Python

Programando de forma idiomática

Eliakin Costa

Instituto Federal da Bahia
campus Salvador
eliakin.costa@kde.con

16 de junho de 2018

Agenda

Quem sou eu?

- Meu nome é Eliakin Costa,
- Sou aluno de ADS (IFBA),
- Apaixonado pelo software livre \o/
- Colaborador do KDE
- **Contatos**:
  - **E-mail**: eliakin.costa@kde.com
  - **GitHub**: github.com/eliakincosta

Introdução

O começo de tudo

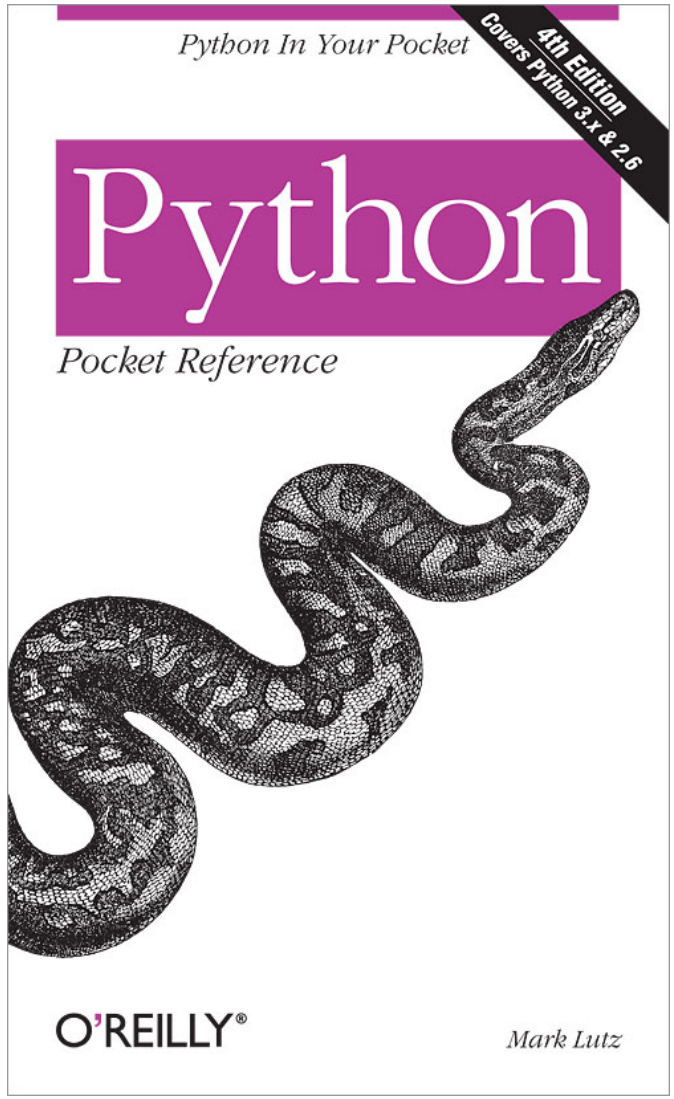
- CWI - Centro Tecnológico
- Guido Van Rossum
- Linguagem ABC

Python!? Que nome é esse?

Python!? Que nome é esse?

- Python Flying in Circus
- Episódio do spam xD
- Guido não gostava da relação do python com uma cobra

Python!? Que nome é esse?



Guido Van Rossum - O ditador benevolente

- Está vivo!!!!
- Google de 2005-2013
- Dropbox de 2013-hoje

Por que eu devo aprender Python?

- Linguagem de Propósito geral
- Simples e intuitivo
- Comunidade espetacular
- Você consegue fazer quase tudo apenas com os recursos da própria linguagem
- Multiparadigma

Quem está usando python?

Quem está usando python?

Tipagem Dinâmica, mas nem tanto

Características da Tipagem Dinâmica

- Verificações de tipo feitas em runtime
- Verificações feita sobre o valor(ex: class int)
- Bob Harper, um dos criadores da linguagem Standard ML, "uma linguagem dinamicamente tipada é uma linguagem estaticamente tipada com apenas um tipo estático".

Como é a tipagem estática?

JAVA

```java
public class HelloWorld
{
  public static void main(String[] args)
  {
    int a = 2;
    String b = "teste";
    a = b;//ERRADO, FALHA NA COMPILACAO
  }
}
```


Como é a tipagem dinâmica?

Python

```
a = 2

a = "texto"

#Variaveis funcionam como referencias polimorficas.
#Verificacao de tipo em runtime. Feita sobre o dado.
```

Tipagem dinâmica, mas forte

- Observe o seguinte código em php:

```
echo 1 + '13 bananas';
```

- Vamos testar o mesmo código em python.

Tipagem forte

- Código mais robusto
- Tipos inválidos ocasionarão exceções
- Explícito é melhor do que implícito

Tipos, Instruções e Estilo

Tipos Básicos do python

- Númericos:
  - inteiro (int)
  - ponto flutuante (float)
  - booleano (bool) 1 ou 0
  - complexo (complex)

- Obs:
  - Suportam as operações básicas, mod, etc (+, -, *, /, %, **, +=, -=, *=, /=,%=, **=).
  - Operações entre diferentes tipo irão resultar no tipo mais complexo
  - Cuidado, números inteiros tem precisão infinita
  - Controle de fluxo e bool

Estruturas Básicas do python

- Iteráveis:

Tabela: Iteráveis

	Ordenada	Modificável	Unicidade
strings	X		
listas	X	X	
tuplas	X		
sets		X	X

- Vamos tentar um pouco de código

Obs: Iremos focar apenas em listas

Cópia Rasa de listas

```
a = [1,2,3]
>>> b = a
>>> b is a
True
>>> a.append(4)
>>> b
[1, 2, 3, 4]
```

Como resolver isso de uma maneira mais fácil?

Hackeando Listas com Python

- Podemos usar fatiamento de iteráveis (slicing)
- O fatiamento faz uma cópia de valor do iterável
- Essa cópia é feita no primeiro nível
- Para casos específicos pode-se usar deepcopy from copy
- **Python aceita níveis arbitrários em estruturas modificáveis**
- Vamos entender isso no quadro!!!

Notas de sintaxe

Atenção para a sintaxe

```
#Exemplo de controle de fluxo e loop
lista = [1, 2, 3, 4, 5]

#Vamos exibir os numeros pares
for elemento in lista:
    if elemento%2==0: # if e : indica inicio de um bloco
        print(elemento)
```


Notas de sintaxe

Atenção para a sintaxe

```
#Eh possivel iterar com while tambem
# Para criar um loop, baseado em uma condicao

lista_2 = [2, 4]
index = 0

while index < len(lista_2):
    if lista_2[index]%2!=0:
        break
    index+=1
else:# Executado apenas quando o loop terminar normalmente
    print('Esta lista tem apenas numeros pares')
```


Hackeando Listas com Python

- O slicing muda a forma como percorremos uma lista
- Mas e se precisarmos fazer verificações para filtrar a lista?

Hackeando Listas com Python

Obtendo Números pares de uma lista

```
lista = [1, 2, 3, 4, 5, 6, 7, 8]

nova_lista = []

for elemento in lista:
    if elemento%2==0:
        nova_lista.append(elemento)

print(nova_lista)
```


Hackeando Listas com Python

Fazendo o mesmo de uma forma mais pythônica

```
lista = [1, 2, 3, 4, 5, 6, 7, 8]

print([elemento for elemento in lista if elemento%2==0])
```

Dicionários

- São indexados por chaves
- Apenas tipos imutáveis podem ser usados como chaves
- São criados com um simples par de chaves -> {}

Dicionários na prática

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> list(tel.keys())
['irv', 'guido', 'jack']
>>> 'guido' in tel
True
>>> 'jack' not in tel
False
```

Dicionários

Dicionários na prática

```
>>> dict(sape=4139, guido=4127, jack=4098)
{'sape': 4139, 'jack': 4098, 'guido': 4127}
```


Funções e Classes

Funções

- São definidas pela palavra reservada *def*
- Podem possuir parâmetros default
- São objetos de primeira classe
  - Criada em tempo de execução
  - Pode ser atribuída a uma variável
  - Passada como argumento para uma função
  - Devolvida como resultado de uma função

Funções

Exemplo de função

```
#Inverter palavra
def reverse(word='reverse'):
    return word[::-1]
```

Classes

- São definidas com a palavra reservada *class*
- Python suporta herança múltipla
- Python possui sobrecarga de operadores, assim como C++
- Todos os atributos são públicos em Python
- Python não possui uma palavra reservada para interface como JAVA, mas o python tem classes abstratas.

Exemplo de classe

```
class Carro(object):

    def __init__(self, marca, cor, velocidade=0):

        self._marca = marca
        self._cor = cor
        self._velocidade = velocidade

    @property
    def marca(self):
        """ Proprieda somente para leitura """
        return self._marca

    @property
    def cor(self):
        return self._cor

    @cor.setter
    def cor(self, cor):
        """ Essa propriedade suporta escrita """
        self._cor = cor

if __name__=='__main__':
    fusca = Carro('BMW', 'Preta')
    print(fusca.marca) # Posso acessar a property como um atributo
```


Prática

CSV

Exemplo de csv

```
import csv

with open('people.csv') as people_file:
    people_reader = csv.reader(people_file, delimiter="|")
    for person in people_reader:
        print(person)
```

Exemplo de JSON

```
import json

with open('books.json') as json_file:
    books = json.load(json_file)
    for book in books:
        print('Author: ' + book['author'])
        print('Title: ' + book['title'])
        print('Year: ' + str(book['year']))
        print('---------------------------------')
```


Peguntas

Referências

- Python Fluente - Luciano Ramalho
- Learning Python - Mark Lutz
- Python Tutorial https://docs.python.org/3/tutorial/