Curso Python Avançado

Back to Basics, repeat!

olist

Listas

Python Basics

Listas

Python possui diversos tipos de dados compostos (compound), o mais conhecido e usado é a lista.

```
>>> numeros = [1, 4, 9, 16, 25]
>>> numeros
[1, 4, 9, 16, 25]
```

Python Basics

Listas

Itens de uma lista podem ser de tipos diferentes.

```
>>> lista_tipos = [1, 'uma', 9,
'string', 25]
>>> lista_tipos
[1, 'uma', 9, 'string', 25]
```

Python Basics

Listas

Semelhante a uma string, `len()` também funciona com listas.

```
>>> lista_tipos = [1, 'uma', 9,
'string', 25]
>>> len(lista_tipos)
5
```

Python Basics

Listas

Strings, listas e outras estruturas são considerados tipos sequenciais (built-in sequence types).

Deste modo, existe uma série de operações comuns sobre estes tipos…
E, assim como fizemos anteriormente com strings, podemos:

```
>>> numeros = [1, 4, 9, 16, 25]
>>> numeros[0]
1
>>> numeros[-1]
25
>>> numeros[-3:]
[9, 16, 25]
>>> squares[:]
[1, 4, 9, 16, 25]
```

Importante: Índices retornam o elemento, fatias (slices) retornam uma nova lista!

Python Basics

Listas

Também é possível concatenar listas.

```
>>> numeros = [1, 4, 9, 16, 25]
>>> numeros + [36, 49, 64, 81, 100]
[1, 4, 9, 16, 25, 36, 49, 64, 81,
100]
```

Python Basics

Listas

Uma diferença entre strings e listas é que strings são mutáveis: é possível alterar o valor de seus elementos.

```
>>> numeros = [1, 4, 9, 16, 25]
>>> numeros[2] = 42
>>> numeros
[1, 4, 42, 16, 25]
```

Python Basics

Listas

Listas podem crescer e diminuir.

```
>>> numeros = [1, 4, 9, 16, 25]
>>> numeros_maior_1 = numeros
>>> numeros_maior_1.append(42)
>>> numeros_maior_2 = numeros + [42]
>>>
>>> numeros_menor_1 = numeros[2:]
>>> numeros_menor_2 = numeros
>>> del numeros_menor_2[:2]
```

Experimente também .extend(), insert().
Os operadores +, -, *, / funcionam entre listas ou listas e itens? Experimente!

Python Basics

Listas

Outros exemplos de operações.

```
>>> letras = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letras
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letras[2:5] = ['C', 'D', 'E']
>>> letras
['a', 'b', 'C', 'D', 'E', 'f', 'g']
>>> letras[2:5] = []
>>> letras
['a', 'b', 'f', 'g']
>>> letras[2:2] = [1, 2]
>>> letras
['a', 'b', 1, 2, 'f', 'g']
>>> letras[:] = []
>>> letras
[]
```

Python Basics

Listas

E, aumentando a complexidade, é possível aninhar listas.

```
>>> a = ['a', 'b', 'c']
>>> n = [1, 2, 3]
>>> x = [a, n]
>>> x
[['a', 'b', 'c'], [1, 2, 3]]
>>> x[0]
['a', 'b', 'c']
>>> x[0][1]
'b'
```

Exercícios de Lista

Exercícios de Lista

Exercício 1

Usando a IDE, crie um arquivo com o nome "exercicio_lista_01.py" com o seguinte conteúdo:

```
pow2 = [2 ** x for x in range(10)]
print(pow2)
```

Lembrete: `**` é operador de potência.

Exercícios de Lista

Exercício 1

Usando a IDE, crie um arquivo com o nome "exercicio_lista_01.py" com o seguinte conteúdo:

```
pow2 = [2 ** x for x in range(10)]
print(pow2)
```

Lembrete: `**` é operador de potência.

Isto é chamado de *"List comprehension"*: Uma *expressão* seguida de um `for` dentro de `[]`.

Equivalente a este codigo:

```
pow2 = []
for x in range(10):
    pow2.append(2 ** x)
```

Exercícios de Lista

Exercício 1

Experimentos:

- Altere a expressão e o range. Use a criatividade!

Exercícios de Lista

Exercício 1 - Continuação

```
pow2 = [2 ** x for x in range(10) if x > 5]
odd = [x for x in range(20) if x % 2 == 1]
conc = [x+y for x in ['Python ','C '] for y in
['Language','Programming']]
```