Curso Preparatório

Semana 9:
Coleções: Tuplas e Sets

Admissão 2025

Instituto J&F
Germinare TECH

Coleções

- As estruturas compostas, também conhecidas como **coleções** permitem armazenar múltiplos itens dentro de uma única unidade, que funciona como um container. Entre as coleções, temos:
- **Tuplas:**
  - Coleção de valores indexada estrutura de dados semelhante a vetores. Ela tem a característica de ser imutável, ou seja, após uma tupla ser criada, ela não pode ser alterada.
- **Sets:**
  - Coleçoes não ordenadas, que não permite elementos duplicados, ou seja, cada elemento é único. Um set em si pode ser modificado, contudo os elementos contidos dentro dele precisam ser de tipos imutáveis.
- **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, pode ser manipulado.
- **Dicionários:**
  - Os dicionários representam coleções de dados que contém na sua estrutura um conjunto de pares chave/valor, nos quais cada chave individual tem um valor associado. Esse objeto representa a ideia de um mapa, que entendemos como uma coleção associativa desordenada. A associação nos dicionários é feita por meio de uma chave que faz referência a um valor.

Tuplas

- Para definir uma Tupla, basta atribuir um conjunto de informações para uma variável, porém essas informações devem estar entre parênteses.

- **estacoes = ("primavera", "verão", "outono", "inverno")**
- Isso é igual a:

“primavera”	“verão”	“outono”	“inverno”
0	1	2	3

- estacoes[0] = **"primavera“**
- estacoes[1] = **“verão"**
- estacoes[2] = **“outono"**
- estacoes[3] = **“inverno“**

- Tudo que aprendemos na manipulação de strings, serve para as Tuplas.
  - Textos são Tuplas de caracteres.

Tuplas

- Podem receber tipos numéricos também. Inclusive misturar INTEIROS, REAIS e TEXTOS em uma mesma Tupla. Mas tenha cuidado, você quem deve tratar tudo, não é a linguagem.

- **numeros = (5.5, 10.3, 6, 7.1, “10”)**
- Isso é igual a:

5.5	10.3	6	7.1	“10”
0	1	2	3	4

- numeros[0] = **5.5 (float)**
- numeros[1] = **10.3 (float)**
- numeros[2] = **6 (int)**
- numeros[3] = **7.1 (float)**
- numeros[4] = **“10” (string)**

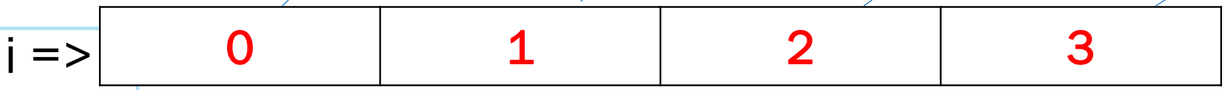
numeros[0]*2 **= 11**
numeros[2]*2 **= 20.6**
numeros[4]*2 **= 12**
numeros[4]*2 **= 14.2**
numeros[4]*2 **= 1010**

Tuplas

- Uma vantagem desses conjuntos em Python, está na estrutura para “varrer” (percorrer) todas as posições das coleções, porque podemos usar ela como intervalo em um **for**

- **estacoes = ("primavera", "verão", "outono", "inverno")**
- Isso é igual a:

for i **in** estacoes:



print (“A estação é”, i)
A estação é primavera
A estação é verão
A estação é outono
A estação é inverno

TUPLAS SÃO IMUTAVEIS:
Não pode adicionar, remover ou trocar valores de uma tupla

Tuplas

- Se quiser usar como era em Java, também, pode, mas ao invés de colocar a Tupla no intervalo, você deve colocar o **range()** e dentro do range, colocar o **len(tupla),** os códigos abaixo mostram as mesmas saídas:

```
for i in estacoes:
    print("A estação é",i)

for n in range (0, len(estacoes)):
    print("A estação é",estacoes[n])
```

Procura os itens
i = itens (valores) de estacoes

Varre os índices
n = índices de estacoes
(igual vetores em JAVA)

- No exemplo anterior tínhamos uma Tupla com 4 textos, se a gente quiser trocar ou editar um dos valores não conseguiríamos. EX:
- **estacoes[1] = “ferias“** <= ERROR porque **estacoes[1]** sempre será **“verão”.**

Outras opções das Tuplas:

- É possível pesquisar um elemento em uma tupla para verificar se ele existe. Para isso, utilizamos o operador **in** (*está em*), que retorna verdadeiro (**True**) ou falso (**False**) de acordo com o resultado da pesquisa.

“inverno” in estacoes => **True**
“hugo” in estacoes => **False**

- Podemos também ordenar uma tupla, usando o comando **sorted():**

estacoes = ("primavera", "verão", "outono", "inverno")
sorted(estacoes) => **['inverno', 'outono', 'primavera', 'verão']**

numeros = (5,7,1,4,9)
sorted(numeros) => **[1, 4, 5, 7, 9]**

Outras opções das Tuplas:

- É possível pesquisar um elemento em uma tupla para verificar se ele existe. Para isso, utilizamos o operador **in** (*está em*), que retorna verdadeiro (**True**) ou falso (**False**) de acordo com o resultado da pesquisa.

“inverno” in estacoes => **True**
“hugo” in estacoes => **False**

- Esse operador **in** também pode ser usado no **for** e no **if**

```
1  estacoes = ("primavera", "verao", "outono", "inverno")
2  escolha = input("Qual sua estacão favorita?")
3  if escolha in estacoes:
4      print("Eu gosto também do",escolha)
5  else:
6      print("Ah, você não gosta de",estacoes)
```

Outras opções das Tuplas:

- Podemos também usar o **in** para validar valores em um **if**, ao invés de usar os operadores lógicos (**and, o**r), ex:

```
1   acumulador = 0
2   while True:
3       numero = int(input("Digite um numero: "))
4       acumulador += numero
5       opcao = input("Deseja continuar? [s/n]: ")
6       if opcao.lower() in ('n', 'não', 'nao'):
7           print("FIM")
8           break
9
10  print("A soma de todos os numeros é",acumulador)
```

Se fosse com **or** ficaria:
if opcao.lower() == 'n' or opcao.lower() == 'não' or opcao.lower() == 'nao':

- O operador **+** serve para juntar duas ou mais tuplas, nunca realizar a operação de soma de valores: EX:

a = (1,4,7,9)
b = (3,6,7,1)
c = a+b (**a** união com **b**)
print(c)
c = (1, 4, 7, 9, 3, 6, 7, 1)
Se: c = b+a (**b** união com **a**)
c = (3, 6, 7, 1, 1, 4, 7, 9)

```
a = (1,4,7,9)
b = (3,6,7,1)
c = []
for elemA, elemB in zip(a, b):
    c.append(elemA + elemB)
print(c)
```

c = [4, 10, 14, 10]

- Se quiser <u>somar</u>, tem que usar o **zip()** que permite juntar/mesclar dois iteráveis.

zip(iteravel1, iteravel2)

- O resultado é um objeto que quando iterado funciona como uma **lista**:

Tuplas - Prática

- Criar um programa que tenha uma Tupla preenchida de 1 à 10 por extenso, o usuário deverá digitar um numero inteiro nesse intervalo e, o programa vai buscar na tupla e mostrar por extenso qual o numero escolhido. O programa deve funcionar infinitamente até que o usuário digite um valor fora do intervalor (1-10)

EX: Digite um numero: 8 -> você digitou o numero oito;
Digite um numero: 4 -> você digitou o numero quatro;
Digite um numero: 14 -> FIM!!

```
1  numeros = ("um","dois","tres", "quatro", "cinco", "seis", "sete", "oito", "nove", "dez")
2
3  while True:
4      escolha = int(input("Escolha um numero entre 1 e 10: "))
5      if (escolha > 10):
6          print("Intervalo errado, encerrando....")
7          break
8      print("Você escolheu o numero",numeros[escolha-1])
```

Tuplas - Prática

- Crie um programa que gere 5 números aleatórios quaisquer e os coloque em uma Tupla. Mostre essa Tupla e diga qual o maior e menor valor.

- Para gerar números aleatórios em Python, pode usar a sintaxe abaixo:

from random **import** randint
n = randint(1,10) # numero aleatório entre 1 e 10.

- Para buscar os valores MÁXIMO e MÍNIMOS em uma tupla, pode usar o **max()** e **min()**:

```
1  from random import randint
2  numeros = (randint(1,10), randint(1,10), randint(1,10), randint(1,10), randint(1,10))
3  print(numeros)
4  print("O maior valor foi:",max(numeros))
5  print("O menor valor foi:",min(numeros))
```

Sets

Python também tem outra estrutura de coleções, que são os **sets. Sets e Listas** são utilizados tanto para armazenar elementos de diferentes tipo. A diferença é que Set é armazenada de maneira desordenada e não permite valores duplicados. List é usado para armazenar elementos de maneira ordenada e permite valores duplicados.

Set: elementos **não** podem ser acessados por uma posição de índice. Nem podem usar as funções das listas.
Lista: elementos podem ser acessados com uma posição de índice.

Os **SETs** são identificados/criados por chaves **{ }**

```
1   lista = [1,2,2,2,3,4,5]
2   print(lista)
3   set_lista = {1,2,2,2,5}
4   print(set_lista)
```

[1, 2, 2, 2, 3, 4, 5]

{1, 2, 5}