Fundamentos de Lógica e Algoritmos

Python para Zumbis
Fernando Masanori

Givanaldo Rocha de Souza

http://docente.ifrn.edu.br/givanaldorocha
givanaldo.rocha@ifrn.edu.br

For, Funções, Random

for === while

>>> Códigos equivalentes <<<

```
for letra in 'aeiou':
    print (letra)
>>>
a
e
i
o
u
```

```
texto = 'aeiou'
k = 0
while k < len(texto):
  letra = texto[k]
  print (letra)
  k = k + 1

>>>
a
e
i
o
u
```

for === while

```
for i in range(5):
  print (i)
>>>
0
1
2
3
4
```

```
lista = list(range(5))
k = 0
while k < len(lista):
  i = lista[k]
  print (i)
  k = k + 1
>>>
0
1
2
3
4
```

>>> Códigos equivalentes <<<

for === while

>>> Códigos equivalentes <<<

```
for x in ['cpbr6', 42, 3.14]:
  print (x)
>>>
cpbr6
42
3.14
```

```
lista = ['cpbr6', 42, 3.14]
k = 0
while k < len(lista):
  x = lista[k]
  print (x)
  k = k + 1
>>>
cpbr6
42
3.14
```

- O for é utilizado quando se há uma contagem fixa ou uma lista de elementos a se percorrer.

- Muitas vezes, é melhor utilizar o for do que o while.

def functions

- Aprendemos algumas funções do Python: len, int, float, print e input
- Agora iremos criar as nossas próprias funções
- Utilizo def para definir a função e return para devolver algum valor
- Existem funções que não retornam nada

def functions

```
def épar(x):
    return x%2 == 0
```

- Esta função retorna se o parâmetro x é par
- Observe que diferentemente do que já vimos até agora, essas linhas não serão executadas imediatamente
- Preciso chamar a função para executá-la

```
>>> épar(13)
False
>>> épar(12)
True
```

Funções

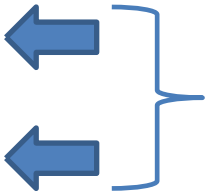
- Defina uma função fatorial

```
def fat(n):
    f = 1
    while n > 0:
        f = f * n
        n = n - 1
    return f
```

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

1
1
2
6
24
```

Variáveis locais e globais

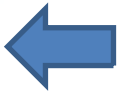
```
a = 5
def muda_e_imprime():
    a = 7
    print ('a dentro da função: %d' %a)
print ('a antes de mudar: %d' %a)
muda_e_imprime()
print ('a depois de mudar: %d' %a)

>>>
a antes de mudar: 5
a dentro da função: 7
a depois de mudar: 5
```



São variáveis diferentes!

Variáveis locais e globais

```
a = 5
def muda_e_imprime():
    global a
    a = 7
    print ('a dentro da função: %d' %a)
print ('a antes de mudar: %d' %a)
muda_e_imprime()
print ('a depois de mudar: %d' %a)
```



É a mesma variável global

```
>>>
a antes de mudar: 5
a dentro da função: 7
a depois de mudar: 7
```

Números aleatórios

```
>>> import random
>>> random.randint(1, 100)
5
>>> random.randint(1, 100)
24
>>> alunos = ['José', 'João', 'Pedro', 'Lucas', 'Tiago']
>>> random.choice(alunos)
'José'
>>> random.choice(alunos)
'Lucas'
>>> random.shuffle(alunos)
>>> alunos
['José', 'Tiago', 'João', 'Pedro', 'Lucas']
>>> random.shuffle(alunos)
>>> alunos
['José', 'João', 'Lucas', 'Pedro', 'Tiago']
```

Números aleatórios

- Defina uma função “embaralha” que retorne as letras de uma string misturadas. Dica: utilize list( ) para converter sua string em lista.

```
def embaralha(s):
    import random
    lista = list(s)
    random.shuffle(lista)
    return ''.join(lista)
```

```
>>> embaralha('palmeiras')
'rlemipasa'
>>> embaralha('palmeiras')
'apmrlseia'
```

Números aleatórios

- Gere uma lista de 15 inteiros aleatórios entre 10 e 100

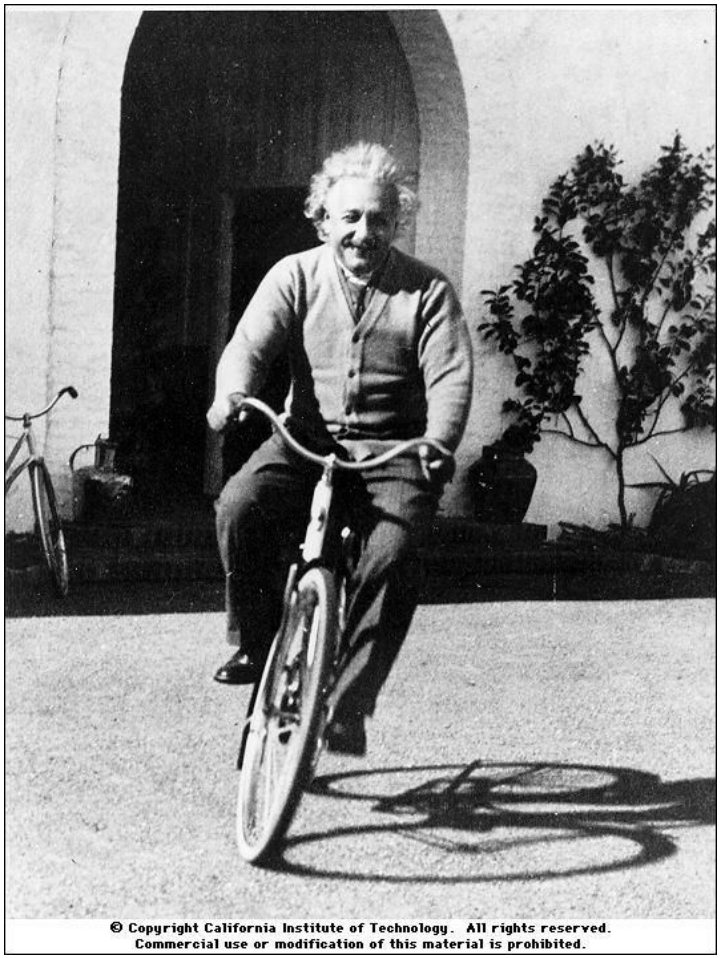
```
import random
lista = []
for k in range(15):
  lista.append(random.randint(10, 100))
print (lista)
```

Números aleatórios

- Gere uma lista de 15 inteiros aleatórios entre 10 e 100 que sejam distintos entre si

```
import random
lista = []
while len(lista) < 15:
    x = random.randint(10, 100)
    if x not in lista:
        lista.append(x)
lista.sort()
print (lista)
```

Lista de Exercícios



"A vida é como andar de bicicleta. Para manter o equilíbrio, é preciso se manter em movimento". Einstein.