Programação e Análise de Dados com Python

Programa de Pós-Graduação em Economia - PPGE

Python - Dicionários e aplicações

Prof. Hilton Ramalho

Prof. Aléssio Almeida

Objetivo

Apresentar noções gerais de operações com dicionários no `Python`.

Conteúdo

Dicionários

- Dicionários são objetos que representam uma coleção de elementos, onde cada elemento (valor) é associado a uma única chave.
- Nos dicionários **chaves** são associadas a valores ou a outros objetos (como outros dicionários ou listas).
- Um dicionário é um **objeto mutável** e pode agrupar diferentes tipos de objetos.
- Sintaxe de um dicionário

```
{'chave': <valor>, 'chave': <valor>, ..., 'chave': <valor>}
```

- Para criar um criar um dicionário com `{ }` ou com a função `dict` passando pares de `dict(chave=valor, chave=valor, ..., chave=valor)
- `help(dict)`

Métodos aplicáveis a objetos dicionários

Método	Descrição
clear()	Removes all items from the dictionary.
copy()	Returns a shallow copy of the dictionary.
fromkeys(seq[, v])	Returns a new dictionary with keys from seq and value equal to v (defaults to None).

Método	Descrição
get(key[,d])	Returns the value of the key. If the key does not exist, returns d (defaults to None).
items()	Return a new object of the dictionary's items in (key, value) format.
keys()	Returns a new object of the dictionary's keys.
pop(key[,d])	Removes the item with the key and returns its value or d if key is not found. If d is not provided and the key is not found, it raises KeyError.
popitem()	Removes and returns an arbitrary item (key, value). Raises KeyError if the dictionary is empty.
setdefault(key[,d])	Returns the corresponding value if the key is in the dictionary. If not, inserts the key with a value of d and returns d (defaults to None).

In [ ]:
```
help(dict)
```

In [ ]:
```
w = {'quantidade': 345, 'valor': 200}
print(f"O objeto {w} é um dicionário {type(w)}")
```

```
O objeto {'quantidade': 345, 'valor': 200} é um dicionário <class 'dict'>
```

A função dict()

- A função função `dict()` pode ser usada para transformar outros objetos em um dicionários.
- Uso simples: `dict(chave=valor, chave=valor)`.

In [ ]:
```
# Usando a função dict e passando tuplas com (chave, valor)
w = dict(quantidade=345, valor=200)
print(f"O objeto {w} é um dicionário {type(w)}")
```

In [ ]:
```
# Tentando criar um dicionário com chave repetida - o valor da última chave
w = {'quantidade': 345, 'valor': 200, 'valor': 400}
print(w)
```

```
{'quantidade': 345, 'valor': 400}
```

Exemplo:

Imagine que estamos com a caderneta de notas de uma turma.

In [ ]:
```
# Chaves são únicas
notas = {'Ana': 10, 'Maria': 9, 'João': None, 'Pedro': 10}
print(f"O objeto {notas} é da classe {type(notas)}")
```

```
O objeto {'Ana': 10, 'Maria': 10, 'João': None} é da classe <class 'dict'>
```

Dicionários mistos:

- Dicionários podem agrupar diferentes tipos de objetos: textos, números, listas, tuplas,

In [ ]:
```
# Exemplo
w = {'lista': [1,2,3], 'tupla': (1,2), 'numero': 4, 'texto': 'Olá'}
print(f"O objeto {w} é da classe {type(w)}")
```

```
O objeto {'lista': [1, 2, 3], 'tupla': (1, 2), 'numero': 4, 'texto': 'Olá'}
é da classe <class 'dict'>
```

In [ ]:
```
# Exemplo - dicionário aninhado
w = {'d': {'a':1, 'b':2}, 'z': {'c':3, 'd':4, 'e':8}}
print(f"O objeto {w} é da classe {type(w)}")
```

```
O objeto {'d': {'a': 1, 'b': 2}, 'z': {'c': 3, 'd': 4, 'e': 8}} é da classe
<class 'dict'>
```

In [ ]:
```
# Exemplo - dicionário misto
notas = {'Ana': {'A1': 10, 'A2': 8, 'A3': 7}, 'Maria': [1,6,10], 'João': (N
print(f"O objeto {notas} é da classe {type(notas)}")
```

Acessando elementos de um dicionário

- Em um dicionário podemos acessar determinado objeto a partir da sua chave.

In [ ]:
```
notas = {'Ana': 10, 'Maria': 9, 'João': None, 'Maria': 10}
notas['Ana']
```

Out[ ]:
```
10
```

In [ ]:
```
# Diferente de listas não usamos indexador de posição
notas[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-53-e3c5c807db71> in <module>()
      1 # Diferente de listas não usamos indexador de posição
----> 2 notas[0]

KeyError: 0
```

In [ ]:
```
# Acessando um valor de dicionário aninhado
notas = {'Ana': {'A1': 10, 'A2': 8, 'A3': 7}, 'Maria': [1,6,10], 'João': (N
notas['Ana']['A2']
```

Out[ ]:
```
8
```

In [ ]:
```
# Acessando um valor de dicionário que contém uma lista
notas = {'Ana': {'A1': 10, 'A2': 8, 'A3': 7}, 'Maria': [1,6,10], 'João': (N
notas['Maria'][1]
```

Out[ ]:
```
6
```

Métodos para dicionários

Método .get

In [ ]:
```
notas = {'Ana': 10, 'Carla': 9, 'João': None, 'Maria': 10}
notas.get("Maria")
```

Out[ ]:
```
10
```

In [ ]:
```
pessoas = {'alunos': {'maria': 10, 'pedro': 20} , 'docentes': {'joão': 14,
pessoas.get("alunos")
```

Out[ ]:
```
{'maria': 10, 'pedro': 20}
```

In [ ]:
```
pessoas.get("alunos").get('pedro')
```

Out[ ]:
```
20
```

Método .pop

- O método pop permite remover elementos de um dicionário a partir de uma chave.

In [ ]:
```
notas = {'Ana': 10, 'Carla': 9, 'João': None, 'Maria': 10}
notas.pop("Ana")
print(notas)
```

```
{'Maria': 10, 'João': None}
```

In [ ]:
```
pessoas = {'alunos': {'maria': 10, 'pedro': 20}, 'docentes': {'joão': 14, '
pessoas.pop("alunos")
print(pessoas)
```

```
{'docentes': {'joão': 14, 'paulo': 8, 'mario': 12}}
```

In [ ]:
```
# Qual é o problema aqui?
pessoas = {'alunos': {'maria': 10, 'pedro': 20}, 'docentes': {'joão': 14, '
pessoas.pop("alunos").pop("maria")
print(pessoas)
```

```
{'docentes': {'joão': 14, 'paulo': 8, 'mario': 12}}
```

In [ ]:
```
# Remover corretamente elemento em dicionário aninhado
pessoas = {'alunos': {'maria': 10, 'pedro': 20}, 'docentes': {'joão': 14, '
pessoas.get("alunos").pop("maria")
print(pessoas)
```

```
{'alunos': {'pedro': 20}, 'docentes': {'joão': 14, 'paulo': 8, 'mario': 1
2}}
```

Método .update

- O método update permite atualizar elementos de um dicionário a partir de uma chave.

In [ ]:
```
notas = {'Ana': 10, 'Carla': 9, 'João': None, 'Maria': 10}
print(notas)
```

```
{'Ana': 10, 'Carla': 9, 'João': None, 'Maria': 10}
```

In [ ]:
```
notas.update({'Pedro':9.5, 'Sabrina': 7.8})
print(notas)
```

```
{'Ana': 10, 'Carla': 9, 'João': None, 'Maria': 10, 'Pedro': 9.5, 'Sabrina':
7.8}
```

In [ ]:
```
alunos = {'alunos': {'maria': {'faltas': [2,0,3,4], 'dias': [2,4,7]},
          'pedro': {'faltas': [0,0,0,4], 'dias': [1,3,2]}   }  }
print(alunos)
```

```
{'alunos': {'maria': {'faltas': [2, 0, 3, 4], 'dias': [2, 4, 7]}, 'pedro':
{'faltas': [0, 0, 0, 4], 'dias': [1, 3, 2]}}}
```

In [ ]:
```
alunos.get("alunos").update({'sabrina': {'faltas': [10,4,5,7], 'dias': [1,2
print(alunos)
```

```
{'alunos': {'maria': {'faltas': [2, 0, 3, 4], 'dias': [2, 4, 7]}, 'pedro':
{'faltas': [0, 0, 0, 4], 'dias': [1, 3, 2]}, 'sabrina': {'faltas': [10, 4,
5, 7], 'dias': [1, 2, 3, 4]}}}
```

Método .keys

- O método keys retorna todas chaves de um dicionário.

In [ ]:
```
notas = {'Ana': 10, 'Carla': 9, 'João': None, 'Maria': 10}
notas.keys()
```

Out[ ]:
```
dict_keys(['Ana', 'Carla', 'João', 'Maria'])
```

In [ ]:
```
notas = {'Ana': 10, 'Carla': 9, 'João': None, 'Maria': 10}
print("Qual é o tipo desse objeto? ",type(notas.keys()))
```

```
Qual é o tipo desse objeto?  <class 'dict_keys'>
```

In [ ]:
```
# Podemos transformá-lo em uma lista com as chaves usando a função list(.)
alunos = list(notas.keys())
print(f"As chaves agora foram imputadas na lista {alunos} do tipo {type(alu
```

```
As chaves agora foram imputadas na lista ['Ana', 'Carla', 'João', 'Maria']
do tipo <class 'list'>
```

In [ ]:
```
pessoas = {'alunos': {'maria': {'faltas': [2,0,3,4], 'dias': [2,4,7]},
          'pedro': {'faltas': [0,0,0,4], 'dias': [1,3,2]}}}
pessoas.keys()
```

Out[ ]:
```
dict_keys(['alunos'])
```

In [ ]:
```
# Acessando chaves de dicionário aninhado
pessoas.get("alunos").keys()
```

Out[ ]:
```
dict_keys(['maria', 'pedro'])
```

In [ ]:
```
# Acessando chaves de dicionário aninhado
pessoas.get("alunos").get("maria").keys()
```

Out[ ]:
```
dict_keys(['faltas', 'dias'])
```

In [ ]:
```
# Passando as chaves para uma lista
lista = list(pessoas.get("alunos").get("maria").keys())
print(lista)
```

```
['faltas', 'dias']
```

Método .values

- O método values retorna todos elementos de um dicionário.

In [ ]:
```
notas = {'Ana': 10, 'Maria': 9, 'João': None, 'Maria': 10}
notas.values()
```

Out[ ]:
```
dict_values([10, 10, None])
```

In [ ]:
```
notas = {'Ana': 10, 'Carla': 9, 'João': None, 'Maria': 10}
print("Qual é o tipo desse objeto? ",type(notas.values()))
```

```
Qual é o tipo desse objeto?  <class 'dict_values'>
```

In [ ]:
```
# Podemos transformá-lo em uma lista de elementos usando a função list(.)
notas = list(notas.values())
print(f"As notas agora foram imputadas na lista {notas} do tipo {type(notas
```

```
As notas agora foram imputadas na lista [10, 9, None, 10] do tipo <class 'l
ist'>
```

In [ ]:
```
# Dicionário aninhado
pessoas = {'alunos': {'maria': {'faltas': [2,0,3,4], 'dias': [2,4,7]},
          'pedro': {'faltas': [0,0,0,4], 'dias': [1,3,2]}}}
pessoas.values()
```

Out[ ]:
```
dict_values([{'maria': {'faltas': [2, 0, 3, 4], 'dias': [2, 4, 7]}, 'pedro
': {'faltas': [0, 0, 0, 4], 'dias': [1, 3, 2]}}])
```

In [ ]:
```
# Dicionário aninhado
pessoas = {'alunos': {'maria': {'faltas': [2,0,3,4], 'dias': [2,4,7]},
          'pedro': {'faltas': [0,0,0,4], 'dias': [1,3,2]}}}
pessoas.get('alunos').values()
```

Out[ ]:
```
dict_values([{'faltas': [2, 0, 3, 4], 'dias': [2, 4, 7]}, {'faltas': [0, 0,
0, 4], 'dias': [1, 3, 2]}])
```

In [ ]:
```
# Dicionário aninhado
pessoas = {'alunos': {'maria': {'faltas': [2,0,3,4], 'dias': [2,4,7]},
          'pedro': {'faltas': [0,0,0,4], 'dias': [1,3,2]}}}
pessoas.get('alunos').get('maria').values()
```

Out[ ]: dict_values([[2, 0, 3, 4], [2, 4, 7]])