SCC 124 - Introdução à Programação para Engenharias

Outras estruturas de dados

Professor: André C. P. L. F. de Carvalho, ICMC-USP
Pos-doutorando: Isvani Frias-Blanco
Monitor: Henrique Bonini de Britto Menezes

Aula de hoje

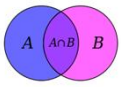
- Introdução
- Conjuntos
  - Métodos
  - Operadores
- Dicionários
  - Métodos
  - Operadores
- Exemplos

Teoria dos conjuntos

- Área de lógica matemática que estuda conjuntos (coleção de objetos)
- Área começou em 1874 após publicação do artigo
  - *On a Property of the Collection of All Real Algebraic Numbers*, por Georg Cantor
- Define operações binárias sobre conjuntos

Teoria dos conjuntos

- Principais operações
  - Pertinência
  - União
  - Interseção
  - Diferença
  - Complemento
  - Leis de De Morgan



Conjuntos em Python

- Coleção não ordenada de elementos sem repetição
  - Existência de objetos é mais importante que ordem e quantas vezes eles ocorrem
  - Elementos são listados entre {...}
    - Comando para definir conjunto vazio: *set()*

Conjuntos em Python

- Tipo mutável
- Principais usos:
  - Testar pertinência de itens
  - Eliminar itens duplicados
  - Manipular conjuntos de itens

Conjuntos em Python

- Operações de teoria dos conjunto podem ser aplicados a um conjunto de itens
  - Pertinência (in)
  - União (|)
  - Interseção (&)
  - Diferença (-)
  - A > b
  - ...

Operadores Relacionais e bitwise

Exemplo

```
>>> cesta = ['maçã', 'laranja', 'maçã', 'pera', 'laranja', 'uva']
>>> frutas = set(cesta)          # cria conjunto sem repetições
>>> frutas

>>> 'laranja' in frutas          # Teste rápido de pertinência

>>> 'pinha' in frutas
```


Exemplo

```
>>> cesta = ['maçã', 'laranja', 'maçã', 'pera', 'laranja', 'uva']
>>> frutas = set(cesta)          # cria conjunto sem repetições
>>> frutas
{'maçã', laranja', 'pera', 'uva'}
>>> 'laranja' in frutas          # Teste rápido de pertinência
True
>>> 'pinha' in frutas
False
```


Conjuntos em Python

- Elementos podem ser incluídos com:
  - Operador /=
  - Método *add()* # formato: *nomedoconjunto.add()*
- Outros métodos
  - Método *copy ()* (copia conjunto)
  - Método *remove()* (remove item de um conjunto)
  - Método *issuperset()* (define se um conjunto contem outro)

Exemplo 1

```
>>> a = (1, 4, "um", 4)
>>> conjunto = set(a)
>>> conjunto

>>> b = set()
>>> b

>>> b |= {2, 'la'}
>>> b
```

Pode receber qualquer sequência
Não inclui itens duplicados

Exemplo 1

```
>>> a = (1, 4, "um", 4)
>>> conjunto = set(a)
>>> conjunto
{1, 4, "um"}
>>> b = set()
>>> b
set()
>>> b |= {2, 'la'}
>>> b
{2, 'la'}
```

Pode receber qualquer sequência
Não inclui itens duplicados

Exemplo 2

```
>>> a = {1, 4, "um"}
>>> b = a.copy()
>>> a.add(0)
>>> a

>>> a.issuperset(b)

>>> b.remove(1)
>>> b
```


Exemplo 2

```
>>> a = {1, 4, "um"}
>>> b = a.copy() # equivalente: b = a
>>> a.add(0)
>>> a
{0, 1, 4, 'um'}
>>> a.issuperset(b)
True
>>> b.remove(1)
>>> b
{4, 'um'}
```


Exemplo 3

```
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                          # letras que aparecem em a

>>> a - b                      # letras em a mas nao em b

>>> a | b                      # letras em a ou em b

>>> a & b                      # letras em a e em b

>>> a ^ b                      # letters exclusivamente em a ou b
```


Exemplo 3

```
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                          # letras que aparecem em a
{'a', 'r', 'b', 'c', 'd'}
>>> a - b                      # letras em a mas nao em b
{'r', 'd', 'b'}
>>> a | b                      # letras em a ou em b
{'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}
>>> a & b                      # letras em a e em b
{'a', 'c'}
>>> a ^ b                      # letters exclusivamente em a ou b
{'r', 'd', 'b', 'm', 'z', 'l'}
```


Exercício

```
>>> a  = {1, 3, 5, 6}
>>> b = {2, 4, 6}
>>> a | b

>>> a & b

>>> a – b

>>> a ^ b
```


Exercício

```
>>> a  = {1, 3, 5, 6}
>>> b = {2, 4, 6}
>>> a | b
{1, 2, 3, 4, 5, 6}
>>> a & b
{6}
>>> a – b
{1, 3, 5}
>>> a ^ b
{1, 2, 3, 4, 5}
```

Comando for

- Conjuntos podem ser usados em comandos for

```
>>> a = {x for x in 'abracadabra' if x not in 'abc'}
>>> a
```


Comando for

- Conjuntos podem ser usados em comandos for

```
>>> a = {x for x in 'abracadabra' if x not in 'abc'}
>>> a
{'r', 'd'}
```


Exercício 1

- Escrever um programa em Python para provar a lei de De Morgan

Exercício 1

- Escrever um programa em Python para provar a lei de distributividade entre conjuntos

Dicionário

- Semelhante a um dicionário onde você pode:
  - Consultar palavras e significados
  - Acrescentar palavras e significados
  - Alterar significados de palavras
  - Excluir palavras e parte de seus significados

Dicionário

- Também pode ser usado como uma caderneta de contatos, onde você pode:
  - Anotar telefone, *email* e endereço
  - Encontra *email* procurando pelo nome
- Cada item (entrada) é formado pelo par:
  - Valor: um conjunto de valores (informações)
  - Chave: uma palavra-chave (ex.: nome completo), que serve de indexador (identificação) de item

Dicionário

- Chave pode ser de qualquer tipo imutável
  - Strings e números podem sempre ser chaves
  - Tuplas podem ser chaves se contêm apenas strings, números ou tuplas
  - Listas não podem ser chaves (tipo mutável)
- Valor pode ser de qualquer tipo, mutável ou imutável

Tipos mutáveis x imutáveis

```
>>> b = [12, 34, 3]
>>> b

>>> b[2] = 6
>>> b

>>> b[1] = 2
>>> b
```

```
>>> x = "aulas"
>>> x[1]

>>> x[0] = 'z'  # ERRO
```


Tipos mutáveis x imutáveis

```
>>> b = [12, 34, 3]
>>> b
[12, 34, 3]
>>> b[2] = 6
>>> b
[12, 34, 6]
>>> b[1] = 2
>>> b
[12, 2, 6]
```

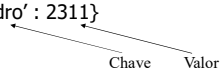
```
>>> x = "aulas"
>>> x[1]
'u'
>>> x[0] = 'z'  # ERRO

ERRO, pois valores
de um string não
podem ser mudados
```


Dicionário

- A chave de cada item é única
  - Mais de um item não pode ter a mesma chave
    - Ex.: email, nome completo, CPF, ...
  - Chaves devem ser objetos simples
- Em Python, dicionários são instâncias (objetos) da classe *dict*

Dicionários

- Podem ser vistos como um conjuntos não ordenados de pares chave-valor
  - Ex.: {'Joao' : 3245, 'Pedro' : 2311}
  - Dicionário vazio: {}



- Chaves devem ser únicas
  - Armazenar novo item com chave existente escreve sobre o item que tem a mesma chave
- Acessar item com chave que não existe gera erro

Dicionários

- Itens podem ser
  - Consultados
    - Indexar dicionário com a chave do item a ser consultado
  - Incluídos
    - Atribuir valor, indexando dicionário com uma nova chave
  - Excluídos
    - Elimina itens, utilizar para isso o comando *del*
  - Alterados
    - Dada uma chave existente, alterar valor do seu item

Exemplo

Incluir novo item

Consultar dicionário

Eliminar item

```
>>> tel = {'joao': 4098, 'leo': 4139}
>>> tel['luis'] = 4127
>>> tel

>>> tel['joao']

>>> del tel['leo']
>>> tel['ivo'] = 4127
>>> tel
```


Exemplo

Incluir novo item

Consultar dicionário

Eliminar item

```
>>> tel = {'joao': 4098, 'leo': 4139}
>>> tel['luis'] = 4127
>>> tel
{'leo': 4139, 'luis': 4127, 'joao': 4098}
>>> tel['joao']
4098
>>> del tel['leo']
>>> tel['ivo'] = 4127
>>> tel
{'luis': 4127, 'ivo': 4127, 'joao': 4098}
```


Métodos pré-definidos

- Método *nomedicionario.items()*
  - Retorna os pares chave-valor de um dicionário
  - Visão de itens
- Método *nomedicionario.keys()*
  - Retorna as chaves do dicionário (em ordem arbitrária)
  - Podem ser ordenadas pelo Método *sorted()*
- Método *nomedicionario.values()*
  - Retorna os valores do dicionário

Visões dos métodos pré-definidos

- Permitem transformar dicionário em uma lista de:
  - Itens
    - Método *nomedicionario.items()*
  - Chaves
    - Método *nomedicionario.keys()*
  - Valores
    - Método *nomedicionario.values()*

Exemplo 1

```
>>> tel

>>> tel.keys()

>>> list(tel.keys())

>>> tel.values()

>>> tel.items()

>>> list (tel.items()

>>> sorted(tel)
```


Exemplo 1

Visão de itens

Retorna chaves ordenadas

```
>>> tel
{'luis': 4127, 'ivo': 4127, 'joao': 4098}
>>> tel.keys()
dict_keys(['Pedro', 'Ivo', 'Joao'])
>>> list(tel.keys())
['luis', 'ivo', 'joao']
>>> tel.values()
dict_values([4127, 4127, 4098])
>>> tel.items()
dict_items([('luis': 4127), ('ivo': 4127), ('joao': 4098)])
>>> list (tel.items()
[('luis': 4127), ('ivo': 4127), ('joao': 4098)]
>>> sorted(tel)
['ivo', 'joao', 'luis']
```

Exemplo 2

```
>>> tel = {'joao': 4098, 'leo': 4139}
>>> oldtel = tel
>>> oldtel

>>> novotel = tel.items()
>>> novotel

>>> sorted(novotel)
```


Exemplo 2

```
>>> tel = {'joao': 4098, 'leo': 4139}
>>> oldtel = tel
>>> oldtel
{'joao': 4098, 'leo': 4139}
>>> novotel = tel.items()
>>> novotel
dict_items ([('leo', 4139), ('joao', 4098)])
>>> sorted(novotel)
[('joao', 4098), ('leo', 4139)]
```

Retorna itens ordenados por chave

Construtores e operadores

- Construtor *dict()*
  - Constrói dicionários diretamente de uma lista de pares (tuplas) chave-valor
- Operador *in*
  - Retorna um valor booleano que indica se uma chave está presente em um dicionário

Exemplo

```
>>> tel2 = {'joao': 4098, 'leo': 4139}
>>> dict([('joao', 4098), ('leo', 4139)])

>>> 'joao' in tel2

>>> 'marcos' not in tel2
```


Exemplo

```
>>> tel2 = {'joao': 4098, 'leo': 4139}
>>> dict([('joao', 4098), ('leo', 4139)])
{'joao': 4098, 'leo': 4139}
>>> 'joao' in tel2
True
>>> 'marcos' not in tel2
True
```


Exercício 2

- Criar um dicionário com o nome e a nota de cinco disciplinas que você cursou nos últimos 12 meses
  - Ordenar as disciplinas por ordem crescente de notas e, para notas iguais, ordem alfabética de nome

Conclusão

- Outras estruturas de dados
- Conjuntos
  - Métodos
  - Operadores
- Dicionários
  - Métodos
  - Operadores
- Exemplos

Perguntas