Data Engineering

Python

Paulo Serra Filho

Thiago Cardoso

“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 Engineering

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

Propósito geral, diferente de R que foca em estatística

Possui muitas bibliotecas específicas para Data Engineering (SciPy)

“But how does Spotify actually use that concept in practice to calculate millions of users’ suggested tracks based on millions of other users’ preferences?

With matrix math, done with Python libraries!”

How Does Spotify Know You So Well?
https://medium.com/s/story/spotifys-discover-weekly-how-machine-learning-finds-your-new-music-19a41ab76efe

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.

“Short version: Python 2.x is legacy, Python 3.x is the present and future of the language”

porém...

Python 2 vs Python 3

Extensa documentação para Python 2

Algumas bibliotecas **não existem** para Python 3

Grande subconjunto comum entre Python 2.6+ e Python 3.3+

Hello, World!

```
$ python
>>> print(“Hello, World!”)

$ python <arquivo>.py
```

Indentação

Indentação define escopo!

~~`begin end, {}`~~

```
def find_prime(num):
    for i in range(...):
        if …
    …
```

Comentários

```
# Comentário
```

```
def find_prime(num):
  # For each number,
  for i in range(...):
    if …
  …
```

Documentação

"""docstrings"""

```
def find_prime(num):
  """Finds primes"""
  # For each number,
  for i in range(...):
    if …
  …
```

Documentação

`help("função")`

```
>>> help("find_prime")
```

Documentação

Python 2

Python 3

Buscar pela página do pacote

- Tutoriais
- Exemplos
- Documentação

Coding-style

PEP8 -- Style Guide for Python Code

```
$ pip install pycodestyle

$ pycodestyle <arquivo>.py
```

Operadores

Lógico	Bitwise
and	&
or	|
not	~

```
…
  if x == 0 and not b:
    continue
…
```

Operadores

`in`, `not in`

```
def find_prime(num):
  for i in range(...):
    if …
  …
```

Condicionais

```
if condição:
   …
elif condição:
   …
elif condição:
   …
else:
   …
```

```
if x == 0:
   …
elif y != 3 and not b:
   …
else:
   …
```

Funcão main

```
if __name__ == “__main__”:
  …
```

```
if __name__ == “__main__”:
  print(“Hello, World!”)
```

Estruturas de repetição

```
while condição:
   …
else:
   # executa quando sair do while
   …
```

Estruturas de repetição

```
for elemento in iterador:
    …
else:
    # executa quando sair do for
    …
```

Controle de laço

`break`

`continue`

`pass`

```
for x in range(100):
  if x % 5 == 0:
    continue
  …
```

`range` vs `xrange`

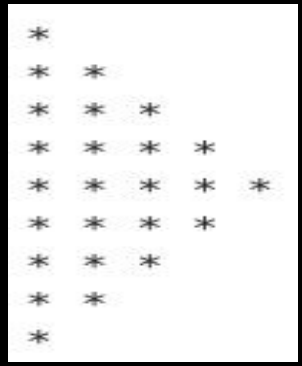
`range` → Gera uma sequência inteira quando chamada.

`xrange` → Gera uma sequência sob demanda.

Em Python 3 `xrange` foi abolido e `range` passou a se comportar da mesma forma que `xrange` de Python 2

Exercícios 1

1. Usando `continue`, imprima todos os números entre 0 e 30 exceto os divisíveis por 9.
2. Crie um programa que imprima o padrão ao lado.
3. Escreva um programa que encontre os números divisíveis por 7 e múltiplos de 5 entre 1500 e 2700.
4. Escreva um programa que nunca acabe e realize as operações:
    - Imprimir uma mensagem na tela
    - Contar até 1000000



Tipos

Fracamente tipada

Além dos tipos básicos:

- Tuplas
- Listas
- Dicionários

```
# Tupla

point = (1.0, 0.0)

# Lista

lang = ['Python', 'C++']

# Dicionários

tr = {}

tr['one'] = 'um'

tr['two'] = 'dois'
```

Strings

`""` ou `''`

Concatenação é trivial

```
str = "This is a string"

str2 = 'this too'

text = str + ' and ' + str2

error = text + 3
```

Strings

Slicing

var[*inicio*:*fim*:*passo*]

```
>>> s = 'This is a string'
>>> s[8:]
'a string'
>>> s[:4]
'This'
>>> s[::4]
'T ar'
```

String

split, replace, find, ...

```
s = 'azul, amarelo, verde'

cores = s
  .replace(' ','')
  .split(',')
```

Listas

Uni ou multidimensionais

Podem conter diferentes tipos de dados

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

any = [1, 'Test', False]
```

Listas

`append`, `pop`, `insert`, `remove`

```
stack.append(10)

stack.pop()

list.insert(2, 'Novo')
```

Tuplas

Conjunto de dados **imutável**

Funções que precisam retornar **múltiplos valores**

```
tp = ('Bacon', 7, True,
11, 'Your mother was a
hamster!')

print(tp[1])
```

Dicionários

Chave ⇒ Valor

Há restrição para tipo da chave

```
c =
{'Debian':'apt-get',
 'Arch':'pacman'}
print(c['Arch'])
```

Funções

```
def nome (arg1, ..., argN):
    …
    # opcional
    return valor
```

```
def find_prime(num):
    for i in range(...):
        if …
    …
```

Funções

Objetos imutáveis são **passados como valor**

- Ex.: strings, tuplas
- Alterações NÃO SÃO refletidos fora dela

Objetos mutáveis são **passados por referência**

- Ex.: listas, dicionários
- Alterações SÃO refletidos fora dela

Exercícios 2

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.
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 um programa que transforme uma lista de caracteres em uma string
6. Encontre o segundo maior elemento de uma lista de inteiros não repetidos
7. Dado um dicionário, caso a chave “teste” exista, imprima o valor associado. Caso não exista, exiba uma mensagem de erro.

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

Exercícios 3

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 2.3 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.

Classes

```
class name:
  member_name = value

  def __init__(self, …):
    …

  def method_name(self, …):
    …

  …
```

```
class MinhaClasse:
  hello = "Hello, World!"

  def __init__(self, nome):
    self.nome = nome

  def hello(self):
    print("Hello, %s"
%self.nome)

  def whoAmI(self):
    return(self.nome)
  …
```

Arquivos

```
open(filename, mode)
file.close()
```

Character	Meaning
`'r'`	open for reading (default)
`'w'`	open for writing, truncating the file first
`'x'`	open for exclusive creation, failing if the file already exists
`'a'`	open for writing, appending to the end of the file if it exists
`'b'`	binary mode
`'t'`	text mode (default)
`'+'`	open a disk file for updating (reading and writing)
`'U'`	universal newlines mode (deprecated)

```
f = open(“arquivo.txt”, “w”)

f.close()
```

Arquivos

file`.read(`*size*`)` ⇒ Lê até size bytes de um arquivo. Caso o parâmetro seja omitido, lê o arquivo inteiro.

file`.readline()` ⇒ Lê uma linha do arquivo.

file`.write(`*string*`)` ⇒ Escreve o conteúdo de string no arquivo.

Datetime

`date` ⇒ Ano, mês, dia

`time` ⇒ Hora, minuto, segundo, micro-segundo, time-zone info.

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

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

Exercícios 4

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 leia uma sequência de números de um arquivo e imprima em um novo arquivo os números em ordem e na sequência invertida.

pip

Gerenciador de pacotes para Python

Instalado em conjunto com Python quando baixado do python.org ou em alguns ambientes virtuais

- Python 2 > 2.7.9
- Python 3 > 3.4

Usando pacotes

```
pip install pacote

import pacote

import pacote as nome
```

```
$ pip install numpy

$ python

>>> import numpy as np

>>> np.array([2,3,4])
```

NumPy

Pacote para computações científicas

Array multidimensional e funções para operações **rápidas** nestes arrays

- Operações matemáticas e lógicas
- Manipulação de forma
- Seleção e ordenação
- Transformadas de Fourier
- …

NumPy

array

arange

zeros

ones

linspace

random

```
>>> np.array([2,3,4])

>>> np.arange(4)

array([0,1,2,3])

>>> np.linspace(1,10,20)
```

NumPy

shape

reshape

```
>>> np.arange(10).shape
(10,)
>>> np.arange(10).reshape(2,5)
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

NumPy

max

min

var

std

mean

```
a = np.linspace(1,10,20)

m = np.mean(a)

…
```

NumPy

Iteração em arrays feita com relação ao primeiro eixo

```
for axis in array:
    …
```

```
a = np
    .linspace(1,10,20)
    .reshape(5,4)
for row in a:
    …
```

Exercícios 5

1. Crie uma função que converta listas e tuplas para arrays.
2. Crie uma matriz 5x5 com os valores de cada linha variando de 0 a 4 e imprima o resultado em um arquivo.
3. 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.
4. Escreva um programa que retorne todos os elementos únicos de um array.
5. Escreva um programa que encontre os valores comuns entre dois arrays

Pandas

```
pip install pandas
```

Python Data Analysis Library

Mais alto nível que NumPy

- Series
- Data frames

Pandas

`pd.Series`

Array **indexado**

Dados são alinhados automaticamente com base nos índices (labels)

```
>>> pd.Series([1,3,np.NaN, 9])
0  1
1  3
2  NaN
4  9
```

pandas.Series
https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html#pandas.Series

Pandas

`pd.DataFrame`

Estrutura **bidimensional**

Similar a um dicionário de Series

```
d = {'Name': ['Paulo', … ],
'Score': [10, 5.75, … ],
'Qual': [True, False, … ]}
df = pd.DataFrame(d)
…
```

Pandas

dataframe[*de*:*para*]

dataframe[*coluna*]

dataframe.T

```
d = {'Name': ['Paulo', … ],
'Score': [10, 5.75, … ],
'Qual': [True, False, … ]}
df = pd.DataFrame(d)
df[1:2]
df['Score']
```

Pandas

dataframe.head

dataframe.tail

dataframe.mean

dataframe.min

dataframe.max

pandas.DataFrame
https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html#pandas.DataFrame

```
>>> d = {'Name': ['Paulo',
… ],
'Score': [10, 5.75, … ],
'Qual': [True, False, … ]}
>>> df = pd.DataFrame(d)
>>> df.max()
Name    Thiago
Score   10
Qual    True
```

Pandas

pd.read_csv

dataframe.to_csv

```
df = pd.read_csv('in.csv')

…

df.to_csv('result.csv')
```

Exercícios 6

1. Usando Pandas, crie uma Serie e depois converta para o tipo nativo python lista.
2. Crie um programa que, dadas duas Series faça as operações de soma, subtração e compare se os elementos da primeira são maiores que os da segunda.
3. Usando o dataframe dado, calcule:
    a. Média de pontos.
    b. Média de tentativas.
    c. Pessoas que tentaram mais de uma vez e não passaram.

Exercícios 6 - cont

4. Adicione uma nova linha ao dataframe contendo os dados:

```
name : ‘Suresh’, score: 15.5, attempts: 1, qualify: ‘yes’
```

5. Salve o novo dataframe em um arquivo CSV, ignorando coluna de labels.

```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
    'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
    'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
    'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no',
'yes']}
```

Exercícios 6 - cont

6. Usando o arquivo final.csv (disponível no GitHub):
    a. Leia os dados em um dataFrame
    b. Adicione uma nova coluna “ratio”, sendo a razão entre pontuação e tentativas para cada pessoa.
    c. Imprima as informações das pessoas com o maior e menor ratio.
    d. Ordene decrescentemente pelo ratio.
    e. Salve o novo arquivo CSV ignorando a coluna de labels.

Material de suporte a estes slides

Referências

About Python

Should I use Python 2 or Python 3 for my development activity?

PEP 8 -- Style Guide for Python Code

Learn Python Fundamentals

Python Exercises, Practice, Solution

Referências

Introduction to Data Science in Python

NumPy Quickstart tutorial

10 Minutes to pandas

Your First Machine Learning Project in Python Step-By-Step

Data Engineering

Python

Paulo Serra Filho

Thiago Cardoso