CIÊNCIA DE DADOS COM PYTHON

CAP.3 - COLEÇÕES

Prof. Renzo Paranaíba Mesquita

OBJETIVOS

- Compreender características dos principais tipos de coleções no Python: Tuplas, Listas, Conjuntos e Dicionários
- Praticar o uso destas estruturas e entender suas aplicabilidades em diferentes situações
- Compreender a importância destas estruturas e como elas são essenciais para a Ciência de Dados em Python

Inatel

TÓPICOS

1. Tuplas (Tuples)
2. Listas (Lists)
3. Conjuntos (Sets)
4. Dicionários (Dictionaries)

Inatel

3.1. TUPLAS (TUPLES)

- Quando criamos uma variável, estamos pedindo para que o computador reserve um espaço de memória para que ele possa guardar um valor lá dentro
- Em uma variável comum, só é possível guardarmos um único valor
- Porém, podemos também trabalhar com as chamadas **variáveis compostas (ou Coleções)**, que nos permitem armazenar uma série de valores ao mesmo tempo
- A forma mais simples de **Coleção** no Python é a **Tupla**
- **EXEMPLOS:**

nomes = (‘Goku’, ‘Vegeta’, ‘Trunks’, ‘Gohan’)

0 1 2 3

( )

- **OBSERVAÇÕES:**
  - Observe a organização dos elementos dentro de ( )

3.1. TUPLAS (TUPLES)

- Cada elemento da Tupla possui um índice
- Observe como acessar um ou mais elementos de uma Tupla
- **EXEMPLOS:**

nomes = (‘Goku’, ‘Vegeta’, ‘Trunks’, ‘Gohan’)

0 1 2 3

```
print(nomes) #(‘Goku’, ‘Vegeta’, ‘Trunks’, ‘Gohan’)
print(nomes[1]) #Vegeta
print(nomes[-2]) #Trunks
print(nomes[1:3]) #(‘Vegeta’, ‘Trunks’)
print(nomes[2:]) #(‘Trunks’, ‘Gohan’)
print(nomes[:2]) #(‘Goku’, ‘Vegeta’)
print(len(nomes)) #4
```

3.1. TUPLAS (TUPLES)

- Quando buscamos um ou mais elementos específicos dentro de uma Coleção, estamos realizando o chamado **SLICING** (fatiamento) de dados, igual fizemos no exemplo do slide anterior
- Porém, é importante ressaltar que uma **Tupla** é uma Coleção **IMUTÁVEL**, ou seja, **não permite que seus valores sejam alterados, excluídos e que novos valores sejam inseridos**
- **EXEMPLOS:**

```
nomes[1] = 'Bulma'
```

TypeError: 'tuple' object does not support item assignment

- Apesar da **Tupla** possuir esta característica, lembre-se que seus elementos podem ser acessados normalmente, até mesmo por meio de um **for**:
- **EXEMPLOS:**

```
for nome in nomes:
    print(nome)
```

3.1. TUPLAS (TUPLES)

- Muitas vezes só mostrar o valor não é o bastante, ou seja, também precisamos mostrar os índices:
- **EXEMPLOS:**

```
for cont in range (0, len(nomes)):
    print(f'Nome {cont}: {nomes[cont]}') #uma forma de apresentar os valores formatados
```

- As Tuplas também aceitam elementos de tipos diferentes:
- **EXEMPLOS:**

```
personagem = ('Goku', 37, 'Saiyajin', 85.5)
```

3.1. TUPLAS (TUPLES)

- Outros exemplo de operações com Tuplas:

- **EXEMPLOS:**

```
print(sorted(nomes)) #[‘Gohan’, ‘Goku’, ‘Trunks’, ‘Vegeta’]
#ordem alfabética  #observe o retorno dos elementos ordenados dentro de [ ] 🤔 ???

x = (2, 6, 8)
Y = (5, 6, 9, 1)
z = x + y
print(z) #(2, 6, 8, 5, 6, 9, 1)

print(z.count(6)) #duas ocorrências de 6 no vetor z

print(max(z), min(z)) #maior e menor elemento da tupla

print(z.index(5)) #pega o índice onde se encontra um valor
```

3.2. LISTAS (LISTS)

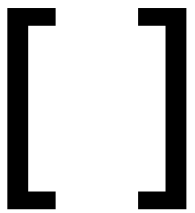
- A **Lista** também é um tipo de variável composta, porém, é uma coleção **MUTÁVEL**
- Permite que **elementos sejam acessados (igual na Tupla), alterados, excluídos e novos elementos sejam inseridos**
- Observe a organização dos elementos dentro de [ ]
- **EXEMPLOS:**

nomes = [‘Goku’, ‘Vegeta’, ‘Trunks’, ‘Gohan’]

0 1 2 3

nomes.append(‘Bulma’) # insere elemento no final da lista

nomes = [‘Goku’, ‘Vegeta’, ‘Trunks’, ‘Gohan’, ‘Bulma’]



nomes.insert(1, ‘Kuririn’) # insere elemento em uma posição específica

nomes = [‘Goku’, ‘Kuririn’, ‘Vegeta’,‘Trunks’, ‘Gohan’, ‘Bulma’]

3.2. LISTAS (LISTS)

- **EXEMPLOS:**

```
nomes = [‘Goku’, ‘Kuririn’, ‘Trunks’, ‘Gohan’, ‘Bulma’]

del nomes[2] # exclui um elemento pelo índice # também pode-se usar nomes.pop(2)
# nomes = [‘Goku’, ‘Kuririn’, ‘Gohan’, ‘Bulma’]

nomes.remove(‘Gohan’) # exclui elemento pelo valor
# nomes = [‘Goku’, ‘Kuririn’, ‘Bulma’]

nomes[1] = ‘Goten’ # alterando o valor de um índice
# nomes = [‘Goku’, ‘Goten’, ‘Bulma’]
```

3.2. LISTAS (LISTS)

- E se tentarmos remover um elemento que não existe na lista? Podemos verificar isso:

EXEMPLOS:

```
if ‘Vegeta’ in nomes:

   nomes.remove(‘Vegeta’)
```

- Para ordenação de elementos na lista, podemos utilizar a função **sorted()** do Python ou os métodos de ordenação da própria lista como **nomes.sort()** (ordem crescente) ou **nomes.sort(reverse=True)** (ordem decrescente)

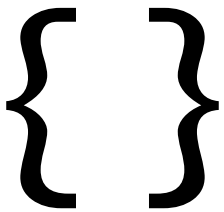
- **OBSERVAÇÕES:**
  - A lista também aceita armazenar elementos heterogêneos, porém, os métodos de ordenação não funcionam neste caso

3.3. CONJUNTOS (SETS)

- Um conjunto, diferente de uma lista, é uma coleção **NÃO ORDENADA** e que **NÃO ADMITE ELEMENTOS DUPLICADOS**
- Observe a organização dos elementos dentro de { }
- **EXEMPLOS:**

```
nomes = {‘Goku’, ‘Vegeta’, ‘Trunks’, ‘Gohan’, ‘Trunks’, ‘Goku’}

print(nomes) # Possível Saída: {‘Trunks’, ‘Goku’, ‘Gohan’, ‘Vegeta’}
```



- Permite a inserção de elementos por meio do método add() e remoção de elementos por meio do método remove()
- Para se atualizar elementos, é necessário remover o elemento desejado e em seguida adicionar outro

3.3. CONJUNTOS (SETS)

- Assim como acontece na matemática, aqui os **Conjuntos** também podem ser utilizados para realização de operações como **união, diferença e interseção**
- **EXEMPLOS:**

```
a = {2, 4, 6}

b = {1, 4, 5}


z = a | b #uniao

z = a - b #diferença

z = a & b #interseção
```

3.4. DICIONÁRIOS (DICTIONARIES)

- Os dicionários são variáveis compostas, **MUTÁVEIS**, mas que permitem se trabalhar com **ÍNDICES PERSONALIZÁVEIS (CHAVE:VALOR)**

- Observe a organização dos elementos também dentro de { }

- **EXEMPLOS:**

```
dados = {‘nome’: ‘Goku’, ‘idade’: 43}
print(dados[‘nome’]) # acessando valores

dados[‘sexo’] = ‘M’ # adicionando nova chave/valor
# dados = {‘nome’: ‘Goku’, ‘idade’: 43, ‘sexo’ = ‘M’}

del dados[‘sexo’] # removendo elementos
# dados = {‘nome’: ‘Goku, ‘idade’: 43}

dados[‘idade’] = 35 # alterando elementos
# dados = {‘nome’: ‘Goku, ‘idade’: 35}
```

{chave:valor}

3.4. DICIONÁRIOS (DICTIONARIES)

- Existem diferentes formas de se “printar” os elementos de um dicionário. São elas:

- **EXEMPLOS:**

```
dados = {‘nome’: ‘Goku’, ‘idade’: 43, ‘sexo’: ‘M’}

print(dados.values()) # mostrando todos os valores de um dicionário
# dict_values(['Goku', 43, 'M'])

print(dados.keys()) # mostrando todas as chaves de um dicionário
# dict_keys(['nome', 'idade', 'sexo'])

print(dados.items()) # mostrando todos os dados de um dicionário
# dict_items([('nome', 'Goku'), ('idade', 43), ('sexo', 'M’)])

for k,v in dados.items(): # varrendo os índices de uma lista
   print(f’{k} é {v})
```

3.4. DICIONÁRIOS (DICTIONARIES)

- Na Ciência de Dados, é mais que comum o aninhamento entre **Coleções**

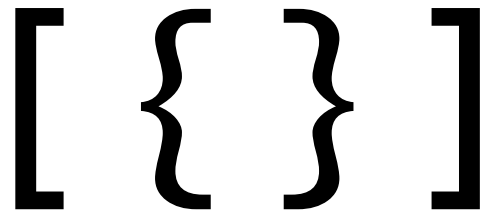
EXEMPLOS:

```
dbz = [dados1, dados2, dados3]

dados1 = {‘nome’: ‘Goku’, ‘idade’: 43, ‘sexo’: ‘M’}

dados2 = {‘nome’: ‘Gohan’, ‘idade’: 23, ‘sexo’: ‘M’}

dados3 = {‘nome’: ‘Pan’, ‘idade’: 5, ‘sexo’: ‘F’}
```




```
print(dbz[0][‘nome’]) #Goku

print(dbz[1][‘idade’]) #23

print(dbz[2][‘sexo’]) #F
```

EXERCÍCIOS (PARTE 1)

1. Crie uma lista preenchida com os 5 primeiros colocados de um Campeonato de Futebol, na ordem de colocação. Depois mostre:
   a. Apenas os 3 primeiros colocados;
   b. Os últimos 2 colocados;
   c. Uma lista com os times em ordem alfabética;
   d. Em que posição da tabela se encontra o Barcelona;

2. Crie dois conjuntos, um para cada loja. Identifique quais modelos de *smartphones* cada uma delas vendem. Em seguida, mostre quais modelos no total você terá opção de comprar se visita-las e quais modelos se encontram disponíveis em ambas as lojas;

3. Faça um programa que leia o nome e a média de um aluno e guarde-os em um dicionário. Em seguida, a partir da média (para ser aprovado deve ter média >=50), gere a situação final do aluno ('AP' ou 'RP'), que também deve ser guardada neste dicionário. No final, mostre todo o conteúdo deste dicionário;

EXERCÍCIOS (PARTE 2)

4. Faça um programa que leia o nome e peso de 3 pessoas e no final mostre o nome da pessoa mais pesada e a mais leve;

5. Desenvolva um programa que leia o nome, idade e sexo de n pessoas. No final, mostre:

a. A média de idade do grupo;
b. Quantas mulheres têm menos de 20 anos.

Dica: em Python, os operadores booleanos básicos são **and, or** e **not.**

inatel.tecnologias
inatel.tecnologias
inateloficial
company/inatel
www.inatel.br

Campus em Santa Rita do Sapucaí
Minas Gerais - Brasil
Av. João de Camargo, 510
Centro - 37536-001

CIÊNCIA DE DADOS COM PYTHON

FIM CAPÍTULO 3

Inatel

_o futuro
não tem hora,
mas tem lugar.