FIAP GRADUAÇÃO

FIAP

TDS

Computacional Thinking using Python
Coleções – Listas e Tuplas

Prof. Dr. Daniel Trevisan Bravo

COLEÇÕES
LISTAS E TUPLAS

LISTAS

- Lista é uma coleção de valores indexada, em que cada valor é identificado por um índice. O primeiro item na lista está no índice 0, o segundo no índice 1 e assim por diante.

- Para criar uma lista com elementos deve-se usar colchetes e adicionar os itens entre eles separados por vírgula.

```
1 programadores = ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
2 print(type(programadores)) # type ‘list’
3 print(len(programadores)) # 5
4 print(programadores[4]) # Luana
```

- OBS: “len (<lista>) retorna o tamanho da lista (quantidade de elementos).

LISTAS

- Principal característica: são mutáveis, podendo ser alteradas depois de terem sido criadas. Em outras palavras, podemos adicionar, remover e até mesmo alterar os itens de uma lista.

```
programadores = ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
print(programadores) # ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']

programadores[1] = 'Carolina'
print(programadores) # ['Victor', 'Carolina', 'Samuel', 'Caio', 'Luana']
```

- podem possuir diferentes tipos de elementos na sua composição. Isso quer dizer que podemos ter strings, booleanos, inteiros e outros tipos diferentes de objetos na mesma lista

```
aluno = ['Murilo', 19, 1.79] # Nome, idade e altura

print(type(aluno)) # type 'list'
print(aluno) # ['Murilo', 19, 1.79]
```

LISTAS – SELECIONANDO INTERVALOS

- Podemos selecionar um trecho de uma lista:

```
lista[start:stop:step]
```

- O trecho inicia na posição start (inclusive) e vai até a posição stop (exclusive), selecionando de step em step os elementos da lista.
- Esta operação retorna uma nova lista, à qual normalmente nos referimos como uma sublista.
- Caso os parâmetros start, stop ou step não sejam especificados, Python automaticamente assume que seus valores são a posição do primeiro elemento (0), o tamanho da lista (len(lista)) e um (1), respectivamente.

LISTAS – SELECIONANDO INTERVALOS

- Exemplo:
  - Selecionando do segundo até o quarto elemento de uma lista:

```
letras = ["A", "B", "C", "D", "E", "F", "G", "H"]
print(letras[1:4])
# ["B", "C", "D"]
```

  - Selecionando os três primeiros elementos de uma lista:

```
letras = ["A", "B", "C", "D", "E", "F", "G", "H"]
print(letras[:3])
# ["A", "B", "C"]
```

  - Selecionando os quatro últimos elementos de uma lista:

```
letras = ["A", "B", "C", "D", "E", "F", "G", "H"]
print(letras[-4:])
# ["E", "F", "G", "H"]
```

LISTAS – VERIFICANDO A INCLUSÃO DE UM ELEMENTO

- Podemos verificar se um elemento está ou não em uma lista utilizando o operador de teste de inclusão in.

```
elemento in lista
```

- Esse operador retorna True ou False caso o elemento esteja ou não na lista, respectivamente.
- Exemplo:

```
top5 = ["Black Mirror", "Breaking Bad", "Friends",
  "Game of Thrones", "The Big Bang Theory"]
print("House MD" in top5)
# False
print("Game of Thrones" in top5)
# True
print("friends" in top5)
# False
```

LISTAS – OBTENDO A POSIÇÃO DE UM ELEMENTO

- O método index é utilizado para obter a posição de um elemento em uma lista.
- Como parâmetro, o método index recebe um elemento a ser buscado na lista.
- A posição da primeira ocorrência do elemento especificado como parâmetro é retornada como resposta.
- Caso o elemento não esteja na lista, um erro será gerado.

```
cinema = ["Sony Pictures", "Walt Disney",
  "Universal Pictures", "Warner"]
print(cinema.index("Warner"))
# 3
print(cinema.index("Disney"))
# ValueError: 'Disney' is not in list
```

LISTAS - OPERAÇÕES

- É possível adicionar itens nelas, pois já vêm com uma coleção de métodos predefinidos que podem ser usados para manipular os objetos que ela contém.

- Método **append**: adiciona elementos no final de uma lista.

```
programadores = ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
print(programadores) # ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana

programadores.append('Renato')
print(programadores) # ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana', 'Renato']
```

LISTAS - OPERAÇÕES

- Método **insert**: adiciona elemento em uma posição específica de uma lista. Ele usa dois parâmetros: o primeiro para indicar a posição da lista em que o elemento será inserido e o segundo para informar o item a ser adicionado na lista

```
programadores = ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
programadores.insert(1, 'Rafael')

print(programadores) # ['Victor', 'Rafael', 'Juliana', 'Samuel', 'Caio', 'Luana']
```

LISTAS - OPERAÇÕES

- Métodos **remove** e **pop**: remove() para a remoção pelo valor informado no parâmetro, e pop() para remoção pelo índice do elemento na lista

```
1  programadores = ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
2  print(programadores) # ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
3
4  programadores.remove('Victor')
5  print(programadores) # ['Juliana', 'Samuel', 'Caio', 'Luana']
```

```
1  programadores = ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
2  print(programadores) # ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
3
4  programadores.pop(0)
5  print(programadores) # ['Juliana', 'Samuel', 'Caio', 'Luana']
```

TUPLAS

- Tupla é uma estrutura de dados semelhante a lista. Porém, ela tem a característica de ser imutável, ou seja, após uma tupla ser criada, ela não pode ser alterada.
- No Python ela deve ser delimitada por parênteses na sua sintaxe.

```
1  times_rj = ('Botafogo', 'Flamengo', 'Fluminense', 'Vasco')
2
3  print(type(times_rj)) # class='tuple'
4  print(times_rj) # ('Botafogo', 'Flamengo', 'Fluminense', 'Vasco')
```

- Assim como é feito nas listas, podemos acessar um determinado valor na tupla pelo seu índice.

```
1  times_rj = ('Botafogo', 'Flamengo', 'Fluminense', 'Vasco')
2
3  print(times_rj[2]) # Fluminense
```

TUPLAS

- Observações importantes:
  - Se ela tiver um único item, é necessário colocar uma vírgula depois dele, pois caso contrário, o objeto que vamos obter é uma string, porque o valor do item é do tipo string.

```
1  objeto_string = ('tesoura')
2  objeto_tupla = ('tesoura',)
3
4  print(type(objeto_string)) # class 'str'
5  print(type(objeto_tupla)) # class 'tuple'
```

  - O fato da tupla ser imutável faz com que os seus elementos não possam ser alterados depois dela já criada.

```
1  vogais = ('a', 'e', 'i', 'o', 'u')
2
3  vogais[1] = 'E'
```

  - São aplicados os mesmos conceitos das listas às tuplas quanto à seleção de intervalos, procura de um elemento contido na tupla e ao retorno do índice do elemento.

TUPLAS – QUANDO USAR?

- Devem ser usadas em situações em que não haverá necessidade de adicionar, remover ou alterar elementos de um grupo de itens. Exemplos bons seriam os meses do ano, os dias da semana, as estações do ano etc. Nesses casos, não haverá mudança nesses itens (pelo menos isso é muito improvável).

PERCORRENDO LISTAS - ENUMERATE

- Enumerate: essa estrutura vai percorrer uma lista com índices, ou seja, você vai pegar o índice do item além da informação desejada dessa lista.
- O enumerate permite percorrer não só uma lista, mas qualquer estrutura "iterable", ou seja, algo que você pode percorrer.

Exemplo **SEM** enumerate

```
vendedores = ["Marcus", "Amanda", "Ale", "Carol"]
vendas = [15, 20, 10, 30]
```

```
# Vendedor 1: Marcus - 15 produtos
# Vendedor 2: Amanda - 20 produtos

for vendedor in vendedores:
    print(vendedor)
```

```
Marcus
Amanda
Ale
Carol
```

```
tamanho_lista = len(vendedores)
for i in range(tamanho_lista):
    print(vendedores[i])
    print(vendas[i])
```

```
Marcus
15
Amanda
20
Ale
10
Carol
30
```

Exemplo **COM** enumerate

```
for i, vendedor in enumerate(vendedores):
    print(vendedor)
    print(vendas[i])
```

```
Marcus
15
Amanda
20
Ale
10
Carol
30
```

EXERCÍCIOS

- Crie uma lista com os nomes dos super-heróis que devem participar da Iniciativa Vingadores seguindo a ordem:
  - Homem de Ferro
  - Capitão América
  - Thor
  - Hulk
  - Viúva Negra
  - Gavião Arqueiro

  Com essa lista, execute as seguintes operações:

  a) inclua o Homem-Aranha no final da lista;

  b) infelizmente a Viúva Negra e o Homem de Ferro não fazem mais parte da Iniciativa Vingadores, então retire-os da lista.

- Dada uma lista L de 10 valores inteiros definidos previamente, escreva um programa que exiba todos os elementos da lista (utilize o enumerate) e, na sequência, remova todos os números pares da lista. Por fim, exiba novamente a lista.

EXERCÍCIOS

- Dadas duas listas P1 e P2, ambas com 15 valores reais, definidos pelo usuário, que representam as notas de uma turma na prova 1 e na prova 2, respectivamente, escreva um programa que calcule e exiba a média da turma nas provas 1 e 2