Python:
Conjuntos

Douglas Duarte

Conjuntos

- Um conjunto é uma coleção de valores distintos
- Pode-se implementar conjuntos de diversas formas
  - Uma lista de valores
    - Têm-se que tomar o cuidado de evitar valores duplicados
  - Um dicionário
    - As chaves de um dicionário são necessariamente únicas
    - O valor associado a cada chave pode ser qualquer um
- Python suporta um tipo primitivo chamado **set** que implementa conjuntos
  - Mais apropriado do que o uso de listas ou dicionários

O tipo *set*

- Pode-se construir um set usando a construção

  set(*sequência*)

  - Onde *sequência* é uma sequência qualquer, como uma lista, uma tupla ou uma string
  - Caso use-se uma lista, os elementos devem ser imutáveis
- Exemplos:

```
>>> set((1,2,3))
set([1, 2, 3])
>>> set ("xxabc")
set(['a', 'x', 'c', 'b'])
>>> set ([1,(1,2),3,1])
set([(1, 2), 1, 3])
>>> set ([1,[1,2],3,1])
ERROR...
```

Trabalhando com sets

- *x* in *s* → True se o elemento *x* pertence a *s*
- *s*.add(*x*) → Inclui o elemento *x* em *s*
- *s*.copy() → Retorna uma cópia de *s*
- *s*.union(*r*) → Retorna a união entre *s* e *r*
- *s*.intersection(*r*) → Retorna a interseção entre *s* e *r*
- *s*.difference(*r*) → Retorna a diferença entre *s* e *r*
- list*(s)* → Retorna os elementos de *s* numa lista
- tuple*(s)* → Retorna os elementos de *s* numa tupla

Exemplos

```
>>> s = set([1,2,3])
>>> r = set([2,5,9,1])
>>> 1 in s
True
>>> 1 in r
True
>>> s.union(r)
set([1, 2, 3, 5, 9])
>>> s.intersection(r)
set([1, 2])
>>> s.difference(r)
set([3])
>>> r.difference(s)
set([9, 5])
>>> s.add(5)
>>> s.intersection(r)
set([1, 2, 5])
```

Iterando sobre sets

- Pode-se também usar o comando for com sets
- Observe-se que a iteração não necessariamente visita os elementos na mesma ordem em que eles foram inseridos no conjunto
- Exemplo:

```
>>> s = set([1,2,9,100,"a"])
>>> for x in s:
                print x,


a 1 2 100 9
```

Outros métodos

- *s*.discard(*x*) $\rightarrow$ Exclui o elemento *x* de *s* (se existir)
- *s*.issubset(*r*) $\rightarrow$ True sse *s* contido em *r*
- *s*.issuperset(*r*) $\rightarrow$ True sse *s* contém *r*
- *s*.symmetric_difference(*r*) $\rightarrow$ Retorna a diferença simétrica entre *s* e *r*, isto é, a união entre *s* e *r* menos a interseção de *s* e *r*
- *s*.update(*r*) $\rightarrow$ mesmo que *s* = *s*.union(*r*)
- *s*.intersection_update(*r*) $\rightarrow$ mesmo que *s* = *s*.intersection(*r*)
- *s*.difference_update(*r*) $\rightarrow$ mesmo que *s* = *s*.difference(*r*)

Exemplos

```
>>> s = set([1,2,3])
>>> r = set([2,5,9])
>>> s.update(r)
>>> s
set([1, 2, 3, 5, 9])
>>> s.issuperset(r)
True
>>> r.issubset(s)
True
>>> s.discard(5)
>>> s
set([1, 2, 3, 9])
>>> s.symmetric_difference(r)
set([3, 5, 1])
```