

Python

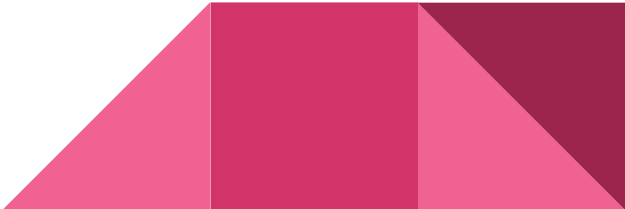
Curso Intensivo

Prof. Cláudio Fleury
Abr-22

Curso baseado no
livro de Eric Mattes,
**Python Crash
Course**

Cap.4 - Mais
Listas

Conteúdo

1. Dividindo o Conteúdo de uma String em Palavras
 2. Laço para Acessar todos os Itens de uma Lista
 3. Lista de Valores Numéricos
 4. Estatísticas com Valores Numéricos de uma Lista
 - 5. Abrangência de Lista**
 6. Fatiando uma Lista
 7. Copiando uma Lista
 - 8. Tuplas**
 9. PEP-8
 10. Resumo
- 

Dividindo o Conteúdo de uma String em Palavras

- A maneira mais fácil de obter a primeira palavra em uma string é acessar o primeiro elemento da lista que é retornada pelo método string **split()**
- Método **split()** - divide a string em uma lista de palavras separadas por um caractere específico (padrão é espaço em branco), fornecido como parâmetro de entrada

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

► colar4

```
1 string = "arroz,feijão,batata,carne,cenoura,vagem"
2 alimentos = string.split(',')
3 print(f"Eu gosto de comer {alimentos[0]}, {alimentos[1]} e {alimentos[3]}")
```

```
Eu gosto de comer arroz, feijão e carne.
[Finished in 692ms]
```

Laço para Acessar todos os itens de uma Lista

- Frequentemente precisaremos acessar todos os itens de uma lista, como por exemplo, quando se quer apresentar os elementos da lista na tela...

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

colar4

```
import this
bikes6.py
escritores3.py
carros2.py

1 escritores = ["Machado de Assis", "Clarice Lispector",
2             "Guimarães Rosa", "Jorge Amado", "Graciliano Ramos",
3             "Chico Buarque", "Carlos Drummond de Andrade", "Oswald de Andrade"]
4 for escritor in escritores:
5     print(escritor, "é um excelente escritor brasileiro!")
6     print(f"Eu gosto de ler as obras de {escritor.split()[0]}.\\n")
7 print("\\nParabéns a todos eles!\\n")
```

Carlos Drummond de Andrade é um excelente escritor brasileiro!
Eu gosto de ler as obras de Carlos.

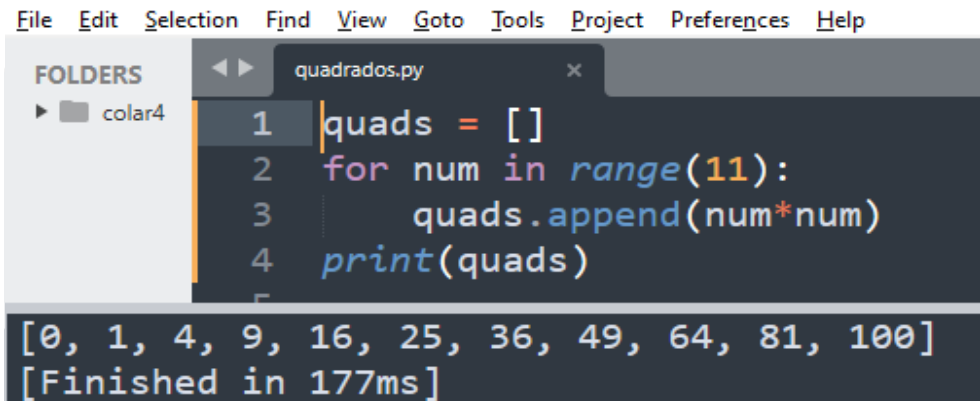
Oswald de Andrade é um excelente escritor brasileiro!
Eu gosto de ler as obras de Oswald.

Parabéns a todos eles!

[Finished in 154ms]

Lista de Valores Numéricos

- Lista de valores numéricos é muito comumente usada para armazenar dados de temperatura, distâncias, populações, localizações, abscissas e ordenadas etc.
- A função **list()** converte uma sequência de valores numéricos em lista Python



```
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  ▶ colar4
quadrados.py
1 quads = []
2 for num in range(11):
3     quads.append(num*num)
4 print(quads)
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
[Finished in 177ms]
```

A função **range(início,final)** gera valores do **início** (inclusive) até o **final** (exclusivo), ou seja, outra função com o comportamento menos um (-1). O resultado não inclui o limite superior indicado como argumento na função

Se somente um argumento for informado, então a faixa inicia-se em zero e termina no valor informado menos um

Estatísticas com Valores Numéricos de uma Lista

- Lista com o peso dos bois de uma fazenda
- Boi mais leve, mais pesado
- Peso médio
- Peso total

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

► colar4

```
1 pesos = [232, 245, 239, 410, 332, 222, 287]
2 print(f"Boi mais leve..: {min(pesos)} kg")
3 print(f"Boi mais pesado: {max(pesos)} kg")
4 print(f"Peso médio.....: {sum(pesos)/len(pesos)} kg")
5 print(f"Peso total.....: {sum(pesos)} kg")
```

```
Boi mais leve..: 222 kg
Boi mais pesado: 410 kg
Peso médio.....: 281.0 kg
Peso total.....: 1967 kg
[Finished in 167ms]
```

Abrangência da Lista (*List Comprehension*)

- A abordagem descrita anteriormente para gerar a lista de quadrados usou 3 ou 4 linhas de código
- A abrangência de lista permite a geração da mesma lista com apenas uma linha de código, pois combina o laço **for** e a criação de novos itens da lista em uma linha
- É preciso prática na escrita de suas próprias abrangência de listas

Sintaxe:

`lista = [expressão for geração]`

expressão define os valores que serão armazenados na lista.

laço **for** para gerar os números que serão jogados na expressão

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

▶ colar4



quadrados.py



bois.py



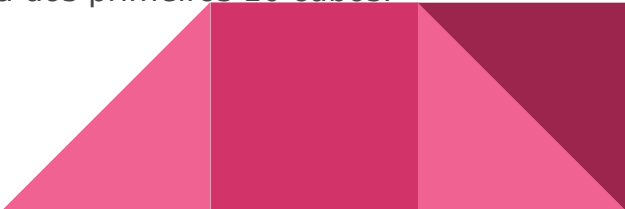
quadrados2.py



```
1 quads = [valor**2 for valor in range(1, 11)]
2 print(quads)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
[Finished in 845ms]
```

Exercícios

- 1. Contando até vinte:** Use um loop `for` para imprimir os números de 1 a 20, com limites inclusos.
 - 2. Um Milhão:** Faça uma lista dos números de um a um milhão e, em seguida, use um loop **`for`** para imprimir os números. (Se a saída estiver demorando muito, interrompa-a pressionando `ctrl-C` ou fechando a janela de saída.)
 - 3. Somando um milhão:** faça uma lista dos números de um a um milhão, e, em seguida, use `min()` e `max()` para garantir que sua lista realmente comece em um e termina em um milhão. Além disso, use a função `sum()` para ver a rapidez com que o Python pode adicionar um milhão de números.
 - 4. Números ímpares:** Use o terceiro argumento da função `range()` para fazer uma lista dos números ímpares de 1 a 20. Use um loop **`for`** para imprimir cada número.
 - 5. Três:** Faça uma lista dos múltiplos de 3, de 3 a 30. Use um loop **`for`** para imprimir os números da sua lista.
 - 6. Cubos:** Um número elevado à terceira potência é chamado de cubo. Por exemplo, o cubo de 2 é escrito como `2**3` em Python. Faça uma lista dos primeiros 10 cubos (que é, o cubo de cada inteiro de 1 a 10), e use um loop **`for`** para imprimir o valor de cada cubo.
 - 7. Abrangência de cubos:** Use uma abrangência de lista para gerar uma lista dos primeiros 10 cubos.
- 

Fatiando uma Lista

- Especifique o índice do primeiro e do último elemento da fatia da lista que se quer trabalhar: **lista[início:fim]**
- Se **início** não for indicado, a fatia inicia no primeiro item da lista
- Se **fim** não for indicado, a fatia finaliza no último item da lista
- Assim como a função **range()**, o Python também para a contagem um item antes do segundo índice especificado na fatia da lista

Lembre-se de que um índice negativo retorna um elemento a uma certa distância do final da lista

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

► colar4

jogadores3.py

```
1 jogadores = ['maria', 'marta', 'miguel', 'joão', 'eli']
2 print(jogadores[-2:])
3 print(jogadores[:-2])
```

```
['joão', 'eli']
['maria', 'marta', 'miguel']
[Finished in 162ms]
```

cópia funda: usa *slice*
cópia rasa: não usa *slice*

Copiando uma Lista

- Para copiar uma lista, faça uma fatia que inclua toda a lista original omitindo o primeiro índice e o segundo índice, `[:]`
- Isso diz ao Python para fazer uma fatia começando no primeiro item e terminando no último item, produzindo uma cópia de toda a lista

```
>>> jogadores = ['maria', 'marta', 'miguel', 'joão', 'eli']
>>> id(jogadores)
2007469807040
>>> reservas = jogadores[:]
>>> id(reservas)
2007469808064
>>> reservas = jogadores
>>> id(reservas)
2007469807040
>>>
```

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

► colar4

copia_lista2.py

```
1 jogadores = ['maria', 'marta', 'miguel', 'joão', 'eli']
2 reservas = jogadores
3 print(jogadores, reservas)
4 jogadores.append("zico")
5 print(jogadores, reservas)
```

```
['maria', 'marta', 'miguel', 'joão', 'eli'] ['maria', 'marta', 'miguel', 'joão', 'eli']
['maria', 'marta', 'miguel', 'joão', 'eli', 'zico'] ['maria', 'marta', 'miguel', 'joão', 'eli', 'zico']
[Finished in 176ms]
```

Exercícios

- 1. Fatias:** Usando um dos programas que você escreveu neste capítulo, adicione várias linhas ao final do programa que fazem o seguinte:
 - Imprime a mensagem “Os três primeiros itens da lista são:”. Em seguida, use uma fatia para imprimir os três primeiros itens da lista desse programa.
 - Imprime a mensagem “Três itens do meio da lista são:”. Use uma fatia para imprimir três itens do meio da lista.
 - Imprime a mensagem “Os três últimos itens da lista são:”. Use uma fatia para imprimir o três últimos itens da lista.
- 2. Minhas Pizzas, Suas Pizzas:** Crie uma lista com as suas pizzas favoritas. Faça uma cópia da lista de pizzas e chame-a de `friend_pizzas`. Então, faça o seguinte:
 - Adicione uma nova pizza à lista original.
 - Adicione uma pizza diferente à lista `friend_pizzas`.
 - Prove que você tem duas listas separadas. Imprima a mensagem “Minhas pizzas favoritas são:” e, em seguida, use um loop **for** para imprimir a primeira lista. Imprima a mensagem “As pizzas favoritas do meu amigo são:” e, em seguida, use um loop **for** para imprimir a segunda lista. Certifique-se de que cada nova pizza esteja armazenada na lista apropriada.
- 3. Mais Loops:** Todas as versões de **foods.py** nesta seção evitaram o uso de loops ao imprimir, para economizar espaço. Escolha uma versão de **foods.py** e escreva dois laços **for** para imprimir cada lista de comidas.

Tuplas

- ◆ São **Listas** imutáveis
- ◆ Algumas vezes precisamos de uma lista de itens que não mudam, e para esses casos temos as **Tuplas**
- ◆ Uma tupla se parece com uma lista, exceto que usamos parênteses como delimitadores, em vez de colchetes
- ◆ Podemos acessar elementos individuais da **tupla** via índice de cada item, assim como usamos para uma **lista**

```
dois elementos --> d upla  
três elementos --> tr ipla  
quatro elementos -> quadr úpla  
cinco elementos -> quint úpla  
...  
t elementos      --> tupla
```

Tuplas, tecnicamente, são definidas pela presença de uma vírgula e delimitadas por parênteses.

Para definir uma **tupla** com um elemento, inclua uma vírgula à direita:

```
minha_tupla = (3,)
```

File Edit Selection Find View Goto Tools Project Preferences Help

```
FOLD  platypus_doc.py x  dimensoes.py x  
1  dimensoes = (200, 50)  
2  print(dimensoes[0])  
3  print(dimensoes[1])  
4  dimensoes[0] = 250
```

200

50

Traceback (most recent call last):

```
File "D:\python\meus_scripts\dimensoes.py", line 4, in <module>  
    dimensoes[0] = 250
```

TypeError: 'tuple' object does not support item assignment

[Finished in 577ms]

Acessando todos os itens de uma Tupla

- ◆ Podemos acessar todos elementos de uma **tupla** via índice de cada item, assim como usamos para uma **lista**
- ◆ Quando comparadas às listas, as tuplas são estruturas de dados simples
- ◆ São usadas no armazenamento de valores que não devem ser alterados ao longo da vida de um programa

Embora não possamos modificar uma tupla, podemos atribuir um novo valor à variável que representa a tupla.

Ex.:

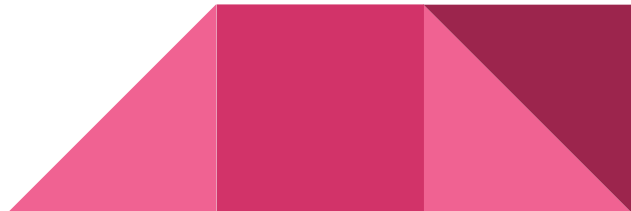
```
dimensoes = (3,5,1.5)
```

```
File Edit Selection Find View Goto Tools Project Preferences Help
FOLD platypus_doc.py x dimensoes2.py
1 dimensoes = (200, 50, 120)
2 for dimensao in dimensoes:
3     print(dimensao)

200
50
120
[Finished in 167ms]
```

Exercícios

- 1. Buffet:** Um restaurante estilo *buffet* oferece apenas cinco alimentos básicos. Pense em cinco alimentos simples e guarde-os numa tupla.
 - Use um laço **for** para imprimir cada comida que o restaurante oferece.
 - Tente modificar um dos itens e certifique-se de que o Python rejeite a mudança.
 - O restaurante muda seu cardápio, substituindo dois dos itens por diferentes alimentos. Adicione uma linha que reescreva a tupla e, em seguida, use um laço **for** para imprimir o menu revisado.

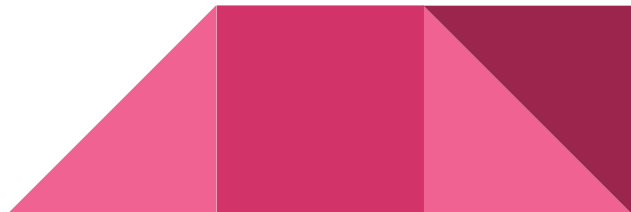


Estilizando seu Código

- Programadores Python adotaram várias convenções de estilo para garantir que o código de todos seja estruturado aproximadamente da mesma forma
- Depois de aprender a escrever código Python limpo, você será capaz de entender a estrutura geral do código Python de qualquer outra pessoa, desde que todos sigam as mesmas diretrizes
- Para tornar-se um programador profissional algum dia, comece a seguir estas diretrizes o quanto antes para desenvolver bons hábitos
- Guia de Estilo
 - Proposta de Aprimoramento do Python (*Python Enhancement Proposal* - PEP). Um dos PEPs mais antigos é o [PEP-8](#), que instrui os programadores Python sobre como estilizar o código

PEP-8

1. Indentação: quatro espaços, preferência à tabulação por se mais rápido e exato que inserção de espaços (TAB deve inserir espaços)
2. Comprimento de linha: no máximo 80 caracteres
3. Comentários de linha: no máximo 72 caracteres
4. Linhas em branco (agrupamentos de comandos) - linhas em branco não afetam a execução do código, mas afetam a legibilidade do código



Resumo

- Como trabalhar com elementos em uma lista
 - Como percorrer uma lista usando o laço **for**
 - Como criar listas numéricas simples
 - Fazendo operações em listas numéricas
 - Como fatiar uma lista para obter um subconjunto de itens
 - Como copiar uma lista (cópia funda/rasa)
 - Definição e manipulação de **Tuplas**
 - PEP-8
- 