Python

Programando de forma idiomática

Eliakin Costa

Instituto Federal da Bahia
campus Salvador
eliakincosta@ifba.edu.br

26 de abril de 2017

Agenda

Quem sou eu?

- Meu nome é Eliakin Costa,
- Sou aluno de ADS (IFBA),
- Apaixonado pelo software livre \o/
- Membro e Colaborador do OPAI e KDE
- **Contatos**:
  - **E-mail**: eliakim170 at gmail.com
  - **GitHub**: github.com/eliakincosta

Introdução

O começo de tudo

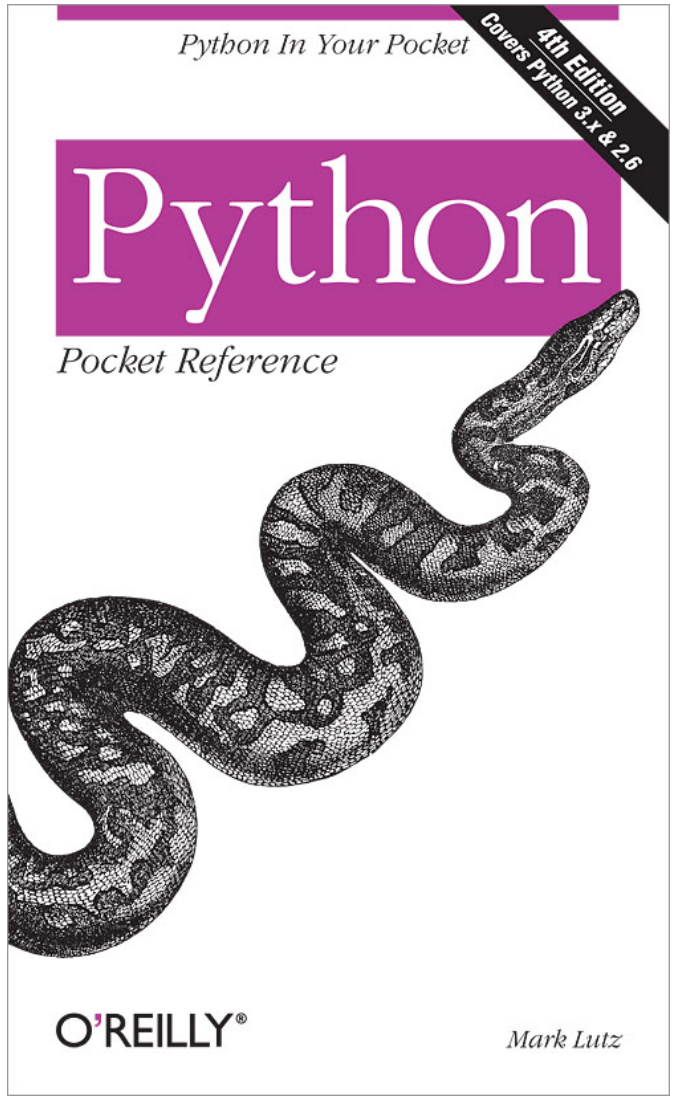
- CWI - Centro Tecnológico
- Guido Van Rossum
- Linguagem ABC
- Guido precisava resolver um problema relacionado com o Amoeba xD

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
- Roda em qualquer sistema
- 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

Tipos Básicos 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

Hackeando Listas com Python

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 estrutura 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])
```

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]
```

Decoradores

Exemplo de decorador

```python
def decorate(func):
    def inner():
        print('running inner')
    return inner


@decorate
def target():
    print('running target')

#ou

def target():
    print('running target')

target = decorate(target)

if __name__=='__main__':
    target()
```


Decoradores

- São *callables* que aceitam função como parâmetro
- Criada em tempo de importação
- Mudar comportamento de funções ou métodos
- Exemplo real de autenticação no Django
- Explícito melhor do que implícito

Classes - Classe Carro

- São abstrações de coisas ou comportamentos do mundo real
- 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.

Classes

```python
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 velocidade(self):
      """ Proprieda somente para leitura """
      return self._velocidade

  @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.velocidade) # Posso acessar a property como um atributo
```

Ainda Tem muito mais

Ainda tem muito mais...

- Descriptors
- Classes abstratas
- Metaclasses
- Toda a biblioteca padrão

Peguntas

Referências

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