Introdução a Python

Paulo Serra Filho

https://www.python.org/about/

“Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.”

Python em Data Science

Fácil de aprender

8 em 10 cursos de ciência da computação dos EUA usam como linguagem inicial para os alunos.

Propósito geral

Diferente de R, que foca apenas em tratar de estatística, Python é uma linguagem completa, capaz de tratar da aquisição dos dados, data cleaning, interação com bancos de dados, etc.

Bibliotecas para Data Science

SciPy

Python Releases

Python Version	Date of Release
Python v0.1.0 (The first Edition)	1990
Python v0.9.5 (Macintosh support)	2nd Jan'1992
Python v1.0.0	26th Jan'1994
Python v1.1.0	26th Jan'1994
Python v1.2.0	Apr'1995
Python v1.3.0	Oct'1995
Python v1.4.0	Oct'1996
Python v1.5.0	3rd Jan'1998
Python v1.6.0 (Latest updated version)	5th Sep'2000
Python v2.0.0 (Added list comprehensions)	16th Oct'2000
Python v2.7.0 (Latest updated version)	3rd Jul'2010
Python v3.0.0	3rd Dec'2008
Python v3.6.x (Latest updated version)	Mar'2017 and continued.

Python 2 x Python 3

“Python 2 é legado, Python 3 é o presente e futuro da linguagem”

https://wiki.python.org/moin/Python2orPython3

Em alguns casos, é preciso se ater ao Python 2.

- Extensa documentação para Python 2.
- Porting de libraries para Python 3 em alguns casos não existe.

Grande subconjunto comum entre Python 2.6+ e Python 3.3+.

Coding-style

PEP8

https://www.python.org/dev/peps/pep-0008/

conda install -c conda-forge pycodestyle

pip install pycodestyle

$pycodestyle <arquivo>.py

Instalação

https://wiki.python.org/moin/BeginnersGuide/Download

Anaconda

https://anaconda.org/anaconda/python

Anaconda - Inicializar ambiente

conda create --name myenv python=3.5

WINDOWS: activate myenv

LINUX, macOS: source activate myenv

Instalando packages

pip install <package_name>

conda install --name myenv <package_name>

conda remove -n myenv scipy

Hello, World!

>>>print (“Hello, World!”)

__name__ == “main”

Operadores boleanos

and

or

not

Funções

```
def function_name (arg1, arg2, ..., argN ):
        #seu código aqui
        return <valor> #return é opcional
```

Tipos

Embora fracamente tipada, os tipos existem em Python.

Além dos tipos básicos

Tuplas

Listas

Dicionários

Strings

Concatenação é trivial

Alguns casos especiais exigem um pouco de atenção (concatenação de strings e números)

Slicing ⇒ string[start:end:step]

Funções nativas facilitam diversas atividades

- split
- replace
- find

Listas

Uni ou multidimensionais (listas de listas)

Podem conter diferentes tipos de dados

Funções nativas

- append
- pop
- insert
- remove

Tuplas

Similar a listas, porém são um conjunto de dados imutável

Podem conter diferentes tipos de dados

Funções que precisam retornar múltiplas variáveis, fazem uso de tuplas

Condicionais

```
if expression1:
        statement(s)
elif expression2:
        statement(s)
elif expression3:
        statement(s)
else:
        statement(s)
```

1 - Crie um script em python que determine se uma dada string é um palíndromo.

2 - Crie um script em python com uma função que, dada uma string de entrada, retorne uma nova string formada pelos primeiros 2 caracteres concatenados com os dois últimos caracteres. Caso o tamanho da string seja menor que 2, retorne uma string vazia.

3 - Dada uma tupla, imprima o terceiro elemento a partir do início e o quinto a partir do final. Considere que a tupla tem tamanho suficiente.

4 - Escreva um programa que determina se uma lista é vazia

5 - Escreva uma função para determinar o máximo entre três números

6 - Escreva um programa que transforme uma lista de caracteres em uma string

Dicionários

Listas associativas ⇒ chave:valor

Chaves podem ser números ou strings

Valores podem ser praticamente de qualquer tipo

Estruturas de repetição

```
while condicao:
    #executa while
else:
    #executa quando condição == FALSE
```

Estruturas de repetição

```
for <elemento> in <iterador>:
    #executa for
else:
    #executa quando terminar de percorrer toda a lista
```

range VS xrange

Retorna uma sequência sobre a qual é possível iterar.

range → Gera a sequência inteira quando chamada.

xrange → Gera a sequência sob demanda.

Em Python 3 xrange foi abolido e range passou a se comportar da mesma forma que xrange de Python2.

Controle de laço

Break

Continue

Pass

1 - Escreva um programa que dadas duas listas confira se a segunda está contida na primeira

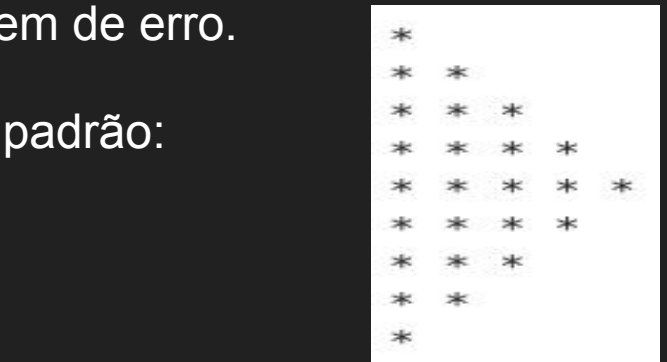
2 - Encontre o segundo maior elemento de uma lista de inteiros não repetidos

3 - Converta uma lista em um dicionário aninhado

4 - Usando “continue”, imprima todos os números entre 0 e 30 exceto os divisíveis por 9.

5 - Dado um dicionário, caso a chave “teste” exista, imprima o valor associado. Caso não exista, exiba uma mensagem de erro.

6 - Crie um programa que imprima o padrão:



Sets

Coleção não ordenada de objetos distintos

Set ⇒ Mutável, com suporte a operações como add() e remove()

Frozenset ⇒ Imutável e hasheável, pode ser usado como chave em um dicionário ou elemento de um outro conjunto.

map

Similar a range, funciona de forma lazy.

Aplica uma função à cada elemento de um iterável

map(function, iterable, …)

Retorna um iterable

Compreensão de listas

Forma abreviada de escrever sequências.

Melhor performance

Lambda

Funções anônimas.

Simples e com propósito específico.

Apenas uma expressão a ser avaliada na função.

Não é possível ter valores default para os parâmetros

1 - Dadas todas as letras minúsculas e todos os algarismos, use compreensão de listas para criar uma lista com todas as possibilidade para duas letras seguidas de dois números.

2 - Escreva um programa que crie um vetor 3x4x6 em que cada elemento é ‘*’.

3 - Transforme a solução do exercício 3.1 em uma função lambda

4 - Dado um dicionário de chaves inteiras positivas, use a função map para encontrar todas as chaves que são maiores do que um número determinado. Imprima chave e valor.

5 - Dados dois dicionários, indique quais pares ‘chave’:’valor’ fazem parte de ambos.

Datetime

Módulo para manipulação de dados de data e horário.

Aritmética é suportada, porém o foco da implementação é na eficiência de manipulação das saídas e manipulação dos dados.

date ⇒ Ano, mês, dia

time ⇒ Hora, minuto, segundo, microssegundo, time-zone info.

datetime ⇒ Ano, mês, dia, hora, minuto, segundo, microssegundo, time-zone info.

timedelta ⇒ Diferença (duração) entre duas instâncias date, time ou datetime.

numpy

Pacote muito útil para diversas situações científicas

Define um objeto array multidimensional

array
shape
reshape
arange
linspace
zeros
ones

1 - Faça um programa que converte uma string para datetime e um datetime para string

2 - Faça um programa que diga quantos meses tiveram uma segunda-feira como primeiro dia entre 2015 e 2016. Nota: A função weekday() pode ajudar.

3 - Escreva um programa que inverta um array.

4 - Crie uma função que converta listas e tuplas para arrays.

5 - Crie uma matriz 5x5 com os valores de cada linha variando de 0 a 4

6 - Escreva um programa que gere um vetor contendo todos os valores múltiplos de 3 ou 5 menores que 100. Em seguida some todos os valores.

Referencias

https://docs.python.org/3/

https://conda.io/docs/user-guide/

https://conda.io/docs/_downloads/conda-cheatsheet.pdf

https://linuxconfig.org/learn-python-fundamentals

www.coursera.org/learn/python-data-analysis

https://docs.scipy.org/doc/numpy-dev/user/quickstart.html

https://www.w3resource.com/python-exercises/

https://machinelearningmastery.com/machine-learning-in-python-step-by-step/

Introdução a Python

Paulo Serra Filho