Inatel
Instituto Nacional de Telecomunicações

# Introdução à Análise de Dados

## Cap.3 - Coleções

Prof. MSc. Renzo P. Mesquita

# Objetivos

- Compreender em detalhes o princípio de funcionamento dos principais tipos de coleções em Python, no caso: as Tuplas, as Listas, os Conjuntos e os Dicionários;
- Compreender a importância destas estruturas e como elas são essenciais para a área de Análise de Dados;

# Capítulo 3
# Coleções

*3.1. Tuplas (Tuples);*
*3.2. Listas (Lists);*
*3.3. Conjuntos (Sets);*
*3.4. Dicionários (Dictionaries);*

# 3.1. Tuplas (Tuples)

*Quando criamos uma variável, estamos pedindo para que o computador reserve um espaço de memória para que ela possa guardar algum valor lá dentro.*

## *Exemplo:*

nome = 'Goku' 


Dentro de uma variável comum só é possível guardamos um único valor.

*Mas seria possível criarmos algum tipo de variável capaz de armazenar vários valores ao mesmo tempo?*

A resposta é SIM! Em Python, uma das formas de fazer isso é utilizando de variáveis compostas, como as TUPLAS.

# 3.1. Tuplas (Tuples)

*A tupla não é uma variável simples, mas um tipo de variável COMPOSTA.*

- Cada elemento dentro de uma Tupla é identificado por um índice.

*Exemplo:*

nomes = ('Goku','Vegeta','Trunks', 'Gohan')

*0 1 2 3*

- Da mesma forma que foi feito com as Strings, também é possível manipular as partes ou elementos das tuplas:
  - *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)

*As Tuplas são tipos de variáveis imutáveis, ou seja, não podem ter seus valores alterados.*

## *Exemplo:*

*nomes[1] = 'Bulma'*

=

TypeError: 'tuple' object does not support item assignment

*Lembrando também que podemos utilizar da estrutura de repetição for para caminhar sobre os elementos de uma tupla:*

## *Exemplo:*

```
print('Personagens do Dragon Ball:')
for nome in nomes:
    print(nome)
```

Personagens do Dragon Ball:
Goku
Vegeta
Trunks
Gohan

# 3.1. Tuplas (Tuples)

*Muitas vezes só mostrar o valor não é o bastante, mas também precisamos mostrar os índices correspondentes a cada um destes valores.*

## *Exemplo:*

```
for cont in range(0,len(nomes)):
    print(f'Nome {cont}: {nomes[cont]}')

for pos,nomes in enumerate(nomes):
    print(f'Nome {nomes}: {pos}')
```

- Veja que nesse tipo de iteração, também é possível mostrar os índices;
- Os dois laços acima fazem a mesma coisa mas de formas diferentes.

# 3.1. Tuplas (Tuples)

*Nas tuplas, é possível misturar elementos de tipos diferentes.*

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

('Goku', 37, 'Saiyajin', 85.5)

## *Outros exemplos de operações com Tuplas:*

- *print(sorted(nomes))* -> ['Gohan', 'Goku', 'Trunks', 'Vegeta']
  # ordem alfabética
- *x = (2,6,8)*
- *y = (5,6,9,1)* -> (2, 6, 8, 5, 6, 9, 1)
  # junção de tuplas
- *z = x + y*
- *print(z)*

- *print(z.count(6))* -> 2 #duas ocorrências de 6 no vetor z
- *print(max(z))* -> 9 #maior elemento da tupla

# 3.2. Listas (Lists)

*A lista também é um tipo de variável composta, porém, seus valores podem ser mudados e em vez de usar () como as tuplas, elas usam [] para organizar seus elementos.*

*Exemplo:*

```
nomes = ['Goku','Vegeta','Trunks', 'Gohan']
```

*0 1 2 3*

Se fizermos nomes[3] = 'Goten', teremos:

```
nomes = ['Goku','Vegeta','Trunks', 'Goten']
```

*0 1 2 3*

*E se quisermos adicionar novos valores dentro das listas?*

# 3.2. Listas (Lists)

*As listas podem crescer dinamicamente! Para adicionarmos novos valores dentro das listas podemos utilizar do comando append(valor).*

## *Exemplo:*

Se fizermos nomes.append('Bulma') teremos:

```
nomes = ['Goku','Vegeta','Trunks', 'Goten', 'Bulma']
         0      1        2         3        4
```

Já o comando insert(posicao,valor) permite adicionar um elemento em uma posição e deslocar os outros elementos:

Se fizermos nomes.insert('Kuririn') teremos:

```
nomes = ['Kuririn', 'Goku','Vegeta','Trunks', 'Goten', 'Bulma']
         0          1       2        3         4        5
```

# 3.2. Listas (Lists)

*Da mesma forma que podemos adicionar elementos, também podemos deletá-los. Para isso, podemos usar dos métodos del, pop() ou remove().*

*Exemplo:*

nomes = ['Kuririn', 'Goku','Vegeta','Trunks', 'Goten', 'Bulma']

*0 1 2 3 4 5*

Se fizermos del nomes[2], nomes.pop(2) ou nomes.remove('Vegeta') teremos:

nomes = ['Kuririn', 'Goku','Trunks', 'Goten', 'Bulma']

*0 1 2 3 4*

Atenção: observe que os métodos del e pop trabalham com índices e o remove com valores (ou conteúdo).

# 3.2. Listas (Lists)

*E se tentarmos remover um elemento que não existe na lista? Podemos verificar isso!*

```
if 'Vegeta' in nomes:
    nomes.remove('Vegeta')
```

Muitas vezes queremos também que os elementos da nossa lista fiquem ordenados, seja em ordem crescente, decrescente ou alfabética. Para isso podemos usar do método sort():

*Exemplo:*

Se fizermos nomes.sort() teremos:

```
nomes = ['Bulma','Goku','Goten','Kuririn','Trunks']
```

Se fizermos nomes.sort(reverse=true) teremos:

```
nomes = ['Trunks','Kuririn','Goten','Goku','Bulma]
```

# 3.3. Conjuntos (Sets)

*Um conjunto, diferente de uma lista, é uma coleção não ordenada e que não admite elementos duplicados. Outro detalhe é que eles usam {} para organizar seus elementos.*

*Exemplo:*

nomes = {'Goku','Vegeta','Trunks', 'Gohan', 'Trunks', 'Goku'}

*0 1 2 3 4 5*

print(nomes) -> {'Trunks', 'Goku', 'Gohan', 'Vegeta'}

Assim como acontece na matemática, aqui os conjuntos também podem ser utilizados para realização de operações como união, diferença e interseção.

*Exemplo:* a = {2,4,6} b = {1,4,5}

z = a | b #união z = a - b #diferença z = a & b #interseção

# 3.4. Dicionários (Dictionaries)

*Os dicionários são variáveis compostas como tuplas e listas mas que permitem se trabalhar com índices literais ou personalizáveis.*

Outro ponto importante: Dicionários são identificados por {}, ao invés de () (Tuplas) e [] (Listas).

*Exemplo:*

```
dados = { 'nome': 'Goku', 'idade': 43 }
```

*print(dados['nome']) -> Goku*
*print(dados['idade']) -> 43*

Observe que agora não temos mais índices 0, 1 ,2 etc..
Mas sim temos o índice 'nome', 'idade', etc..

# 3.4. Dicionários (Dictionaries)

*Os dicionários crescem dinamicamente e adicionar elementos dentro deles é muito simples.*

*Exemplo:*

*dados['sexo'] = 'M'*

dados = { 'nome': 'Goku', 'idade': 43, 'sexo': M }

Excluir elementos também é fácil, basta utilizarmos do comando del.

*Exemplo:*

*del dados['sexo']*

# 3.4. Dicionários (Dictionaries)

*Existem diferentes formas de "printar" os elementos de um dicionário. São elas:*

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

*1. Mostrando todas as informações de um dicionário:*

```
print(dados.values())
```

*2. Mostrando todas as chaves de um dicionário:*

```
print(dados.keys())
```

*3. Mostrando tudo de um dicionário:*

```
print(dados.items())
```

# 3.4. Dicionários (Dictionaries)

*Caminhar sobre as chaves e valores dos dicionários fazendo uso do laço de repetição for é uma ótima forma de ler seus elementos.*

*Exemplo:*

```
for k,v in dados.items():
    print(f'{k} é {v}')
```

nome é Goku
idade é 43
sexo é M

**Podemos também criar uma lista de dicionários!**

*Exemplo:*

*dbz = [dados1, dados2, dados 3]* em que:

```
dados1 = { 'nome': 'Goku', 'idade': 43, 'sexo': 'M' }
dados2 = { 'nome': 'Gohan', 'idade': 23, 'sexo': 'M' }
dados3 = { 'nome': 'Pan', 'idade': 5, 'sexo': 'F' }
```

Para printar um elemento dessa lista poderíamos usar do seguinte:

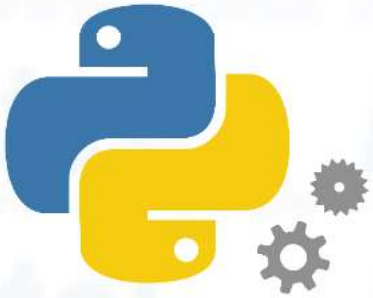
*print(dbz[0]['nome'])*  Goku

# 3.4. Dicionários (Dictionaries)

## *Exercícios Propostos*

1. Crie uma LISTA preenchida com os 5 primeiros colocados do Campeonato Mundial 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 está 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 visitá-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 (se for >= 60), gere a situação final do aluno 'AP' ou 'RP' e também a guarde no dicionário. No final mostre o conteúdo do dicionário;

# FIM
# DO
# CAPÍTULO 3

*Próximo Capítulo*

Análise de Dados com Numpy