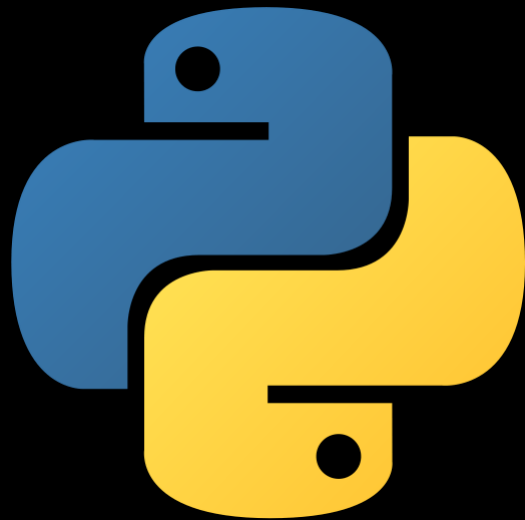


Compilação

Contudo, um código “.py” pode ser “compilado” utilizando o módulo “py2exe” (‘gambiarra’). Ou usando o “Pyinstaller” junto com o “PyWin32” na instalação padrão (quando vai instalar o python).



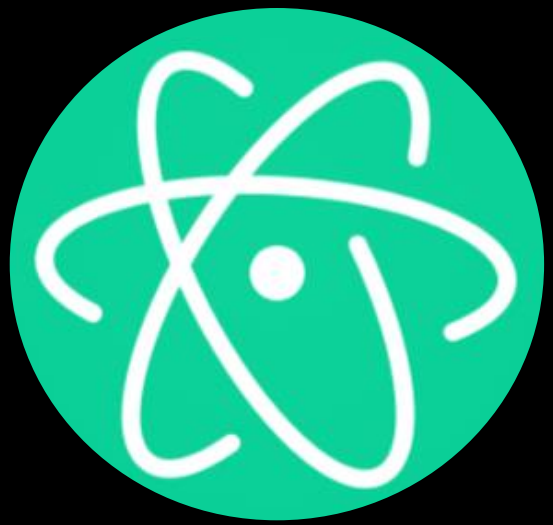
IDE



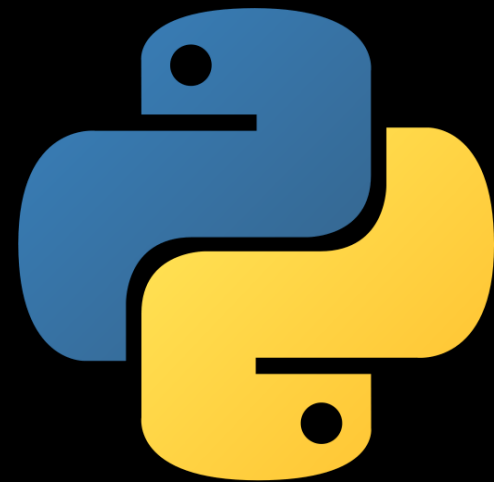
IDE

WING PYTHON IDE

THE INTELLIGENT DEVELOPMENT ENVIRONMENT FOR PYTHON



Motivação

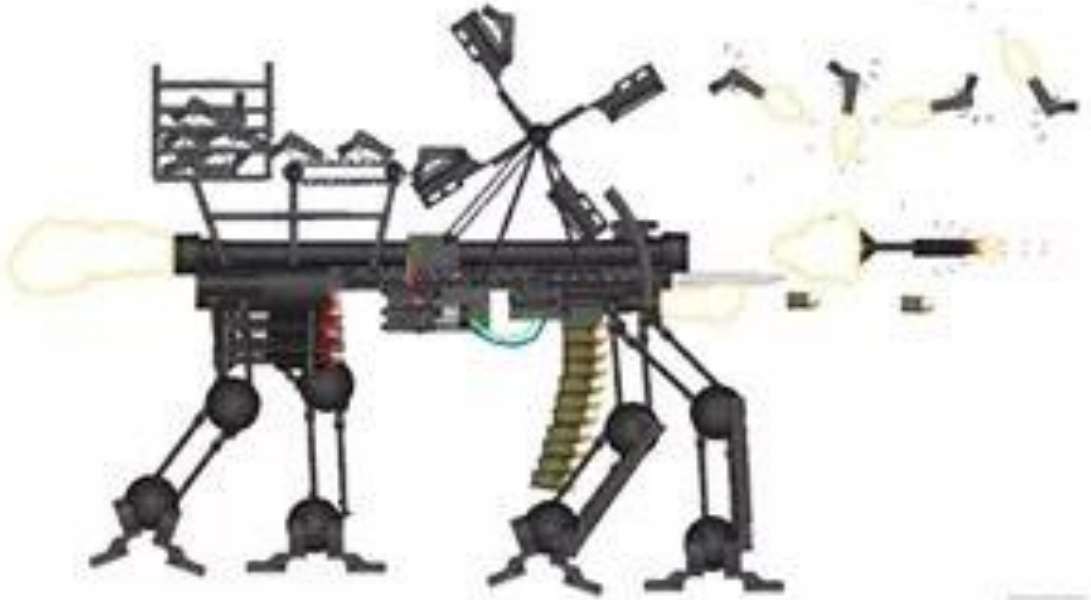


Assembly

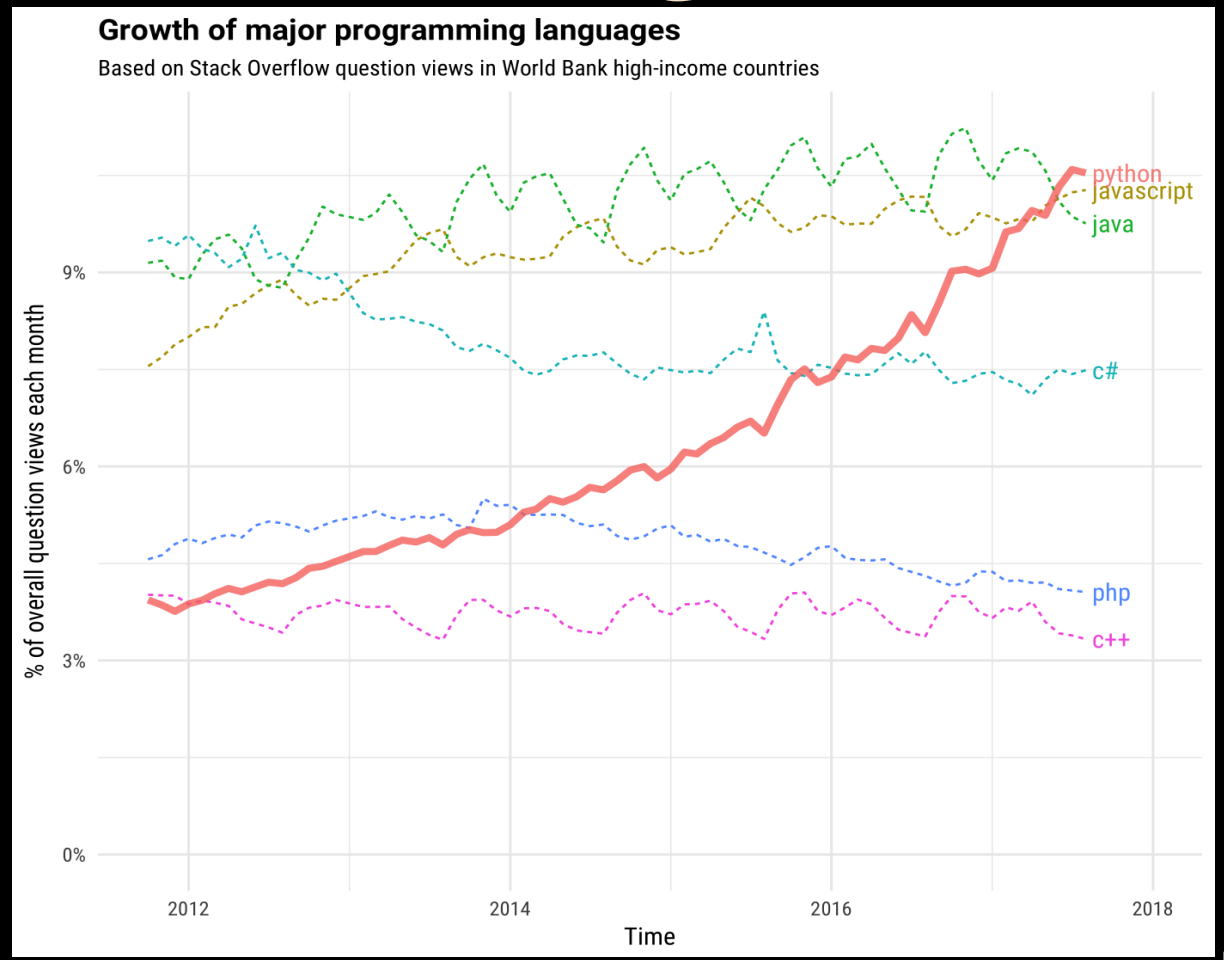
C++

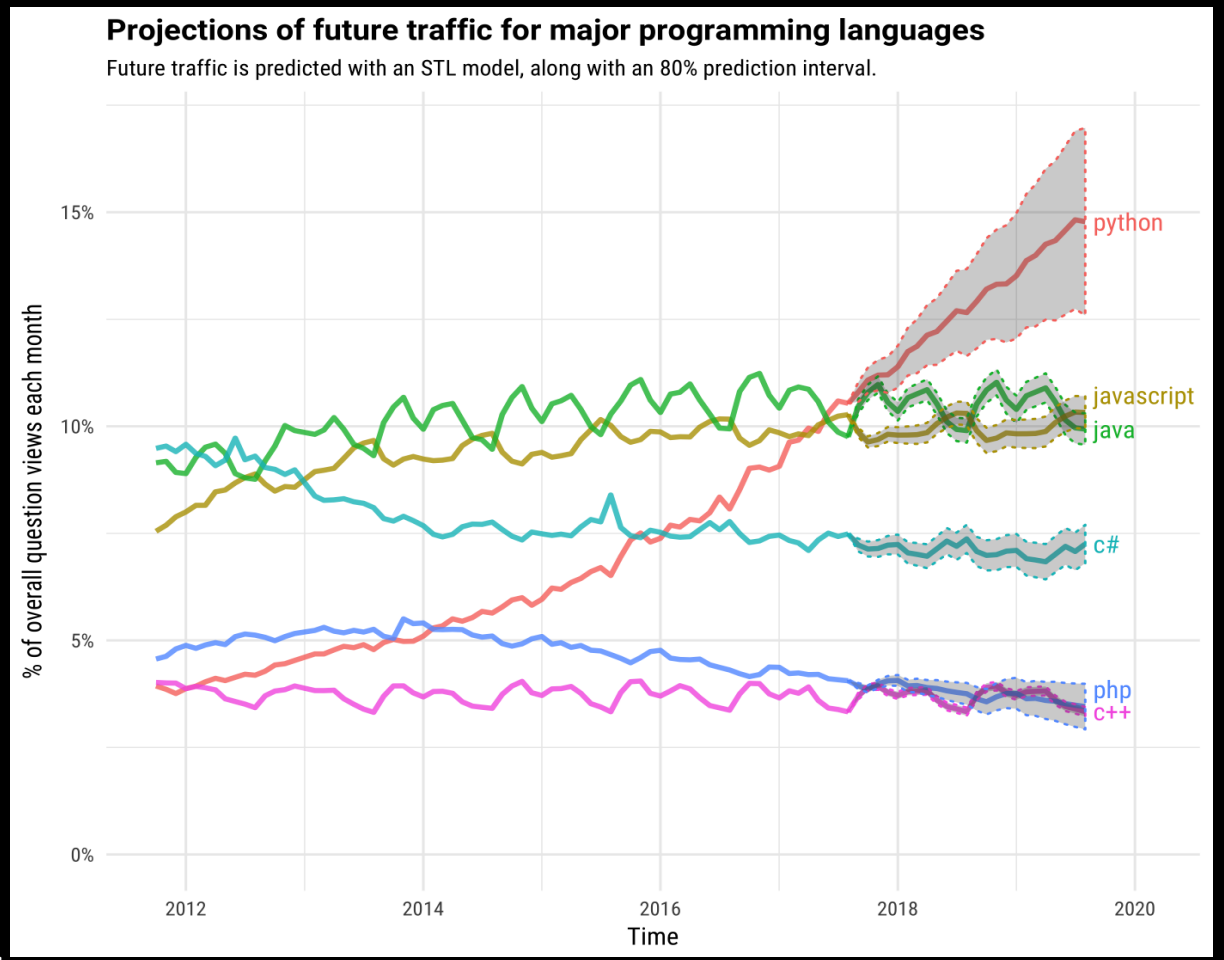
Python

C



Motivação



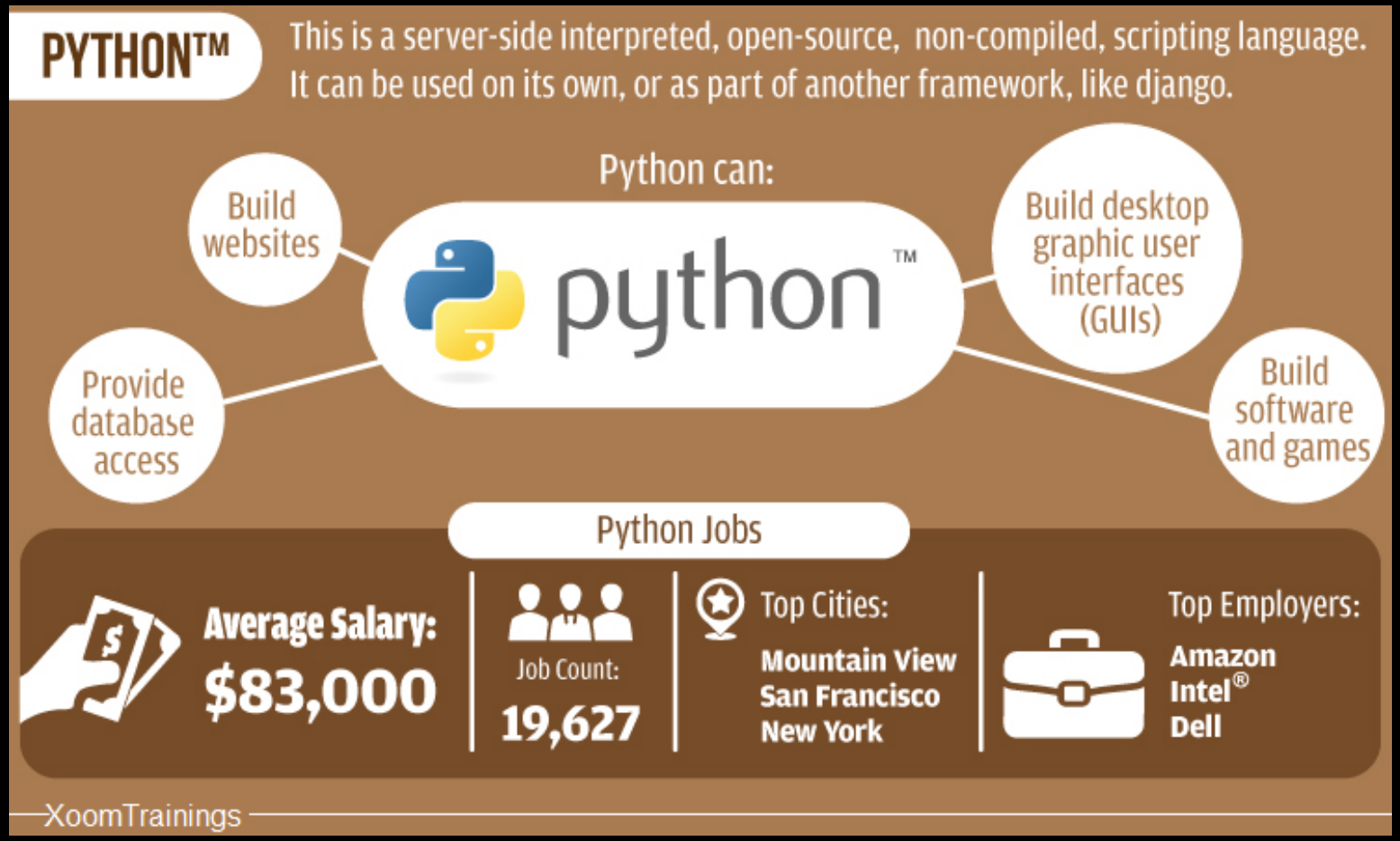


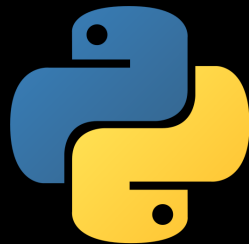
FONTE: Stackoverflow. By: David Robinson, on September 6, 2017.

Motivação

Talvez, de todas as linguagens, esta seja a “melhor” (abrange muitas áreas, é simples e poderosa).

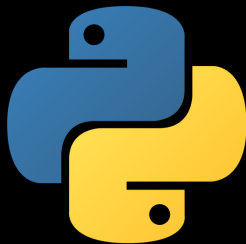
- Fácil de se aprender;
- Python pode ser usado em inúmeras áreas;
- É uma poderosa linguagem;
- Imensa comunidade ;
- Open Source.





Motivação

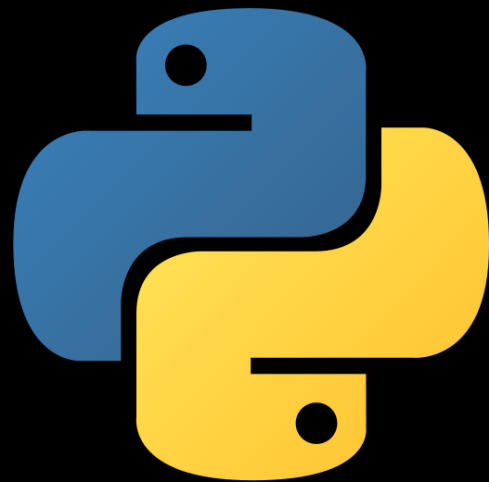
Talvez, de todas as linguagens, esta seja a “melhor” (abrange muitas áreas, é simples e poderosa).



Motivação

Talvez, de todas as linguagens, esta seja a “melhor” (abrange muitas áreas, é simples e poderosa).

Prós e contras



Prós e contras

Desvantagens:

- **Difficulty in Using Other Languages**
  Talvez, ao se aprender python a pessoa pode ficar “mal acostumada”.
- **Weak in Mobile Computing**
  Python tem forte presença em aplicações desktop e server platforms, mas é pouco visto em aplicações mobile.

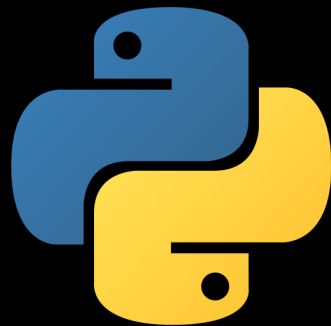
Desvantagens:

- **Gets Slow in Speed**
  Se velocidade é um requisito muito importante para determinada aplicação, então python não é a melhor opção. Isso se deve ao fato de ser uma linguagem interpretada.
- **Run-time Errors**
  Devido ao fato de ser uma linguagem digitada dinamicamente, podem existir algumas restrições relatadas por alguns desenvolvedores em algumas áreas.

Desvantagens:

- **Underdeveloped Database Access Layers**
  O acesso ao banco de dados do Python é considerada primitiva, em relação a tecnologias atuais.

FONTE: https://medium.com/.
– Medium Corporation.



Prós e contras

Vantagens:

Extensas bibliotecas de suporte;

Recursos de integração;

Grande produtividade ao programador;

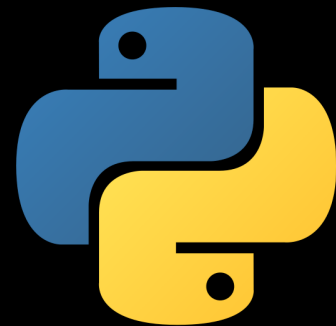
Ampla gama de aplicações;

Programação orientada a objetos e programação funcional;

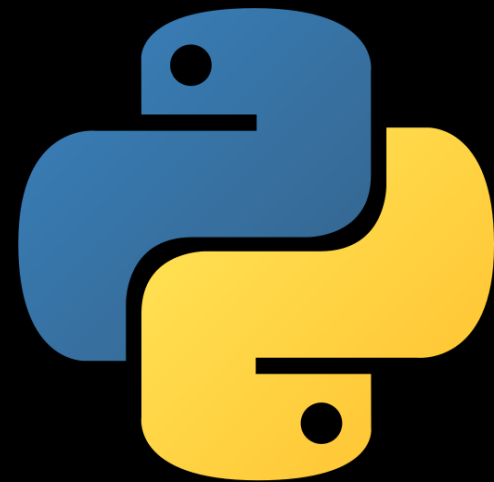
Completamente gratuita e Open-Source;

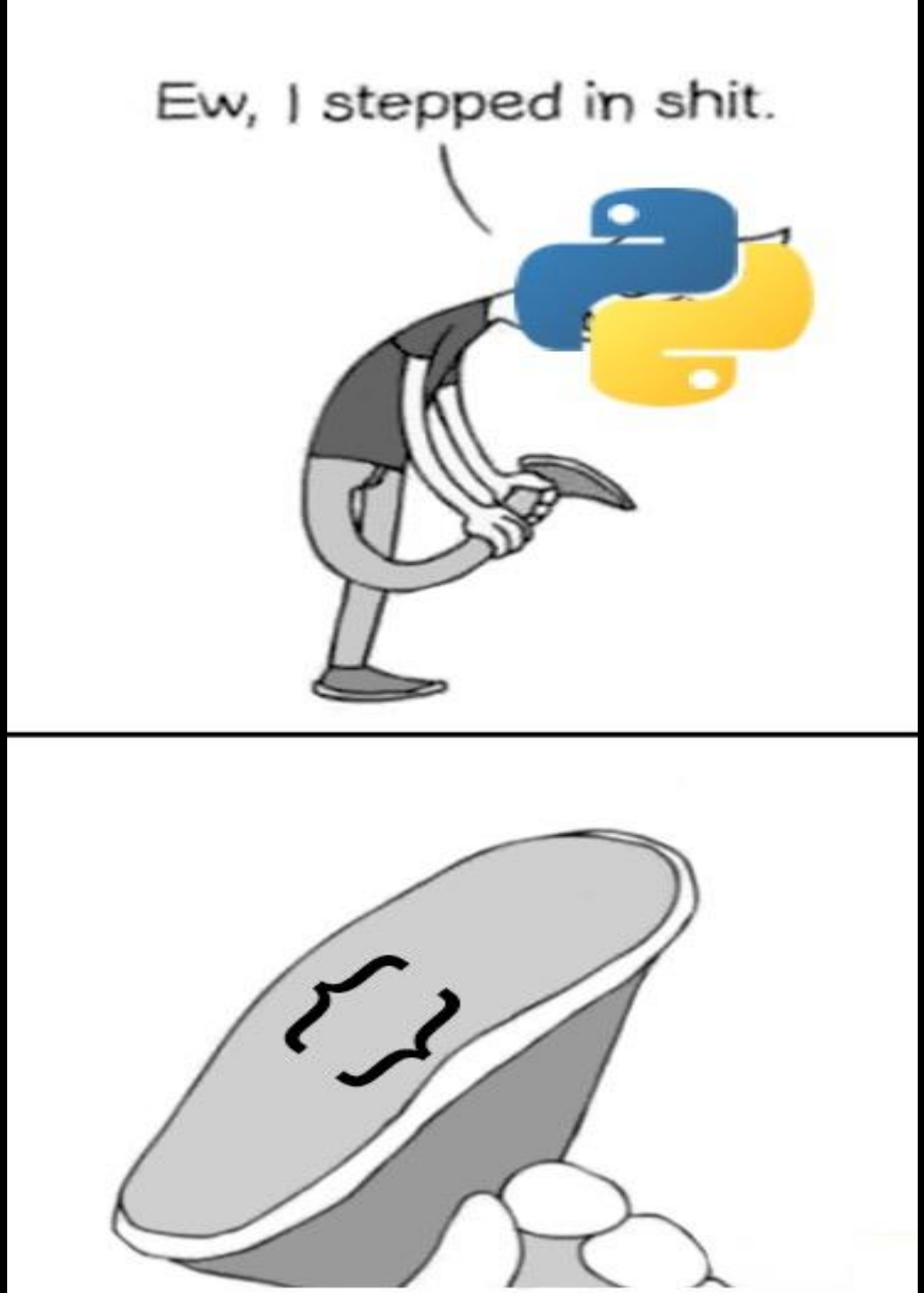
Vantagens:

- **Extensível;**
- **Simples e fácil (writing and Reading);**
- **Portável;**
- **Grande comunidade ao redor do mundo;**

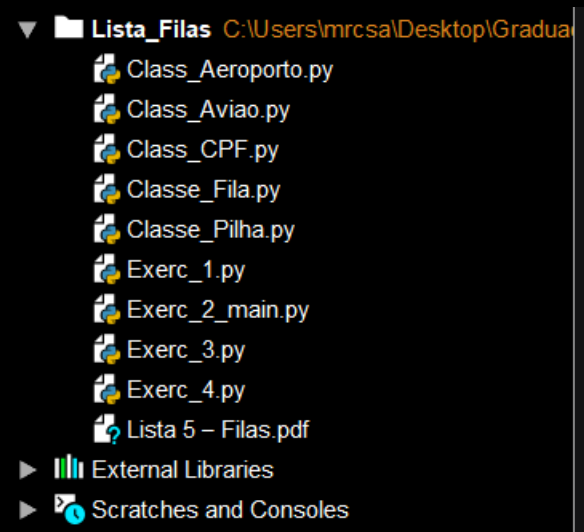


Sintaxe

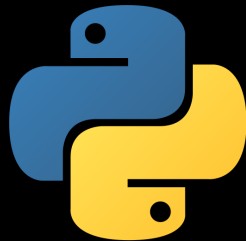



Ew, I stepped in shit.
{ }

Sintaxe




```python
from Class_Aeroporto import Aeroporto

sair = False
a = Aeroporto()
while not sair:
    _1 = input('Adicionar um novo avião à lista? [s ou S / n ou N] ')
    if _1 != 'n' and _1 != 'N':
        print('_1 = ' + _1)
        a.aviaoParaFila()
        _2 = input('Autorizar decolagem do primeiro veículo da fila? ')
        if _2 == 'S' or _2 == 's':
            print('_2 = ' + _2)
            a.autorizaDecolagem()
        else:
            pass
        _3 = input('Dejesa listar todos os aviões que estão na lista de espera? ')
        if _3 == 'S' or _3 == 's':
            print('_3 = ' + _3)
            a.mostraFilaDeEspera()
        else:
            pass
        _4 = input('Deseja listar as caracteristas do primeiro avião da fila? ')
        if _4 == 'S' or _4 == 's':
            print('_4 = ' + _4)
            print('Primeiro veículo da fila: ', a.caracteristicas_PrimeiroDaFila())
        else:
            pass

    e = input('\n\nEXIT? [OK] ')
    if e == 'OK':
        sair = True
    else:
        sair = False

```



Sintaxe

advanced › path_tutorial.py

Project

advanced C:\Users\mrcsa\Desktop\Projeto_Modelagem\Exemp
- path_tutorial.py
- patheffects_guide.py
- transforms_tutorial.py

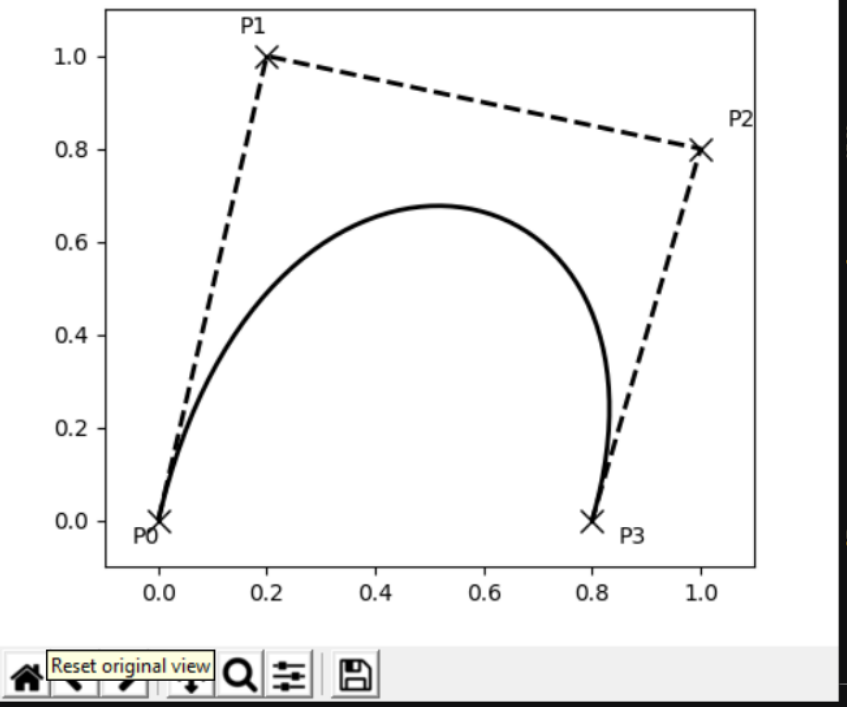
External Libraries

Scratches and Consoles

path_tutorial.py | patheffects_guide.py | transforms_tutorial.py

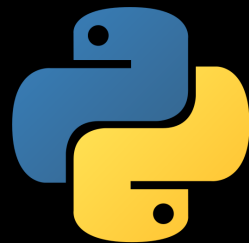
```
"""
=============
Path Tutorial
=============

Defining paths in your Matplotlib visualization.

The object underlying all of the :mod:`matplotlib.patch` objects is
the :class:`~matplotlib.path.Path`, which supports the standard set of
moveto, lineto, curveto commands to draw simple and compound outlines
consisting of line segments and splines.  The ``Path`` is instantiated
with a (N,2) array of (x,y) vertices, and a N-length array of path
codes.  For example to draw the unit rectangle from (0,0) to (1,1), we
could use this code
"""

import matplotlib.pyplot as plt
from matplotlib.path import Path
import matplotlib.patches as patches

verts = [
    (0., 0.),  # left, bottom
    (0., 1.),  # left, top
    (1., 1.),  # right, top
    (1., 0.),  # right, bottom
    (0., 0.),  # ignored
```

Figure 1

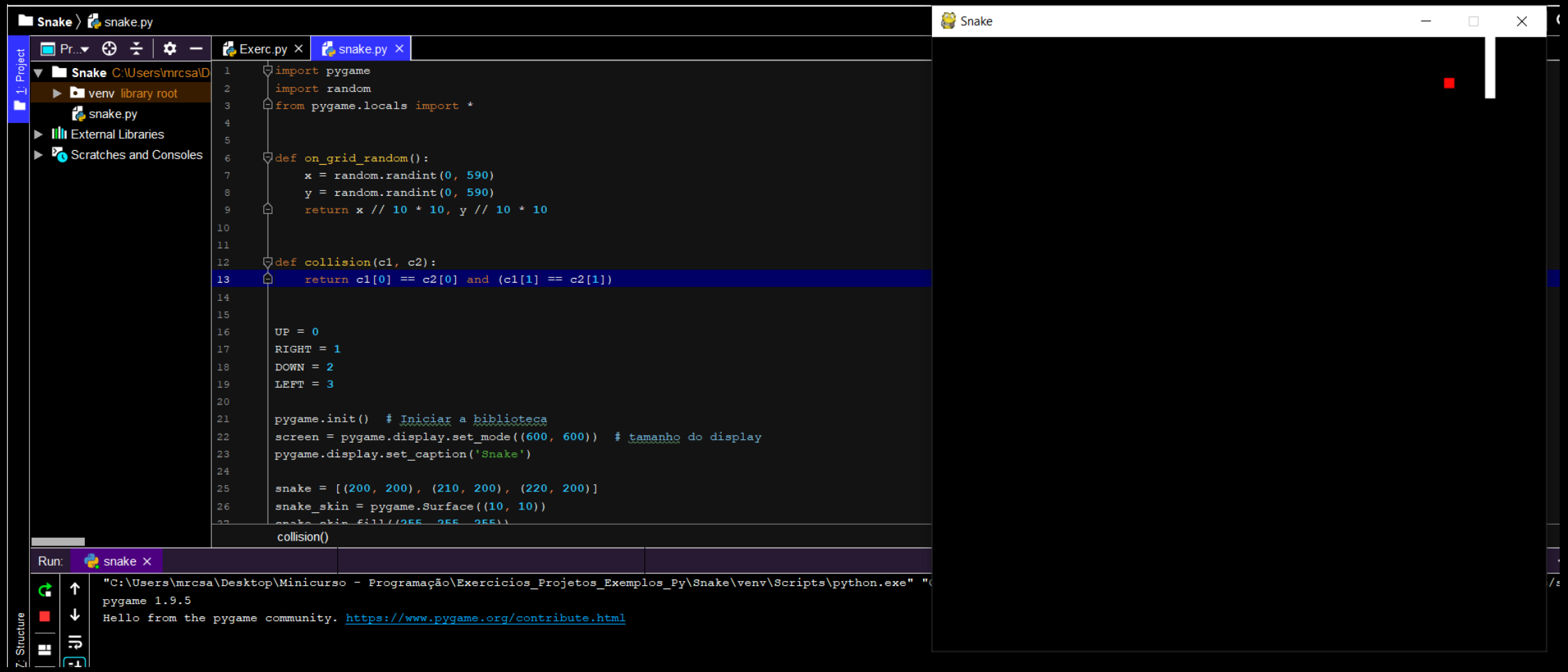


Run: path_tutorial

```
C:\Users\mrcsa\AppData\Local\Programs\Python\Python37\python.exe C:/Users/mrcsa/Desktop/Projeto_Modelagem/Exemplos_Biblioteca_MATPLOTLIBouSCIPY/advanced/path_tutorial.py
```



Sintaxe


Snake 〉 snake.py
Snake
Exerc.py
snake.py
1 Project
Snake C:\Users\mrcsa\D
venv library root
snake.py
External Libraries
Scratches and Consoles
import pygame
import random
from pygame.locals import *
def on_grid_random():
x = random.randint(0, 590)
y = random.randint(0, 590)
return x // 10 * 10, y // 10 * 10
def collision(c1, c2):
return c1[0] == c2[0] and (c1[1] == c2[1])
UP = 0
RIGHT = 1
DOWN = 2
LEFT = 3
pygame.init() # Iniciar a biblioteca
screen = pygame.display.set_mode((600, 600)) # tamanho do display
pygame.display.set_caption('Snake')
snake = [(200, 200), (210, 200), (220, 200)]
snake_skin = pygame.Surface((10, 10))
collision()
Run:
snake
"C:\Users\mrcsa\Desktop\Minicurso - Programação\Exercicios_Projetos_Exemplos_Py\Snake\venv\Scripts\python.exe"
pygame 1.9.5
Hello from the pygame community. https://www.pygame.org/contribute.html
Z: Structure
Snake

Sintaxe

Exemplo_Minicurso.py

```
1  print('Hello, MiniCurso! :D ')
2
```

```
Hello, MiniCurso! :D

Process finished with exit code 0
```

Exemplo_Minicurso.py

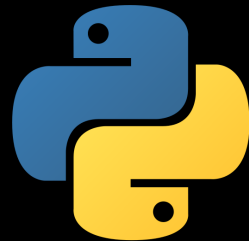
```
1  a = 5
2  print('a = ', a)
3
4  a = 'Viram como isso? '
5  print('a = ' + a)
6
```

```
a =  5
a = Viram como isso?

Process finished with exit code 0
```

Exemplo_Minicurso.py

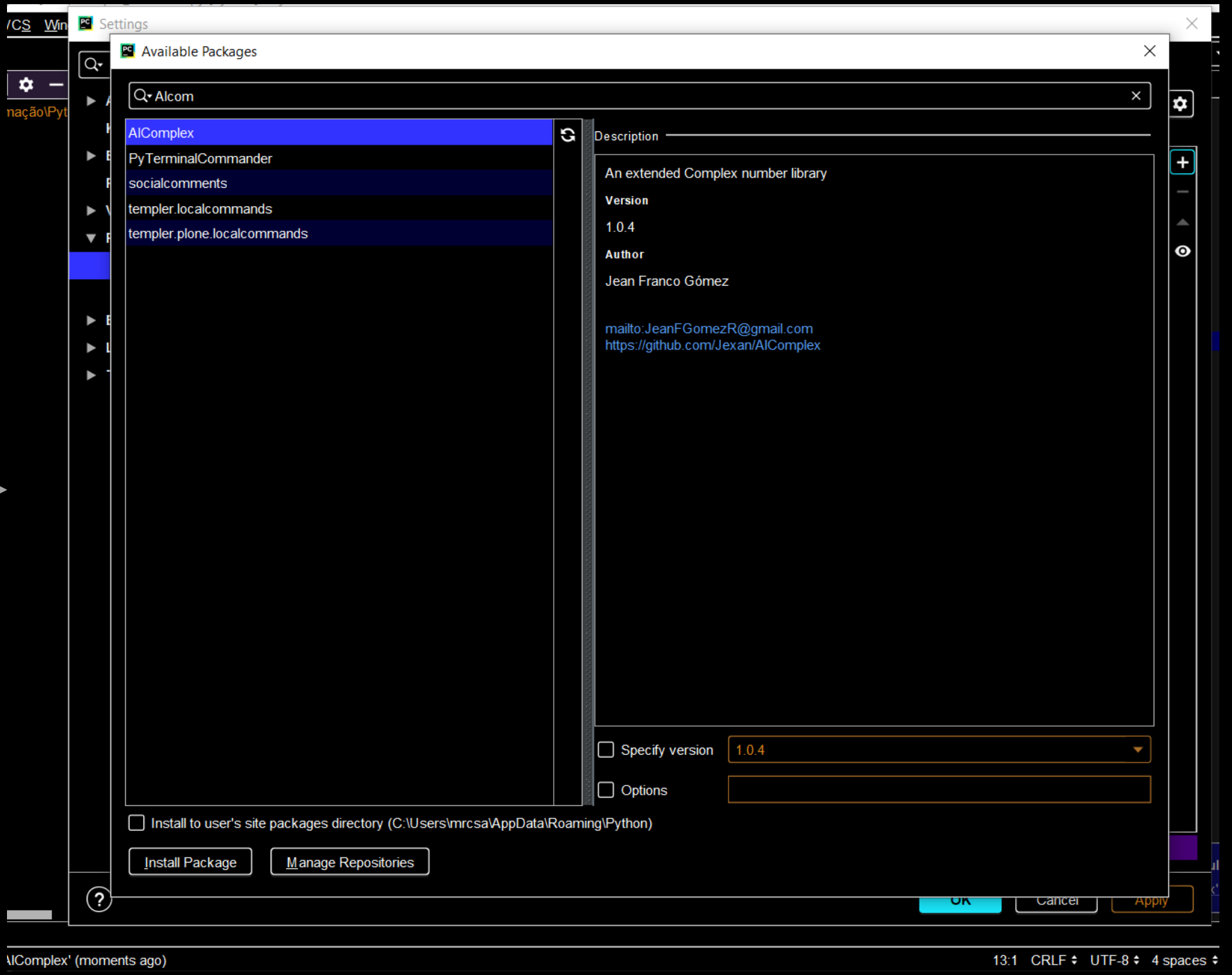
```
1  a = 5
2  print('a = ', a)
3
4  a = 'Viram como isso? '
5  print('a = ' + a)
6
7  a = []
8
9  for i in range(10):
10     a.append(i)
11 print('a = ', a)
12
```

```
a =  5
a = Viram como isso?
a =  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Process finished with exit code 0
```



Sintaxe

Muitas bibliotecas prontas para uso, lembram?



Exemplo_Minicurso.py

```
1   from AlComplex import *  # BASTA ESCREVER ESTA LINHA PARA IMPORTAR TODAS AS CLASSES, MÉTODOS E ATRIBUTOS DESTA BIB.
2   from math import *
3   # Classe criada para o cálculo deste circuito.
4   class Circuito:...
19  # IMPEDÂNCIAS DO CIRCUITO:
20  z1 = AlComplex.polar(2000, 0)
21  z2 = AlComplex.polar(1000, -pi / 2)
22  z3 = AlComplex.polar(500, 0)
23  z4 = AlComplex.polar(1000, pi / 2)
24  z5 = AlComplex.polar(500, -pi / 2)
25  z6 = AlComplex.polar(1000, 0)
26  z7 = AlComplex.polar(100, pi / 2)
27  Z1 = Circuito().paralelo(z1, z2)  # Método da classe "Circuito" sendo utilizado.
28  Z2 = Circuito().serie(Z1, z3)  # Método da classe "Circuito" sendo utilizado.
29  Z3 = Circuito().paralelo(Z2, z4)  # Método da classe "Circuito" sendo utilizado.
30  Z4 = Circuito().serie(Z3, z5)  # Método da classe "Circuito" sendo utilizado.
31  Z5 = Circuito().paralelo(Z4, z6)  # Método da classe "Circuito" sendo utilizado.
32  Z6 = Circuito().serie(Z5, z7)  # Método da classe "Circuito" sendo utilizado.
33  i = Circuito().corrente(Z6)  # Método da classe "Circuito" sendo utilizado.
34  print('Impedância equivalente (ângulo em radianos) = ', Z6.to_polar())
35  print('Corrente "I" (ângulo em radianos) é igual a : ', i.to_polar())
```

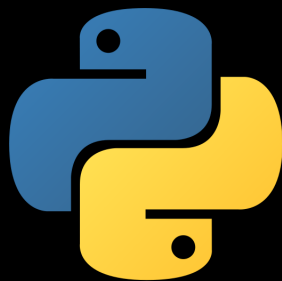
Pycharm apontando que deveria ter linhas de espaço. Python é muito visual!

```
Impedância equivalente (ângulo em radianos) =  (546.5685696828788, 0.2998303053289617)
Corrente "I" (ângulo em radianos) é igual a :  (0.18295966059303478, -0.2998303053289617)

Process finished with exit code 0
```

Python 101

UNIDADE 3 – Conceitos Básicos:

1. Hello World
2. Variáveis
3. Expressões Aritméticas
4. Expressões Lógicas
5. Entrada e Saída de dados
6. Atribuição
7. Bibliotecas



UNIDADE 3

Hello World:

IDE ou prompt → Hello....

UNIDADE 3

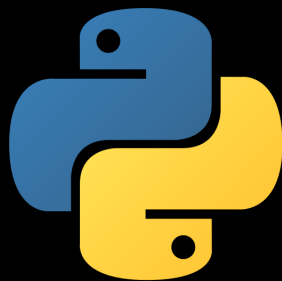
Variáveis:

Função type() → int, float, complex, str, list, dict, etc...

UNIDADE 3

Variáveis:

```
>>> type(1)
<class 'int'>
>>>
>>> type(1.1)
<class 'float'>
>>>
>>> type(1 + 1.1j)
<class 'complex'>
>>>
>>> type('Python')
<class 'str'>
>>>
>>> type([1, 1.1, 1 + 1.1j, 'Python'])
<class 'list'>
>>>
```

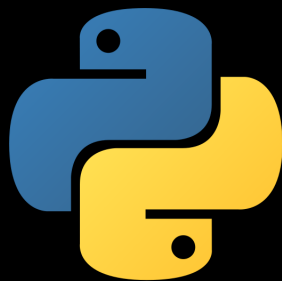


UNIDADE 3

Variáveis:

Não precisa declarar o tipo;

Nome das variáveis → (1abc; çabc; class; etc...).



UNIDADE 3

Variáveis:

Em Python, variáveis mutáveis e imutáveis → Exemplo: Dicionários, tuplas, listas;

UNIDADE 3

Variáveis:

Podem ser do tipo ‘bool’, booleanos.

UNIDADE 3

Expressões aritméticas:

+, -, *, / → normais

// → inteiros

** → potência

UNIDADE 3

Expressões aritméticas:

```
>>> 11 + 7
18
>>> 18 - 11
7
>>> 7 * 11
77
>>> 77 / 11
7.0
>>> 7 % 11
7
>>> 7 ** 11
1977326743
>>> 1977326743 // 11
179756976
>>> 1977326743 / 11
179756976.636363
```

UNIDADE 3

Expressões lógicas:

Operadores	Descrição
and	Retorna um valor verdadeiro se, e somente se, receber duas expressões verdadeiras.
or	Retorna um valor falso se, e somente se, receber duas expressões falsas.
not	Retorna verdadeiro se receber uma expressão falsa, e vice-versa.
is	Retorna verdadeiro se, e somente se, receber duas expressões cujos valores são iguais.
in	Retorna verdadeiro se, e somente se, receber um valor contido numa lista, tupla, num dicionário, etc.
X « Y	Retorna X com os bits descolados à esquerda por Y lugares.
X » Y	Retorna X com os bits deslocados à direita por Y lugares.
X	Retorna o complemento de X. É equivalente a: - X - 1.
X ^ Y	É um bitwise exclusivo (ou XOR a cada bit). O bit da saída é o mesmo que o da entrada (X) se o bit de Y for 0, e é o complemento do bit de entrada (X) se esse bit em Y for 1.
X == Y	Retorna verdadeiro se, e somente se, X for igual a Y.
X != Y	Retorna verdadeiro se, e somente se, X for diferente de Y.
X > Y	Retorna verdadeiro se, e somente se, X for maior que Y.
X < Y	Retorna verdadeiro se, e somente se, X for menor que Y.
=> ou =>	Retorna verdadeiro se, e somente se, X for maior ou igual a Y.
<= ou <=	Retorna verdadeiro se, e somente se, X for menor ou igual a Y.

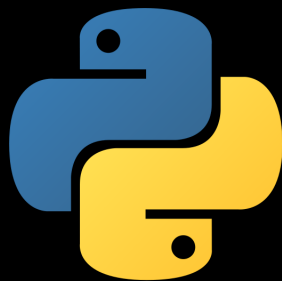
Tabela 1: Operadores lógicos, relacionais e bitwise. Fonte: Autores.

UNIDADE 3

Expressões lógicas:

Em Python é possível a criação e a análise de intervalos.
Ou seja, é possível utilizar um intervalo como uma estrutura lógica.

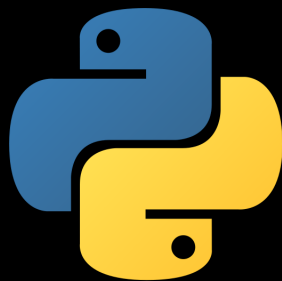
```
>>> a = 5
>>> if 1 <= a <= 6: print('A variável a está contida no intervalo de 1 até 6!')
...
A variável a está contida no intervalo de 1 até 6!
```



UNIDADE 3

Entrada e saída de dados:

- Entrada → input()
  - Obs.: Sempre String.
    - Caso queira converter, basta utilizar as funções intuitivas: int(), float(), complex(), etc... Olhar apostila.



```
>>> a = input('\Digite um número: ')
Digite um número: 7
>>> a
'7'
>>> # O número 7 é representado por um tipo string. Por isso está dentro das aspas.
...
>>> # Para que esse número seja entendido como um tipo numérico , é necessário
convertê -lo, como segue:
...
>>> # Conversão de '7' para inteiro:
...
>>> a = int(a)
>>> a
7
>>>
```

```
>>> # Conversão de '7' para float:
...
>>> a = float(a)
>>>
>>> a
7.0
>>>
>>> # Conversão de '7' para complexo:
...
>>> a = complex(a)
>>> a
(7+0j)
>>>
```

UNIDADE 3

Expressões Entrada e saída de dados:

- Saída → print()
  - Aspas simples ou normal
  - Se String, usar (+); se != usar (,); → concatenação
    - Pode-se concatenar Strings de várias formas. É como uma soma.
    - Se é soma, então também se pode multiplicar Strings.

```
a = ‘Minicurso é top!’
a = 5*a
a = 100*a
```

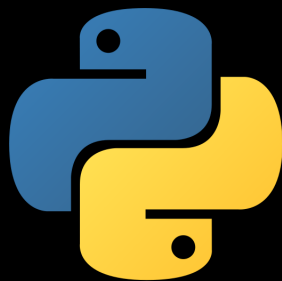
UNIDADE 3

Expressões Entrada e saída de dados:

- Também é possível interpolar Strings.
  - Para isso, utiliza-se o operador “%”.

Símbolo	Tipo de variável
%s	String
%d	Inteiro
%f	Float
%o	Octal
%x	Hexadecimal
%e	Real exponencial
%%	Sinal de percentagem

Quadro I: Símbolos utilizados para interpolação de strings. Fonte: Autores.

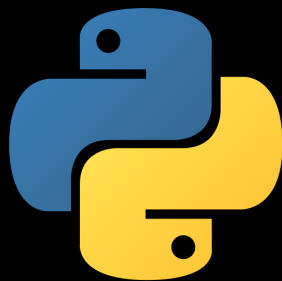


UNIDADE 3

Expressões Entrada e saída de dados:

- Também é possível interpolar Strings.
  - Para isso, utiliza-se o operador “%”.

```
>>> a = 7
>>> b = 5
>>> print("\nHORÁRIO: %02dh%02d" %(a,b))
HORÁRIO: 07h05
>>>
>>> print("\nPORCENTAGEM: %.0f.%.0f%%" %(a,b))
PORCENTAGEM: 7.5%
>>>
>>> print("\nExponencial: %.3e" %a)
Exponencial: 7.000e+00
>>>
>>> print("\nHexadecimal: %x, Decimal: %d, Octal: %o " %(10,10,10))
Hexadecimal: a, Decimal: 10, Octal: 12
>>>
```



UNIDADE 3

Atribuição:

Em Python, assim como em uma ampla gama de linguagens de programação, as atribuições são feitas através do operador “=”, sendo que o valor à esquerda do operador recebe o valor à direita do mesmo. Dessa forma, com as variáveis são atribuídas, as Listas, vetores e outras estruturas são iniciadas.

UNIDADE 3

```
>>> # Atribuição de variáveis:
...
>>> a = 7
>>> b = 7 + 7.7j
>>> c = 'Texto....'
>>>
>>> # Atribuição de listas , Tuplas e Dicionários
...
>>> lista = []
>>> tupla = ()
>>> dicionario = {}
>>>
```

UNIDADE 3

```
>>> a = 7
>>> print('a = ', a, '\nTipo: ', type(a))
a = 7
Tipo: <class 'int'>
>>>
>>> a = a + 7j
>>> print('a = ', a, '\nTipo: ', type(a))
a = (7+7j)
Tipo: <class 'complex'>
>>>
>>> a = 'Muito simples , não?!'
>>> print('a = ', a, '\nTipo: ', type(a))
a = Muito simples , não?!
Tipo: <class 'str'>
>>>
```

UNIDADE 3

Bibliotecas:

- Mostrar como instalar no PyCharm.

- Para usar as bibliotecas e módulos → import ou from import.

```
from math import *

print('cosseno de 0: ', cos(0))
```

UNIDADE 3

Bibliotecas:

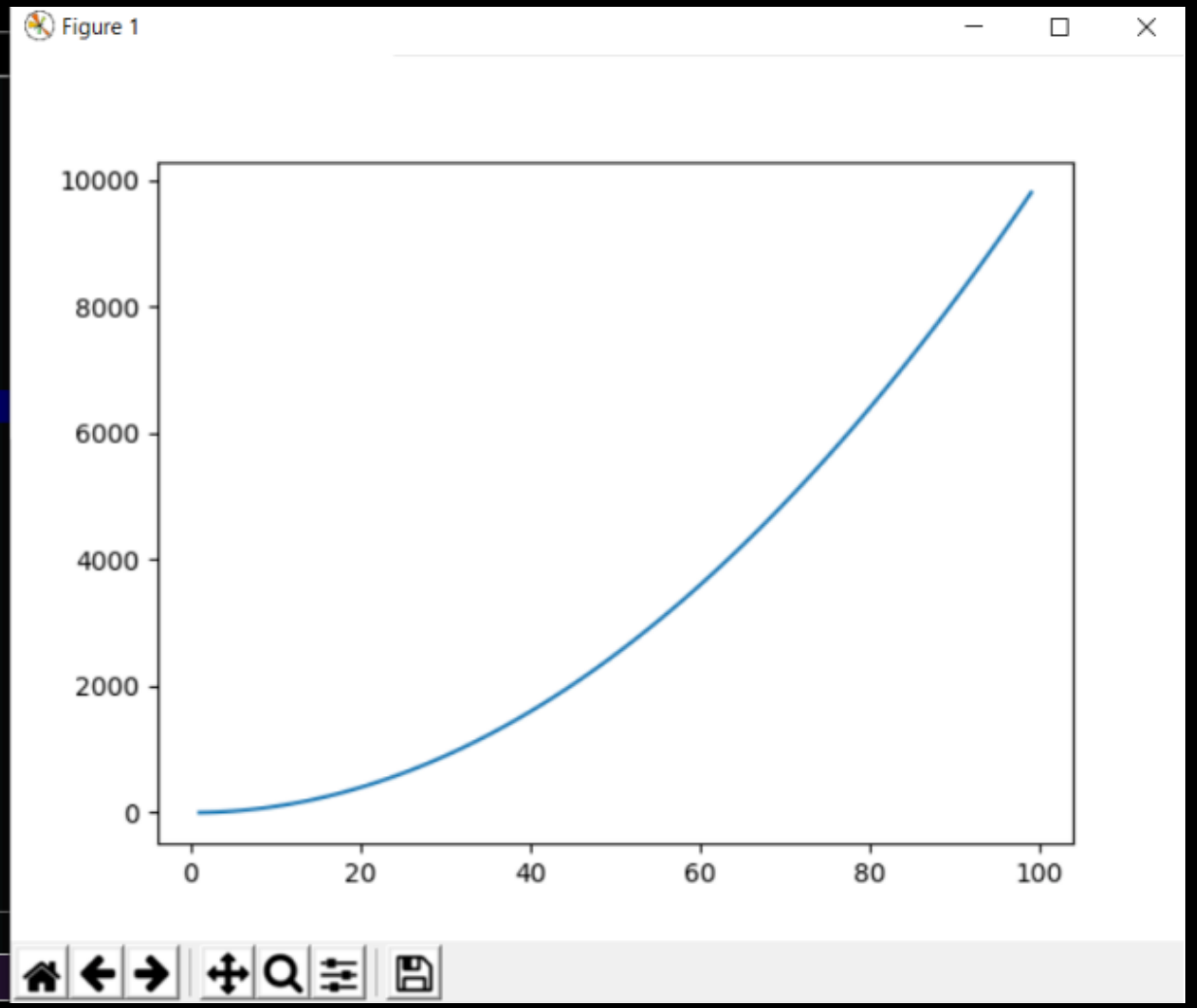
```
from matplotlib import pyplot as plt
x = []
y = []

for i in range(100):
    if i != 0:
        if 100/i != 1:
            x.append(i)
            y.append(x[i-1]**2)
plt.plot(x, y)
plt.show()
```

UNIDADE 3

Bibliotecas:

Bibliotecas.py

```
1  from matplotlib import pyplot as plt
2  x = []
3  y = []
4  for i in range(100):
5      if i != 0:
6          if 100/i != 1:
7              x.append(i)
8              y.append(x[i-1]**2)
9  plt.plot(x, y)
10 plt.show()
11
```



UNIDADE 3

Bibliotecas:

- **Matemática:** *math, cmath, decimal e random;*
- **Sistema:** *os, glob, shutils e subprocess;*
- **Threads:** *threading;*
- **Persistência:** *pickle e cPickle;*
- **XML:** *xml.dom, xml.sax e elementTree (a partir da versão 2.5);*
- **Configuração:** *ConfigParser e optparse;*
- **Tempo:** *time e datetime;*
- **Outros:** *sys, logging, traceback, types e timeit.*

Exercício de Revisão: Unidade 3

Faça um programa que calcula a área de um triângulo qualquer:

- Utilize a fórmula de Heron;

$$p = \frac{a+b+c}{2}$$

$$A = \sqrt{p(p-a)*(p-b)*(p-c)}$$

Resolução:

Ex.Rev_U3.py

```
'''
Faça um programa que calcula a área de um triângulo qualquer:
Utilize a fórmula de Heron;

p = (a+b+c)/2

A = sqrt(p*(p-a)*(p-b)*(p-c))
'''

# Importando uma biblioteca:
import math

print('Digite os valores dos lados do triangulo!')

# Recebendo variáveis do usuário:
a = input('a = ')

# Convertendo a variável string para inteiro:
a = int(a)

# Recebendo variáveis em inteiro:
b = int(input('b = '))
c = int(input('c = '))

# Utilizando as fórmulas de Heron:
p = (a+b+c)/2
raiz = p*(p-a)*(p-b)*(p-c)

print('O valor do semiperímetro é p =', p, 'e a área do triângulo é A =', math.sqrt(raiz))
```

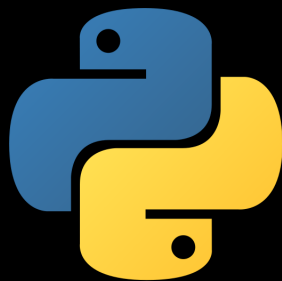
Exercício de Revisão: Unidade 3

Faça um programa que calcule o valor em reais de um banho:

- Considere o valor dado do kWh, a potência do chuveiro e o tempo do banho em minutos.

Exercício de Revisão: Unidade 3

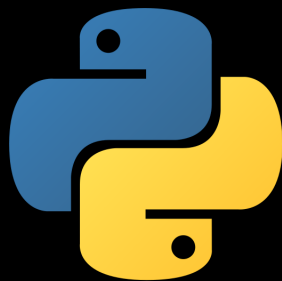
```
kWh_RS = float(input('Preço do kWh: '))
pot_Chuveiro = float(input('Potência do Chuveiro [kW]: '))
tempo_Banho = float(input('Tempo do banho em minutos: '))

tempo_Banho_Horas = tempo_Banho / 60

gasto = tempo_Banho_Horas * kWh_RS * pot_Chuveiro

print('GASTO DO CHUVEIRO: R$', gasto)
```



Exercício de Revisão: Unidade 3

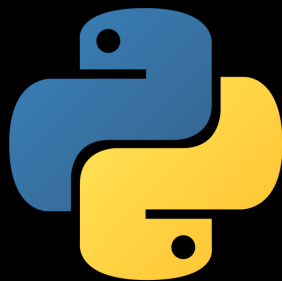
Dados os coeficientes de uma equação do segundo grau (a, b, c). Determine as raízes desta equação:

- Faça um algoritmo genérico para todas as possibilidades (reais e complexas).



Exercício de Revisão: Unidade 3

```
#  from math import sqrt

a = float(input('a: '))
b = float(input('b: '))
c = float(input('c: '))

x1 = (-b + pow((pow(b, 2) - 4 * a * c), 0.5)) / (2 * a)
x2 = (-b - pow((pow(b, 2) - 4 * a * c), 0.5)) / (2 * a)

print('x1 = ', x1, '\tx2 = ', x2)

# Obs.: Se usar o "sqrt()" da bib. math, não é possível
# obter raízes complexas de forma direta.
# Conclusão: Muitas vezes os builtins do python são
# melhores que bibliotecas externas.
```



Python 101

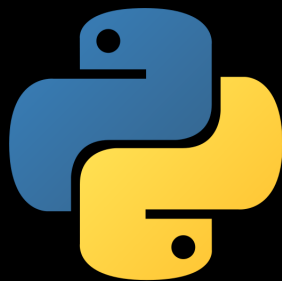
UNIDADE 4 – Desvios Condicionais:

1. If, Elif e Else

UNIDADE 4

Em Python, os operadores condicionais são o **“if”** e o **“else”**. Desses, existe a ramificação **“elif”**, que é a junção do “if” com o “else” para endentar os condicionais em sequência lógica.

```
a = 1
b = 2
if b > a:
            c = 'is cool! '
elif a == b:
            c = "isn't cool! "
else:
            c = "isn't cool! "
a = 'Apostila '
b = 'of Python '
print(a + b + c)
```

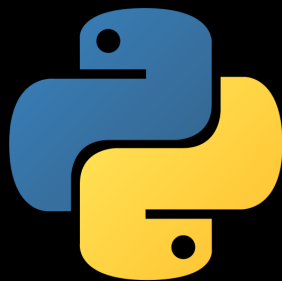


UNIDADE 4

É importante destacar que, eventualmente é necessário utilizar algum condicional que não execute comando algum. Para isso, basta utilizar a instrução **“pass”**, que não executa nenhum comando.

Além disso, pontua-se que os condicionais podem não conter o último comando **“else”**. Nesse caso, se as condições dos “if’s” e/ou “elif’s” forem atendidas, o programa continua seu “raciocínio”; caso contrário, nenhum comando será executado e o programa continua seu “raciocínio”.

```
a = 1
b = 2
if b > a:
        if a < b:
                if b > a:
                        print('\nb > a')
```

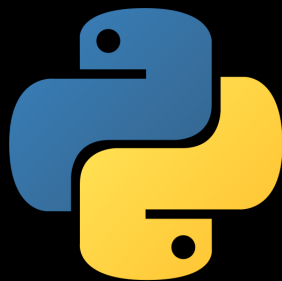


UNIDADE 4

Em Python, caso haja a necessidade de se verificar a condição de um intervalo ou de valores diferentes em relação a um terceiro, é possível simplificar sua sintaxe como segue no exemplo abaixo.

```
import random

a = random.random()
if -10 < a and a < 10:
        print('\n (-10, 10) é o intervalo de a.')
else:
        print('\n (-10, 10) não é o intervalo de a.')
```

```
import random

a = random.random()
if -10 < a < 10:
        print('\n (-10, 10) é o intervalo de a.')
else:
        print('\n (-10, 10) não é o intervalo de
                a.')
```



Exercício de Revisão: Unidade 4

Faça um Programa que verifique se uma letra digitada é vogal ou consoante.

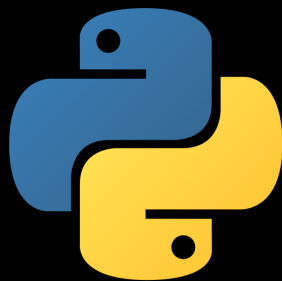
Resolução:

Ex.Rev_U4.py

```
'''
Faça um Programa que verifique se uma letra digitada é vogal ou consoante.
'''

char = input('Digite um caractere: ')

if char == 'a' or char == 'e' or char == 'i' or char == 'o' or char == 'u':
    print('Vogal')

elif char == 'A' or char == 'E' or char == 'I' or char == 'O' or char == 'U':
    print('Vogal')

else:
    print('Consoante')

```

Exercício de Revisão: Unidade 4

Faça um programa que leia um número e escreva na tela se o número é menor, igual ou maior que zero:



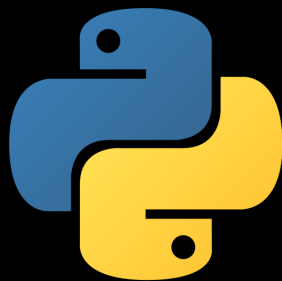
Exercício de Revisão: Unidade 4

```
num = float(input('Número: '))

if num < 0:
    print('\nO número ', num, ', é menor que 0.')
elif num == 0:
    print('\nO número ', num, ', é igual a 0.')
else:
    print('\nO número ', num, ', é maior que zero.')
```

Exercício de Revisão: Unidade 4

Em C:

```c
#include<stdio.h>
#include<conio.h>
int main()
{
float numero;
printf("\nQual o numero?\n");
scanf("%f", &numero);
if(numero>0)
printf("\nO numero entrado e maior que zero");
else
if(numero<0)
printf("\nO numero entrado e menor que zero");
else
printf("\nO numero e igual a zero");
getch();
}
```



Python 101

UNIDADE 5 – Laços de Repetição:

1. For
2. While

UNIDADE 5

For:

```
for i in range(5):
    print(i)
```

Qual será a saída?

Neste caso, o laço “for” imprime na tela o respectivo valor de “i” para cada iteração. O primeiro valor de “i” é 0 e segue sendo incrementado de um em um até o número 4.

UNIDADE 5

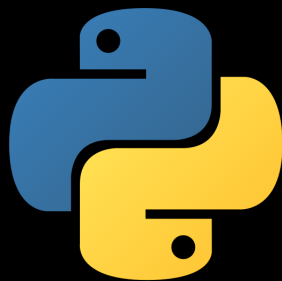
For:

Esta é uma das possíveis formas de se “limitar” o “for”, porém existem inúmeras outras. O que deve ser primeiramente entendido é o raciocínio de sua utilização. Ou seja, para se utilizar o “for” é sempre necessário indicar uma variável iteradora que irá assumir um valor diferente para cada iteração do laço, e é sempre necessário indicar os limites de iteração.

Em Python, as iterações do “for” podem ser realizadas de diversas formas. A variável iteradora pode assumir tanto o valor de um número, quanto de uma String.

```
x = 'Apostila'
for i in x:
        print(i)
```

```
for i in [0, 1, 2, 3]:
        print(i)
```

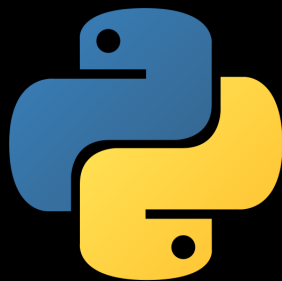


UNIDADE 5

For:

Além disso, existem funções que podem ser utilizadas para limitar o “for”, como a função “range(m, n, p)” (utilizada anteriormente), que cria uma lista de inteiros começando em “m” e terminando em “n-1” sendo incrementada de “p” em “p”; e a função “len()” que retorna o tamanho de determinado vetor.

```
m = -1
n = +1
p = 1
for i in range(m, n, p):
        print(i)
```



UNIDADE 5

For:

Geralmente, a função “len()” é utilizada quando se quer realizar um ”loop”por uma lista, de tal forma que o valor da variável iteradora não assuma cada valor da lista e sim cada posição da lista.

Ou seja, não importa o conteúdo de cada posição da lista, a variável iteradora irá assumir o valor de cada posição. O exemplo abaixo exemplifica isso.

```
x = ['Apostila', ' é', ' nota', 100]

for i in range(len(x)):
        print(i)
for i in x:
        print(i)
```

UNIDADE 5

For:

Além disso, o laço “for” pode ser melhor utilizado através das instruções **“break”** e **“continue”**. Essas instruções também seguem o mesmo sentido de outras linguagens de programação.

A instrução “break” interrompe o laço (terminando-o por completo) e a instrução “continue” pula para a próxima iteração imediatamente (não termina o laço, apenas passa à próxima iteração). O exemplo abaixo ilustra a utilização dessas instruções.

Qual a saída?

```
for i in range(5):
    if i == 0:
        print('\ni = 0, Então: ', i)
    elif i == 1:
        print('\ni = 1, Então: continue')
        continue
    elif 1 < i < 3:
        print('\nA variável i, é: ', i)
    elif i == 3:
        print('\ni = 3, Então: break')
        break
    else:
        print('\ni > 3, Então: ', i)
```

UNIDADE 5

Wilhe:

De forma simples, o laço é executado enquanto sua condição for verdadeira. Sua sintaxe em Python é demonstrada na abaixo, como segue.

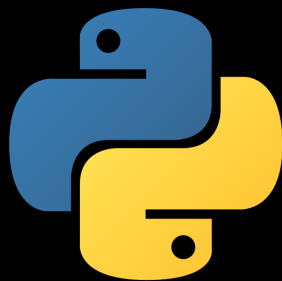
```
a = 0
b = 2
while a <= b:
    print('\n', a, ' <= ', b, ' ')
    a += 1
```

UNIDADE 5

Wilhe:

Uma utilização muito comum do laço “while” é para se criar laços infinitos para a modelagem e criação de games.

```
i = 0
while True:
    print(i)
    i += 1
```



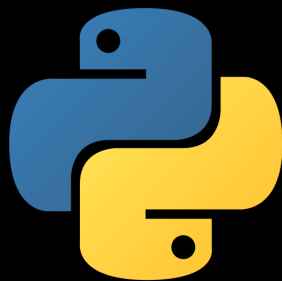
UNIDADE 5

Wilhe:

Uma utilização muito comum do laço “while” é para se criar laços infinitos para a modelagem e criação de games.

```
i = 0
while True:
    print(i)
    i += 1
```

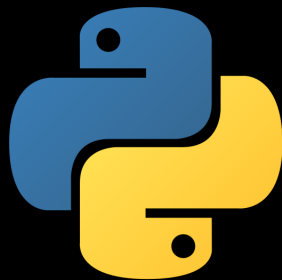
```
# 3046131
# 3046132
# 3046133
# 3046134
# 3046135
# 3046136
# 3046137
# 3046138
# 3046139
# .....
```



UNIDADE 5

While:

Segundo Borges (2010) (REFERÊNCIA NA APOSTILA), o laço “while” é adequado quando não se sabe quantas iterações devem ocorrer até se atingir um objetivo específico e quando não há uma sequência a ser seguida.



Exercício de Revisão: Unidade 5

Faça um programa que soma X números gerados aleatoriamente no intervalo de 1 a 10, onde X é informado pelo usuário.

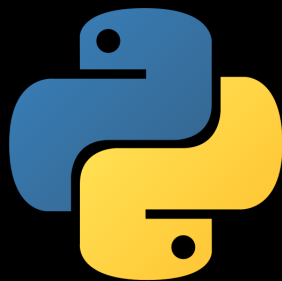
Resolução:

Ex.Rev_U5.py

```
'''
Faça um programa que soma X números gerados aleatoriamente no
intervalo de 1 a 10, onde X é informado pelo usuário.
'''

from random import randint

x = int(input('Digite um numero: '))
soma = 0
contador= 0

while contador < x:
    numero_sorteado = randint(1,10)
    print(numero_sorteado)
    soma = soma + numero_sorteado
    contador = contador + 1

print('A soma eh', soma)

```

Exercício de Revisão: Unidade 5

Sendo H = 1 + 1/2 + 1/3 + 1/4 + ... + 1 /N, elabore um programa para gerar o número H. O número N é fornecido pelo usuário:

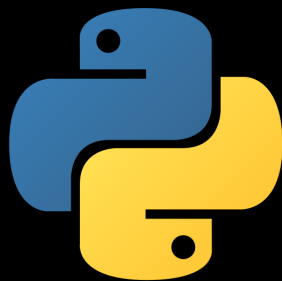


Exercício de Revisão: Unidade 5

```
n = int(input('Valor de n: '))
h = 1
i = 1
while i < n:
    print('\ni = ', i, '\tH = ', h)
    i += 1
    h += 1/i
print('\nN = ', n, '\nH = ', h)
```

Exercício de Revisão: Unidade 5

Faça um programa que calcule o fatorial de um número:



Exercício de Revisão: Unidade 5

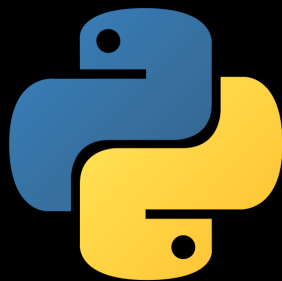
```
n = int(input('Número: '))
fat = 1
if n > 0:
    i = n
    while i > 0:
        fat *= i
        i -= 1
print('\nfat = ', fat)
```

Exercício de Revisão: Unidade 5

Faça um programa que calcule o desvio padrão corrigido e a média de 10 números.

- Utilize:

$$\sqrt{\frac{1}{N-1}\left[\sum_{i=1}^{N} X_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N} X_i\right)^2\right]}$$



Exercício de Revisão: Unidade 5

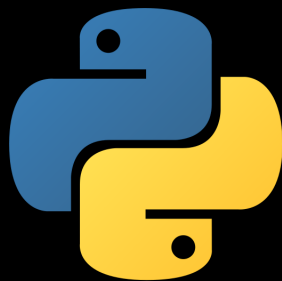
```
nums = []
for i in range(10):
    n = float(input('Número ' + str(i+1) + ': '))
    nums.append(n)
##########################################
media = 0
for i in nums:
    media += i
media = media/10
##########################################
X1 = 0
X2 = 0
for i in nums:
    X1 = X1 + i**2
    X2 += i
X2 = X2**2
dPadrao = pow((1 / (10-1) * (X1 - X2 / 10)), 0.5)
##########################################
print('Média: ', media, '\tDesvio Padrão: ', dPadrao)
```

Exercício de Revisão: Unidade 5

Faça um programa que calcule a série de Taylor da função exp(x).

- Utilize:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots + \frac{x^n}{n!} = \sum_{i=0}^{n} \frac{x^i}{i!}$$



Exercício de Revisão: Unidade 5

```python
n = 60  # Número de repetições do somatório
exp = 0

x = float(input('Digite um valor para X: '))

for i in range(n):
    m = 1
    pot = 1
    while m <= i:
        pot *= x
        m += 1

    fat = 1
    cont = i
    while cont > 0:
        fat *= cont
        cont -= 1

    exp += pot/fat
    print('exp: ', exp)

print('EXP(', x, '), é : ', exp)
```

Exercício de Revisão: Unidade 5

$$\sin x = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{(2k+1)!} = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \cdots$$

EXERCÍCIO EXEMPLO.

```python
from math import pi
n = 5  # Número de repetições do somatório
sinal = -1
sen = 0
ang_g = float(input('Digite o ângulo em graus: '))
ang_rad = pi * ang_g / 180
for i in range(n):
    sinal *= (-1)
    potencia = 1
    m = 1
    while m <= 2*i+1:
        potencia *= ang_rad
        m += 1
    cont = 2 * i + 1
    fat = 1
    while cont > 0:
        fat *= cont
        cont -= 1
    sen += sinal * potencia / fat
    print(sen)
print('\nSEN(', ang_g, '°), é : ', sen)
```