Tópicos avançados em Informática I
Programação Python

Estruturas de dados compostas
- Parte 2 -

Profa. Ana Paula Canal
Universidade Franciscana

Sumário

- Tuplas
- Conjuntos
- Dicionários

Tuplas (*tuple*)

- São estruturas de dados compostas que armazenam itens de qualquer tipo
- São imutáveis, isto é, não são permitidas alterações nos elementos de uma tupla, inclusive o ordenamento dos elementos não pode ser realizado
- Armazenamento de dados heterogêneos, sendo possível armazenar mais de um dado em campos diferentes, permitindo o agrupamento de informações que estão relacionadas
- É representada por uma sequência de itens separados por vírgula. O uso de parênteses na definição da tupla não é obrigatório, mas fortemente recomendado

Tuplas

- Se o item for definido entre apóstrofes o Python vai interpretar a atribuição como uma *string*, para este item fazer parte de uma tupla ele deve vir precedido de uma vírgula. A vírgula é o caractere que constrói a tupla

- O acesso aos elementos de uma tupla é realizado com a utilização de índices entre colchetes

- O recurso de *slicing* ou fatiamento também pode ser aplicado nas tuplas

Tuplas

```
>>> tupla = 'João'
>>> type(tupla)
<class 'str'>
>>> tupla = 'João',
>>> type(tupla)
<class 'tuple'>
>>> tupla = ('João','Santos','da Silva')
>>> tupla
('João', 'Santos', 'da Silva')
>>> carro = ('Chevrolet', 'S10', 'Branca', 2017, 'IJR-2017')
>>> carro[1]
'S10'
>>> tupla[-1]
'da Silva'
>>> tupla[:]
('João', 'Santos', 'da Silva')
```

Tuplas

```
>>> carro = ('Chevrolet', 'S10', 'Branca', 2017, 'IJR-2017')
>>> 'S10' in carro
True
>>> for c in carro:
        print(c)


Chevrolet
S10
Branca
2017
IJR-2017
>>> carro[1] = 'Onix'
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    carro[1] = 'Onix'
TypeError: 'tuple' object does not support item assignment
>>> (marca, modelo, cor, ano, placa) = carro
>>> marca
'Chevrolet'
>>> modelo
'S10'
>>> ano
2017
>>> tupla = ('Lista',[1,2,3])
>>> tupla[1][1]=5
>>> tupla
('Lista', [1, 5, 3])
```

Tuplas

```
>>> tupla = (1,2,3)
>>> tupla += (9,)*3
>>> tupla
(1, 2, 3, 9, 9, 9)
>>> tupla2 = (8,7,6)
>>> tupla += tupla2
>>> tupla
(1, 2, 3, 9, 9, 9, 8, 7, 6)
>>> len(tupla)
9
>>> min(tupla)
1
>>> max(tupla)
9
>>> tupla.count(9)
3
>>> tupla.index(3)
2
```

Tuplas

- Tupla é definida da mesma forma que uma lista, o que a diferencia, são os caracteres delimitadores.

- Uma Lista, tem seus elementos delimitados por colchetes, enquanto que a Tupla, tem seus elementos delimitados por parênteses.

- A ordem dos elementos numa Tupla é imutável, assim, não é possível ordená-la em tempo de execução.

Conjuntos (*set*)

- Define um tipo de dados para conjuntos onde é permitido aplicar as operações matemáticas dos conjuntos: união, interseção, diferença e diferença simétrica

- Sequência desordenada de elementos, com a característica de não apresentar itens repetidos

- Pode ser definido pelo uso de *set*, com uma vírgula separando os elementos ou fazer uso do comando *set* seguido da sequência de elementos entre parênteses

- Uma tupla ou uma lista podem ser convertidas para um conjunto, nestes casos, deve-se usar o comando *set* e entre os parênteses informar o nome da lista ou tupla, nestes casos os elementos não ficarão ordenados

- É possível percorrer os elementos de um conjunto com o uso de um laço for e para verificar se um determinado elemento pertence a um conjunto faz-se uso do operador ***in***

Conjuntos

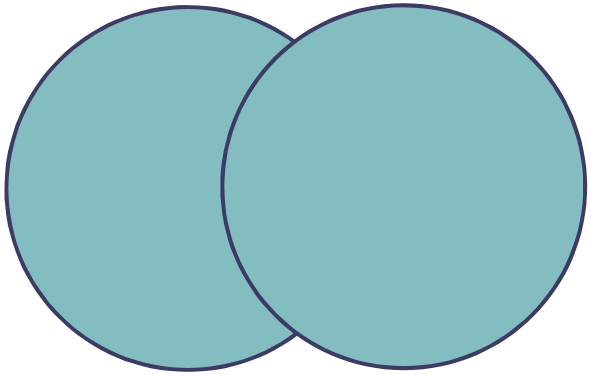
```
>>> set1 = {'Violão', 'Guitarra', 'Baixo', 'Guitarra'}
>>> set1
{'Baixo', 'Guitarra', 'Violão'}
>>> set2 = set(('Bateria','Acordeon','Piano'))
>>> set2
{'Acordeon', 'Piano', 'Bateria'}
>>> tupla = ('Surdo','Tarol','Pandeiro')
>>> set3 = set(tupla)
>>> set3
{'Pandeiro', 'Surdo', 'Tarol'}
>>> lista = ['Saxofone','Flauta','Trompete']
>>> set4 = set(lista)
>>> set4
{'Trompete', 'Saxofone', 'Flauta'}
>>> 'Baixo' in set1
True
>>> for instrumento in set1:
        print(instrumento)


Baixo
Guitarra
Violão
```

Conjuntos

Método	Descrição
add(elemento)	Adiciona o elemento no conjunto
clear()	Remove todos os elementos do conjunto
copy()	Retorna uma cópia do conjunto
difference(conjunto)	Retorna a diferença entre os conjuntos
discard(elemento) ou remove(elemento)	Remove o elemento do conjunto
intersection(conjunto)	Retorna o a interseção entre os conjuntos
isdisjoint(conjunto)	Verifica se a interseção é nula entre os conjuntos
issubset(conjunto)	Verifica se um conjunto está contido no outro
issuperset(conjunto)	Verifica se um conjunto contém o outro
pop()	Remove o primeiro elemento do conjunto, retornando este valor
symmetric_difference(conjunto)	Retorna a diferença simétrica entre os conjunto, isto é, a união entre os conjuntos menos a interseção entre eles
union(conjunto)	Retorna a união entre os conjuntos

Conjuntos: união

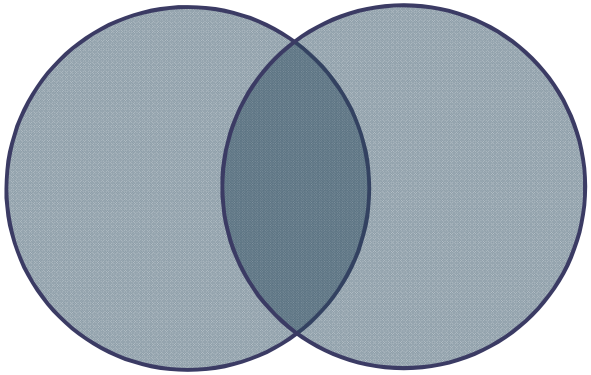


A U B

- Defina os conjuntos a e b e verifique a operação de união:

```
>>> a
{2, 3, 7, 10, 11}
>>> b
{2, 4, 6, 10, 11}
>>> a.union(b)
{2, 3, 4, 6, 7, 10, 11}
>>>
```

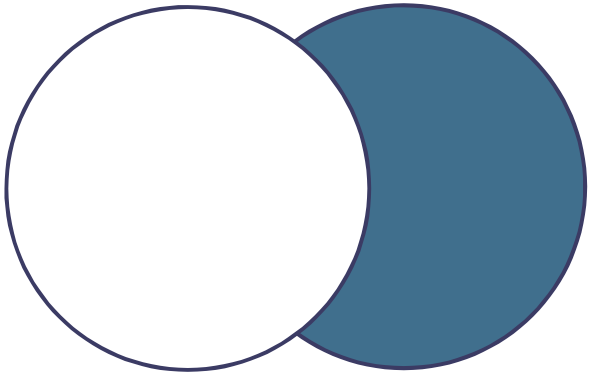
Conjuntos: interseção



A ∩ B

```
>>> a.intersection(b)
{11, 2, 10}
>>> a.intersection_update(b)
>>> a
{11, 2, 10}
>>>
```

Conjuntos: diferença

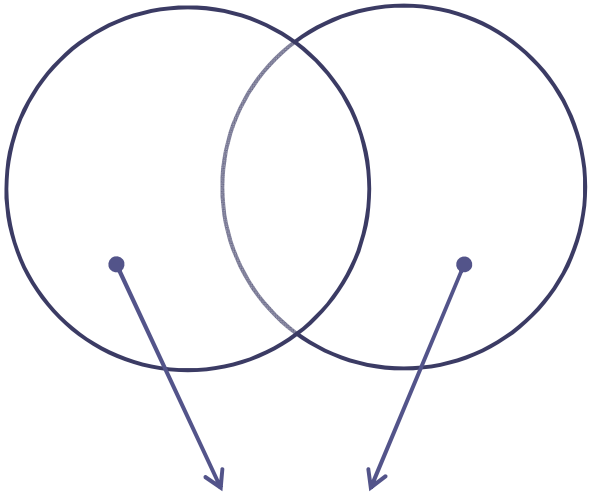


A - B

```
>>> a
{2, 3, 7, 10, 11}
>>> a.difference(b)
{3, 7}
>>> b.difference(a)
{4, 6}
>>>
```

Conjuntos: diferença simétrica

- É a operação entre dois conjuntos que retorna todos os elementos (de ambos conjuntos) que pertencem a somente um dos conjuntos.



```
>>> a.symmetric_difference(b)
{3, 4, 6, 7}
```

Conjuntos

```
>>> set1 = {'a','e','i','o','u'}
>>> set1
{'i', 'u', 'e', 'o', 'a'}
>>> set1.add('Vogais')
>>> set1
{'o', 'i', 'u', 'Vogais', 'e', 'a'}
>>> set1.clear()
>>> set1
set()
>>> set1 = {'a','e','i','o','u'}
>>> set2 = set1.copy()
>>> set2
{'i', 'u', 'e', 'o', 'a'}
>>> set1.pop()
'i'
>>> set1
{'u', 'e', 'o', 'a'}
>>> set2.discard('i')
>>> set2
{'u', 'e', 'o', 'a'}
```

Conjuntos

```
>>> set1 = {'a','e','i','o','u'}
>>> set2 = {'i','j','k','x','y'}
>>> set2.add('z')
>>> set2
{'x', 'k', 'j', 'i', 'y', 'z'}
>>> set1.intersection(set2)
{'i'}
>>> set3 = set1.difference(set2)
>>> set3
{'a', 'e', 'o', 'u'}
>>> set1.isdisjoint(set2)
False
>>> set1.issubset(set2)
False
>>> set2.issuperset(set2)
True
>>> set4 = set1.union(set2)
>>> set4
{'e', 'x', 'k', 'j', 'i', 'o', 'y', 'a', 'z', 'u'}
>>> set5 = set4.symmetric_difference(set3)
>>> set5
{'x', 'k', 'j', 'i', 'y', 'z'}
```

Dicionários (*Dict*)

- Tipo de dado composto – mutável
- São coleções que fazem associações com dois valores, o primeiro é identificado como a chave (*key*) e o segundo faz referência ao conteúdo
- Aceita como chave qualquer tipo imutável, na maioria das vezes utilizam *strings*, mas é possível definir chaves com tuplas ou valores numéricos
- Os conteúdos podem ser valores mutáveis ou imutáveis

Dicionários (*Dict*)

```
>>> agenda = {}
>>> agenda['nome'] = 'João'
>>> agenda['fone'] = '99999-1111'
>>> agenda['e_mail'] = 'joao@teste.com.br'
>>> agenda['dt_nasc'] = '20/03/2000'
>>> agenda
{'nome': 'João', 'fone': '99999-1111', 'e_mail': 'joao@teste.co
m.br', 'dt_nasc': '20/03/2000'}
>>> agenda2 = {'nome2': 'Maria', 'fone2': '99999-2222', 'e_mail
2': 'maria@teste.com.br', 'dt_nasc2': '10/04/1998'}
>>> agenda2
{'nome2': 'Maria', 'fone2': '99999-2222', 'e_mail2': 'maria@tes
te.com.br', 'dt_nasc2': '10/04/1998'}
>>> agenda3 = dict([('nome3','José'), ('fone3', '99999-3333'),
('e_mail3', 'jose@teste.com.br'), ('dt_nasc3', '25/08/1990')])
>>> agenda3
{'nome3': 'José', 'fone3': '99999-3333', 'e_mail3': 'jose@teste
.com.br', 'dt_nasc3': '25/08/1990'}
>>> agenda4 = dict(nome4 = 'Eva', fone4 = '99999-4444', e_mail4
= 'eva@teste.com.br', dt_nasc4 = '15/02/1985')
>>> agenda4
{'nome4': 'Eva', 'fone4': '99999-4444', 'e_mail4': 'eva@teste.c
om.br', 'dt_nasc4': '15/02/1985'}
```

Dicionários (*Dict*)

Procedimento	Descrição
Acessar um conteúdo	Informar o nome do dicionário e identificar a chave entre colchetes
Alterar o conteúdo	Informar o nome do dicionário identificando a chave entre colchetes, posteriormente deve ser feita a atribuição do novo valor
Apagar um elemento	Utiliza-se o comando *del*, informando o nome do dicionário com a chave entre colchetes
Verificar o tamanho	Fazer uso da função *len*, passando como parâmetro o nome do dicionário
Verificar a existência de uma chave no dicionário	Fazer uso do operador *in*

Dicionários (*Dict*)

```
>>> agenda = {'nome': 'João', 'fone': '99999-1111', 'e_mail': '
joao@teste.com.br', 'dt_nasc': '20/03/2000'}
>>> agenda2 = {'nome2': 'Maria', 'fone2': '99999-2222', 'e_mail
2': 'maria@teste.com.br', 'dt_nasc2': '10/04/1998'}
>>> print (agenda['nome'])
João
>>> print (agenda2['fone2'])
99999-2222
>>> agenda['dt_nasc'] = '20/04/1998'
>>> agenda
{'nome': 'João', 'fone': '99999-1111', 'e_mail': 'joao@teste.co
m.br', 'dt_nasc': '20/04/1998'}
>>> agenda2['fone2'] = '99999-3331'
>>> agenda2
{'nome2': 'Maria', 'fone2': '99999-3331', 'e_mail2': 'maria@tes
te.com.br', 'dt_nasc2': '10/04/1998'}
>>> del(agenda['dt_nasc'])
>>> agenda
{'nome': 'João', 'fone': '99999-1111', 'e_mail': 'joao@teste.co
m.br'}
>>> del(agenda2['e_mail2'])
>>> agenda2
{'nome2': 'Maria', 'fone2': '99999-3331', 'dt_nasc2': '10/04/19
98'}
>>> len(agenda)
3
>>> len(agenda2)
3
>>> 'cpf2' in agenda2
False
>>> 'nome2' in agenda2
True
>>> 'fone' in agenda
True
```

Dicionários (*Dict*)

- A iteração de um dicionário com o uso do laço *for* vai ocorrer sobre as chaves do dicionário

```
>>> agenda = {'nome': 'João', 'fone': '99999-1111', 'e_mail': '
joao@teste.com.br', 'dt_nasc': '20/03/2000'}
>>> for k in agenda:
        print(k)


nome
fone
e_mail
dt_nasc
>>> for k in agenda:
        print(agenda[k])


João
99999-1111
joao@teste.com.br
20/03/2000
```

Dicionários (*Dict*)

- Métodos da classe Dict

Método	Descrição
clear()	Remove todos os elementos do dicionário
copy()	Retorna um novo dicionário com os mesmos elementos, cujo valores apontam para a mesma referência
fromkeys(lista, valor)	Retorna um novo dicionário tendo como chaves os itens de lista e os valores todos iguais ao parâmetro valor. Se valor não for informado será *None*
get(chave)	Retorna o valor do elemento identificado pela chave ou *None* quando não existir
get(chave,alternativo)	Retorna o valor do elemento identificado pela chave ou o valor definido como alternativo quando não existir
items()	Retorna uma lista com todos os elementos (chave:valor) do dicionário
keys()	Retorna uma lista com todas as chaves do dicionário
pop(chave)	Retorna o valor da chave e remove o elemento
popitem()	Retorna e remove um elemento aleatório do dicionário
update(dicionáro)	Atualiza o dicionário com os elementos de outro dicionário
values()	Retorna uma lista com todos os valores dos elementos

Dicionários (*Dict*)

```
>>> agenda = {'nome': 'João', 'fone': '99999-1111', 'e_mail': 'joao@teste.com.br'
, 'dt_nasc': '20/03/2000'}
>>> agenda_aux = agenda.copy()
>>> agenda_aux
{'nome': 'João', 'fone': '99999-1111', 'e_mail': 'joao@teste.com.br', 'dt_nasc':
'20/03/2000'}
>>> agenda_aux['sena']=[15,28,36,47,48,52]
>>> agenda_aux
{'nome': 'João', 'fone': '99999-1111', 'e_mail': 'joao@teste.com.br', 'dt_nasc':
'20/03/2000', 'sena': [15, 28, 36, 47, 48, 52]}
>>> agenda_aux['sena'] +=[57]
>>> agenda_aux
{'nome': 'João', 'fone': '99999-1111', 'e_mail': 'joao@teste.com.br', 'dt_nasc':
'20/03/2000', 'sena': [15, 28, 36, 47, 48, 52, 57]}
>>> agenda
{'nome': 'João', 'fone': '99999-1111', 'e_mail': 'joao@teste.com.br', 'dt_nasc':
'20/03/2000'}
>>> agenda['nome'] = 'João Carlos'
>>> agenda
{'nome': 'João Carlos', 'fone': '99999-1111', 'e_mail': 'joao@teste.com.br', 'dt_
nasc': '20/03/2000'}
>>> agenda_aux
{'nome': 'João', 'fone': '99999-1111', 'e_mail': 'joao@teste.com.br', 'dt_nasc':
'20/03/2000', 'sena': [15, 28, 36, 47, 48, 52, 57]}
>>> agenda_aux.clear()
>>> agenda_aux
{}
```

Dicionários (*Dict*)

```
>>> agenda_aux = {}
>>> dic = agenda_aux.fromkeys(['chave1','chave2','chave3'],5)
>>> dic
{'chave1': 5, 'chave2': 5, 'chave3': 5}
>>> agenda = {'nome': 'João', 'fone': '99999-1111', 'e_mail': '
joao@teste.com.br', 'dt_nasc': '20/03/2000'}
>>> print(agenda.get('nome'))
João
>>> print(agenda.get('Eva'))
None
>>> print(agenda.get('Eva','Erro!'))
Erro!
>>> lista = agenda.items()
>>> lista
dict_items([('nome', 'João'), ('fone', '99999-1111'), ('e_mail'
, 'joao@teste.com.br'), ('dt_nasc', '20/03/2000')])
>>> chaves = agenda.keys()
>>> chaves
dict_keys(['nome', 'fone', 'e_mail', 'dt_nasc'])
>>> conteudos = agenda.values()
>>> conteudos
dict_values(['João', '99999-1111', 'joao@teste.com.br', '20/03/
2000'])
```

Dicionários (*Dict*)

```
>>> agenda = {'nome': 'João', 'fone': '99999-1111', 'e_mail': '
joao@teste.com.br', 'dt_nasc': '20/03/2000'}
>>> agenda2 = {'nome2': 'Maria', 'fone2': '99999-2222', 'e_mail
2': 'maria@teste.com.br', 'dt_nasc2': '10/04/1998'}
>>> agenda2.pop('nome2')
'Maria'
>>> agenda2
{'fone2': '99999-2222', 'e_mail2': 'maria@teste.com.br', 'dt_na
sc2': '10/04/1998'}
>>> agenda2.popitem()
('dt_nasc2', '10/04/1998')
>>> agenda2
{'fone2': '99999-2222', 'e_mail2': 'maria@teste.com.br'}
>>> agenda.update(agenda2)
>>> agenda
{'nome': 'João', 'fone': '99999-1111', 'e_mail': 'joao@teste.co
m.br', 'dt_nasc': '20/03/2000', 'fone2': '99999-2222', 'e_mail2
': 'maria@teste.com.br'}
>>> agenda.update(fone = '99999-1114', fone2 = '99999-2221')
>>> agenda
{'nome': 'João', 'fone': '99999-1114', 'e_mail': 'joao@teste.co
m.br', 'dt_nasc': '20/03/2000', 'fone2': '99999-2221', 'e_mail2
': 'maria@teste.com.br'}
```