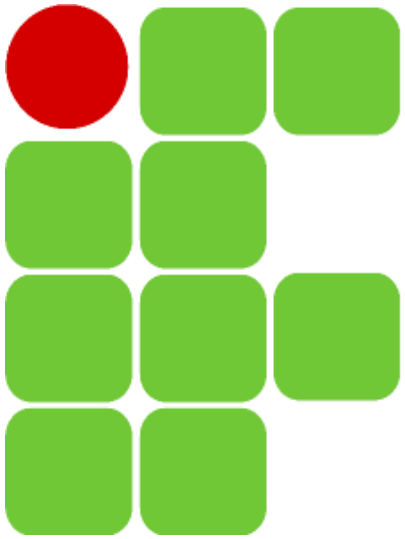

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
BAHIA
Campus Salvador

Disciplina : Programação Python

Prof. Domingos Mainart

Email: prof.mainart@gmail.com

Dicionários

Dicionário é uma coleção de itens (chamados chaves) e seus respectivos significados (chamados de valores): {chave: valor}

- **Cada chave do dicionário deve ser única! Ao contrário de listas, dicionários, não podem ter chaves repetidas.**
- **Declaração**
  - **Declaramos um dicionário colocando entre colchetes {} cada chave e o seu respectivo valor, da seguinte forma:**

```
>>> telefones = {"ana": 123456, "yudi": 40028922, "julia": 4124492}
>>> telefones
{'ana': 123456, 'yudi': 40028922, 'julia': 4124492}
```

No caso acima, a chave "ana", por exemplo, está relacionada ao valor 123456. Cada par chave-valor é separado

por uma vírgula , .

Dicionários

Função dict()

A função dict() constrói um dicionário. Existem algumas formas de usá-la:

- Com uma lista de listas:

```
>>> lista1 = ["brigadeiro", "leite condesado, achocolatado"]
>>> lista2 = ["omelete", "ovos, azeite, condimentos a gosto"]
>>> lista3 = ["ovo frito", "ovo, óleo, condimentos a gosto"]
>>> lista_receitas = [lista1, lista2, lista3]
```

```
>>> print(lista_receitas)
[['brigadeiro', 'leite condesado, achocolatado'], ['omelete', 'ovos, azeite,␣
↪condimentos a gosto'], ['ovo frito', 'ovo, óleo, condimentos a gosto']]
>>> receitas = dict(lista_receitas)
>>> print(receitas)
{'brigadeiro': 'leite condesado, achocolatado', 'omelete': 'ovos, azeite,␣
↪condimentos a gosto', 'ovo frito': 'ovo, óleo, condimentos a gosto'}
```

Dicionários

- Atribuindo os valores diretamente:

```
>>> constantes = dict(pi=3.14, e=2.7, alpha=1/137)
>>> print(constantes)
{'pi': 3.14, 'e': 2.7, 'alpha': 0.007299270072992701}
```

Neste caso, o nome das chaves deve ser um identificador válido. As mesmas regras de *nomes de variáveis* (Página 41) se aplicam.

- Usando as chaves `{}`:

```
>>> numerinhos = dict({"um": 1, "dois": 2, "três": 3})
>>> print(numerinhos)
{'um': 1, 'dois': 2, 'três': 3}
```

E nesse caso se não houvesse a função *dict()*, o resultado seria exatamente o mesmo...

Dicionários

- Chaves
  - Acessamos um determinado valor do dicionário através de sua chave:

```
>>> capitais = {"SP": "São Paulo", "AC": "Rio Branco", "TO": "Palmas", "RJ": "Rio de␣
↪Janeiro", "SE": "Aracaju", "MG": "Belo Horizonte"}
>>> capitais["MG"]
'Belo Horizonte'
```

Até o momento, usamos apenas *strings*, mas podemos colocar todo tipo de coisa dentro dos dicionários, incluindo listas e até mesmo outros dicionários:

```
>>> numeros = {"primos": [2, 3, 5], "pares": [0, 2, 4], "ímpares": [1, 3, 5]}
>>> numeros["ímpares"]
[1, 3, 5]
```

Mesmo que os pares chave-valor estejam organizados na ordem que foram colocados, não podemos acessá-los por *índices* como faríamos em listas:

```
>>> numeros[2]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 2
```

Dicionários

Podemos alterar o valor relacionado a uma chave da seguinte forma:

```
>>> pessoa = {"nome": "Cleiton", "idade": 34, "família": {"mãe": "Maria", "pai": "Enzo
↪"}}
>>> pessoa["idade"]
34
>>> pessoa["idade"] = 35
>>> pessoa["idade"]
35
```

Para adicionar um elemento novo à um dicionário, podemos simplesmente fazer o seguinte:

```
>>> meses = {1: "Janeiro", 2: "Fevereiro", 3: "Março"}
>>> meses[4] = "Abril"
>>> meses
{1: "Janeiro", 2: "Fevereiro", 3: "Março", 4: "Abril"}
```

Dicionários

Removemos um conjunto chave-elemento de um dicionário com o comando `del`:

```
>>> meses
{1: "Janeiro", 2: "Fevereiro", 3: "Março", 4: "Abril"}
>>> del(meses[4])
>>> meses
{1: "Janeiro", 2: "Fevereiro", 3: "Março"}
```

Para apagar *todos* os elementos de um dicionário, usamos o método `clear`:

```
>>> lixo = {"plástico": ["garrafa", "copinho", "canudo"], "papel": ["folha amassada",
↪"guardanapo"], "orgânico": ["batata", "resto do bandeco", "casca de banana"]}
>>> lixo
{"plástico": ["garrafa", "copinho", "canudo"], "papel": ["folha amassada", "guardanapo
↪"], "organico": ["batata", "resto do bandeco", "casca de banana"]}
>>> lixo.clear()
>>> lixo
{}
```

Dicionários

- Função list()

A função `list()` recebe um conjunto de objetos e retorna uma lista. Ao passar um dicionário, ela retorna uma lista contendo todas as suas *chaves*:

```
>>> institutos_uspsc = {"IFSC": "Instituto de Física de São Carlos", "ICMC":
↪"Instituto de Ciências Matemáticas e de Computação", "EESC": "Escola de Engenharia␣
↪de São Carlos", "IAU": "Instituto de Arquitetura e Urbanismo", "IQSC": "Instituto␣
↪de Química de São Carlos"}
>>> list(institutos_uspsc)
['IQSC', 'IFSC', 'ICMC', 'IAU', 'EESC']
```

Dicionários

- Função len()

A função `len()` retorna o número de elementos («tamanho») do objeto passado para ela. No caso de uma lista, fala quantos elementos há. No caso de dicionários, retorna o *número de chaves* contidas nele:

```
>>> institutos_uspsc
{'IQSC': 'Instituto de Química de São Carlos', 'IFSC': 'Instituto de Física de São
↪Carlos', 'ICMC': 'Instituto de Ciências Matemáticas e de Computação', 'IAU':
↪'Instituto de Arquitetura e Urbanismo', 'EESC': 'Escola de Engenharia de São Carlos
↪'}
>>> len(institutos_uspsc)
5
```

Você pode contar o número de elementos na lista gerada pela função `list()` para conferir:

```
>>> len(list(institutos_uspsc))
5
```

Dicionários

- Método get()
  - O método get(chave, valor) pode ser usado para retornar o valor associado à respectiva chave! O segundo parâmetro <valor> é opcional e indica o que será retornado caso a chave desejada não esteja no dicionário:

```
>>> institutos_uspsc.get("IFSC")
'Instituto de Física de São Carlos'
```

Dá para ver que ele é muito parecido com fazer assim:

```
>>> institutos_uspsc["IFSC"]
'Instituto de Física de São Carlos'
```

Dicionários

- Método get()
  - O método get(chave, valor) pode ser usado para retornar o valor associado à respectiva chave! O segundo parâmetro <valor> é opcional e indica o que será retornado caso a chave desejada não esteja no dicionário:

```
>>> institutos_uspsc.get("IFSC")
'Instituto de Física de São Carlos'
```

Dá para ver que ele é muito parecido com fazer assim:

```
>>> institutos_uspsc["IFSC"]
'Instituto de Física de São Carlos'
```

Dicionários

- Alguns métodos

- O método `items()` pode ser comparado com o *inverso* da função `dict()`:

```
>>> pessoa = {"nome": "Enzo", "RA": 242334, "curso": "fiscomp"}
>>> pessoa.items()
dict_items([('curso', 'fiscomp'), ('nome', 'Enzo'), ('RA', 242334)])
```

Usando a função `list()` nesse resultado, obtemos:

```
  >>> pessoa.items()
  dict_items([('curso', 'fiscomp'), ('nome', 'Enzo'), ('RA', 242334)])
  >>> itens = list(pessoa.items())
  >>> itens
  [('curso', 'fiscomp'), ('nome', 'Enzo'), ('RA', 242334)]

Experimente usar a função ``dict()`` na lista ``itens``!
```

- O método `values()` nos retorna os *valores* do dicionário:

```
>>> pessoa.values()
dict_values(['fiscomp', 'Enzo', 242334])
>>> valores = list(pessoa.values())
>>> valores
['fiscomp', 'Enzo', 242334]
```

Dicionários

- Ordem dos elementos

Dicionários não tem sequência dos seus elementos. As listas têm. Dicionários mapeiam um valor a uma chave. Veja este exemplo:

```
>>> numerinhos = dict({"um": 1, "dois": 2, "três": 3})
>>> numeritos = {"três": 3, "dois": 2, "um": 1}
>>> numerinhos == numeritos
True
>>> numeritos
{'três': 3, 'dois': 2, 'um': 1}
>>> numerinhos
{'um': 1, 'dois': 2, 'três': 3}
```

Vemos que `numerinhos` e `numeritos` têm as mesmas chaves com os mesmos valores e por isso são iguais. Mas quando imprimimos cada um, a ordem que aparece é a que os itens foram inseridos.

Dicionários

- **Está no dicionário?**

Podemos checar se uma chave está ou não em um dicionário utilizando o comando `in`. Voltando para o dicionário que contem os institutos da USP São Carlos:

```
>>> institutos_uspsc
{'IQSC': 'Instituto de Química de São Carlos', 'IFSC': 'Instituto de Física de São
↪Carlos', 'ICMC': 'Instituto de Ciências Matemáticas e de Computação', 'IAU':
↪'Instituto de Arquitetura e Urbanismo', 'EESC': 'Escola de Engenharia de São Carlos
↪'}
>>> "IFSC" in institutos_uspsc
True
>>> "ESALQ" in institutos_uspsc
False
```

E checamos se uma chave *não está* no dicionário com o comando `not in`:

```
>>> institutos_uspsc
{'IQSC': 'Instituto de Química de São Carlos', 'IFSC': 'Instituto de Física de São
↪Carlos', 'ICMC': 'Instituto de Ciências Matemáticas e de Computação', 'IAU':
↪'Instituto de Arquitetura e Urbanismo', 'EESC': 'Escola de Engenharia de São Carlos
↪'}
>>> "IFSC" not in institutos_uspsc
False
>>> "ESALQ" not in institutos_uspsc
True
```