Resumo – Estruturas de dados

Estruturas de dados

Algumas das principais estruturas de dados do Python são:

- Listas (`list`): `[1, 2, 3]`
- Tuplas (`tuple`): `(1,  2, 3)`
- Conjuntos (`set`): `{1, 2, 3}`
- Dicionários (`dict`): `{"a": 1, "b": 2, "c": 3}`

Listas (`list`)

Estrutura de dados que armazena elementos de forma **ordenada** (ordem em que foram inseridos). Por exemplo:k

```
lista = []
lista.append(5)
lista.append(3)
lista.append(7)
print(lista)
=> [5, 3, 7]
```

Os elementos em uma lista possuem **índices**:

```
lista = [1, 5, 3, 9]
#        0  1  2  3  índices
print(lista[0])
=> 1
```

```
print(lista[3])
=> 9
```

Vimos que `list` tem alguns métodos interessantes como:

- `lista.append(el)`: adiciona o elemento `el` ao final da lista.
- `lista.insert(pos, el)`: insere o elemento `el` na posição `pos`.
- `lista.pop()`: remove o último elemento da lista.
- `lista.sort()`: ordena os elementos da lista segundo algum critério. Por padrão, ordem crescente.

> 💡 Se quiser ver mais métodos de listas, você pode acessar o nosso Guia Rápido. Outra opção, é acessar a documentação oficial do Python sobre listas.

Tuplas (`tuple`)

Tuplas são estruturas ordenadas e, apesar de serem parecidas com listas, são normalmente utilizadas para representar objetos. E além disso, normalmente acessamos seus elementos utilizando a técnica de ***unpacking***:

```
pessoa = ("Gabriel", 27, "RJ")
nome, idade, estado = pessoa
print(nome)
=> "Gabriel"
print(idade)
=> 27
print(estado)
=> "RJ"
```

> 💡 Uma tupla após criada não pode ter seus elementos modificados! Isso porque é uma estrutura de dados ***imutável***.

Conjuntos

Conjuntos (`set`) são estruturas que remetem aos conjuntos da matemática, ou seja, possuem operação como: união, interseção, diferença, etc.

```
A = {1, 2, 3}
B = {1, 2, 4}
print(A.union(B))  # União de conjuntos A U B
=> {1, 2, 3, 4}

print(A.intersection(B)) # Interseção de conjuntos A ^ B
=> {1, 2}

print(A.difference(B))  # Diferença de conjuntos A - B
=> {3}
print(B.difference(A))  # Diferença de conjuntos B - A
=> {4}
```

Mais detalhes e métodos de conjuntos podem ser vistos na documentação oficial da linguagem.

Dicionários (`dict`)

Dicionários são estruturas que armazenam pares de chave-valor da seguinte maneira:

```
evento = {
  "nome": "Aula de Python",
  "data": "2022-02-07",
  "limite_alunos": 40,
}

print(evento["nome"])
=> "Aula de Python"

print(evento["limite_alunos"])
=> 40
```

Percorrendo estruturas de dados

Podemos percorrer estruturas utilizando o `while` ou o `for`.

Sintaxe do for:

```
for <elemento> in <estrutura que pode ser iterada>:
    <instrução 1>
    <instrução 2>
    ...
    <instrução n>
```

Exemplo `while` vs `for`

```
# Imprimir todos elementos de uma lista
lista = [1, 3, 5, 10]

# Com while
i = 0
while i < len(lista):  # len(lista) retorna o tamanho da lista
  print(lista[i])

# Com o for
for el in lista:
  print(el)
```