

Curso Preparatório

Semana 10:

Coleções: Dicionários

Admissão 2025



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:**
 - Coleções 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.



Dicionários

- Os dicionários são criados por chaves e dois pontos - `{}` e `:` - e representam coleções de dados que contém na sua estrutura um conjunto de pares chave/valor, nos quais cada chave individual tem um valor associado. A associação nos dicionários é feita por meio de uma chave (**key**) que faz referência a um valor.

```
1 dados_cliente = {  
2     'Nome': 'Renan',  
3     'Endereco': 'Rua Cruzeiro do Sul',  
4     'Telefone': '982503645'  
5 }  
6  
7 print(dados_cliente['Nome']) # Renan
```

- O Tamanho de um dicionário é definido pela quantidade de chaves (no exemplo acima: `len(dados_cliente) = 3`)
- Os dicionários também são **mutáveis**, podendo ser alteradas depois de terem sido criadas.



Dicionários

- Se quiser saber os valores de um dicionário, pode usar o comando **values()**. Para saber as chaves, pode usar o **keys()**. Por fim, tem também o **items()**.
- Sabendo que você tem chaves (keys), valores (values) e itens (items), você pode varrer um for com 2 parâmetros, usando algo como:

```
for k,v in biblioteca.items():  
    print(k,":",v)
```

ou

```
for i in biblioteca:  
    print(i,":",biblioteca[i])
```

O i aqui é a chave (key)

- Os 2 for acima apresentam a mesma saída.



Dicionários

- Para adicionar elementos em um dicionário basta associar uma nova chave ao objeto e dar um valor a ser associado a ela. Para inserir uma nova informação **Idade** em **dados_cliente**.

```
1 dados_cliente = {  
2     'Nome': 'Renan',  
3     'Endereco': 'Rua Cruzeiro do Sul',  
4     'Telefone': '982503645'  
5 }  
6  
7 print(dados_cliente) # {'Nome': 'Renan', 'Endereco': 'Rua Cruzeiro do Sul',  
8     'Telefone': '982503645'}  
9  
10 dados_cliente['Idade'] = 40  
11  
12 print(dados_cliente) # {'Nome': 'Renan', 'Endereco': 'Rua Cruzeiro do Sul',  
13     'Telefone': '982503645', 'Idade': 40}
```



Dicionários

- Remover chaves de um dicionário, sempre informe as **chaves** (não os valores).
`dados_cliente.pop` (“Telefone”) adicionando um **None** depois da chave, evita erro de busca.

`del dados_cliente[“Telefone”]`

```
1 dados_cliente = {
2     'Nome': 'Renan',
3     'Endereco': 'Rua Cruzeiro do Sul',
4     'Telefone': '982503645'
5 }
6
7 print(dados_cliente) # {'Nome': 'Renan', 'Endereco': 'Rua Cruzeiro do Sul',
8     'Telefone': '982503645'}
9
10 dados_cliente.pop('Telefone', None) ou del dados_cliente['Telefone']
11
12 print(dados_cliente) # {'Nome': 'Renan', 'Endereco': 'Rua Cruzeiro do Sul'}
```



Dicionários

- Voltando um pouco para listas, assim como é possível criar listas compostas, isso é, uma lista com outra lista, podemos também criar uma lista de bibliotecas.

- Ex:

```
cliente1 = {  
    "Nome": "Marcelo",  
    "Sobrenome": "Grilo",  
    "Idade": 30,  
}  
cliente2 = {  
    "Nome": "Pedro",  
    "Sobrenome": "Lucas",  
    "Idade": 24,  
}
```

```
clientes = [cliente1, cliente2]
```

Se `print(clientes[1][0]) => ??? --- ERROR`. O correto é: `print(clientes[1]["Nome"])`

Pedro

| Chave: Valor | Chave: Valor |
|------------------|------------------|
| Nome: Marcelo | Nome: Pedro |
| Sobrenome: Grilo | Sobrenome: Lucas |
| Idade: 30 | Idade: 24 |

0

1



Dicionários - Prática

- Crie um programa que leia o nome de um aluno e sua média, armazenando esses valores em um dicionário. A partir da nota desse aluno, o seu dicionário irá adicionar uma nova chave chamada “Situação”, que vai depender do valor da média, conforme intervalos abaixo:
 - Média ≥ 6 : **Situação: Aprovado**
 - Média ≥ 4 e < 6 : **Situação: Recuperação**
 - Média < 4 : **Situação: Reprovado**
- No final você deve mostrar as informações desse aluno por um **for**, algo parecido com:

Nome : Lucas

Media : 6.0

Situação : Aprovado










```
1 aluno = {}
2 aluno["Nome"] = input("Nome:\n")
3 print("A média de",aluno["Nome"],"foi de:")
4 aluno["Media"] = float(input())
5 if (aluno["Media"] >= 6):
6     aluno["Situação"]="Aprovado"
7 elif (aluno["Media"] >= 4 and aluno["Media"] < 6):
8     aluno["Situação"]="Recuperação"
9 else:
10     aluno["Situação"]="Reprovado"
11 print("=*"*25)
12 for i in aluno:
13     print(i,":",aluno[i])
```




- REVISANDO

Structures

@mlwithpython

| | Indexing | Ordered | Mutable | Duplicate |
|---|---|---|---|---|
|  List |  |  |  |  |
|  Tuple |  |  |  |  |
|  Set |  |  |  |  |
|  Dictionary |  |  |  |  |

@mlwithpython