1 Programas Python: laços for e strings

VOCÊ NÃO PRECISA SABER PYTHON PARA TERMINAR ESTA DISCIPLINA!

Pode ignorar este texto se quiser!

(Mas se estiver curioso/curiosa para conhecer uma linguagem que é mais próxima de pseudocódigo, este texto pode ajudar)

1.1 Laços FOR

1.1.1 Listas

Em Python, uma lista de elementos é escrita `[a, b, c, ...]`

1.1.2 Intervalos

Em Python existe um tipo de dados chamado "range" (intervalo).
range(a,b,p) é o intervalo de a (INCLUSIVE) até b (NÃO INCLUÍDO), andando `p` de cada vez. A, b e p dvem ser inteiros.
Veja:

```
range(1,4) == [1, 2, 3]
range(1,1) == []
range(0,10,2) == [0, 2, 4, 6, 8]
```

Como você pode perceber, intervalos são maneiras simples de especificar listas. Você pode usar variáveis para especificar intervalos:

```
a = 10
b = 20

range(a,b,3) == [10, 13, 16, 19]
```

Para conseguir intervalos com números de ponto flutuante, você precisa importar "arange":

```
from Numeric import arange
```

Depois de incluir esta linha em seu programa, você pode usar arange:

```
arange(0.0,2.0,0.2) == array([ 0. ,  0.2,  0.4,  0.6,  0.8,  1. ,  1.2,  1.4,  1.6,  1.8])
```

1.1.3 O laço for

Em Python o laço for é da seguinte forma:

```
for VAR in LISTA:
    COMANDOS
```

Isto significa que os comandos serão repetidos, com a variável VAR assumindo cada valor na LISTA.
Por exemplo, o programa:

```
for i in [1, -1, 0]:
    print 2 * i
```

Produzirá a saída:

```
2
-2
0
```

porque a lista tem três valores, e o comando "`print 2 * i`" foi repetido uma vez para cada valro de i. Como um intervalo é só uma maneira simples de especificar uma lista, podemos usá-los nos laços for:

```
for i in range(10,5,-1):
    print 2 * i
```

A saída será:

```
20
18
16
14
12
```

1.2 Strings

```
a = "abcdef"

a[0] == "a"
a[1] == "b"
a[2] == "c"
```

Substrings em Python são obtidas com `string[a:b]`, que retorna a parte de string que começa em a e vai até b (mas **não** inclui o b-ésimo caracter:

```
a = "abcdef"

a[1:2] == "b"
a[1:3] == "bc"
```

IMPORTANTE: Em Python, curiosamente, não existe tipo "caracter". Assim, o i-ésimo elemento de uma string é uma outra string de tamanho um:

```
a = "abcdef"

a[1] == "b"

x = "k"
x[0] == "k"
x[0] == x
```

Ou seja, para uma string `x` de tamanho um, `x` é igual a `x[0]`.