Inatel
Instituto Nacional de Telecomunicações

AC-309 – Atividades Complementares

(Engenharias Biomédica e de Telecomunicações)

4. Tuplas, Dicionários e Matrizes

Prof. Evandro Luís Brandão Gomes

Tuplas

Tupla, assim como a Lista,é um conjunto sequencial de valores, onde cada valor é identificado através de um índice.

A principal diferença entre elas é que as tuplas são imutáveis, ou seja, seus elementos não podem ser alterados.

Dentre as utilidades das tuplas, destacam-se as operações de empacotamento e desempacotamento de valores.

Uma tupla em Python é declarada da seguinte forma:

Nome_tupla = (valor1, valor2, ..., valorN)

Tuplas

Exemplos:

```
>>> tupla = (0, 1, 2, 'muito parecio com listas')
>>> tupla
(0, 1, 2, 'muito parecido com listas')
>>> tupla = 0, 1, 2, 'muito parecio com listas'
>>> tupla
(0, 1, 2, 'muito parecido com listas')
>>> # Parenteses são opcionais.
```

```
T = (1,2,3,4,5)
print(T)
(1, 2, 3, 4, 5)
print(T[3])
4
T[3] = 8
Traceback (most recent call last):
  File "C:/Python34/teste.py", line 4, in <module>
    T[3] = 8
TypeError: 'tuple' object does not support item assignment
```

Tuplas

Uma forma muito utilizada em tuplas é o desempacotamento, que permite atribuir os elementos armazenados em uma tupla a diversas variáveis.

Exemplo:

```
T = (10,20,30,40,50)
a,b,c,d,e = T
print("a=",a,"b=",b)
a= 10 b= 20
print("d+e=",d+e)
d+e= 90
```

Tuplas

É possível se converter listas em tuplas ou vice-versa.

```
>>> # Lista para Tupla:
. . .
>>> lista = [0, 1, 2, 3]
>>> tupla = tuple(lista)
>>> tupla
(0, 1, 2, 3)
>>>
>>> # Tupla para Lista:
. . .
>>> lista = list(tupla)
>>> lista
[0, 1, 2, 3]
```

Dicionários

Dicionário é um conjunto de valores, onde cada valor é associado a uma chave de acesso.

Um dicionário em Python é declarado da seguinte forma:

```
Nome_dicionario = {  chave1: valor1,
                     chave2: valor2,
                     chave3: valor3,
                     ......
                     chaveN: valorN}
```

Dicionários

Exemplo:

```
D={"arroz": 17.30, "feijão":12.50,"carne":23.90,"alface":3.40}
print(D)
{'arroz': 17.3, 'carne': 23.9, 'alface': 3.4, 'feijão': 12.5}
print(D["carne"])
23.9
print(D["tomate"])
Traceback (most recent call last):
  File "C:/Python34/teste.py", line 4, in <module>
    print(D["tomate"])
KeyError: 'tomate'
```

Dicionários

É possível acrescentar ou modificar valores no dicionário:

```
D["carne"]=25.0
D["tomate"]=8.80
print(D)
{'alface':3.4 ,'tomate':8.8,'arroz':17.3,'carne':25.0, 'feijão':12.5}
```

Os valores do dicionário não possuem ordem, por isso a ordem de impressão dos valores não é sempre a mesma.

Dicionários

Operações:

A seguir, são apresentados alguns comandos para a manipulação de dicionários.

Comando	Descrição	Exemplo	

del	Exclui um item informando a chave.	`del D["feijão"]`
		`print(D)`
		`{'alface':3.4 'tomate':8.8,'arroz':17.3,'carne':25.0}`	

in	Verificar se uma chave existe no dicionário.	`"batata" in D` `False`	`"alface" in D` `True`
keys()	Obtém as chaves de um dicionário.	`D.keys()` `dict_keys(['alface', 'tomate','carne', 'arroz'])`	
values()	Obtém os valores de um dicionário.	`D.values()` `dict_values([3.4, 8.8, 25.0, 17.3])`	

Dicionários

A seguir são mostrados alguns métodos e funções geralmente utilizadas com dicionários.

```
>>> tel = {'John' : 456789, 'Peter' : 784785, 'Vini' : 987584, 'James'
                                        : 142365, 'Mary' : 456987}
>>> # Adicionar um novo número à agenda:
. . .
>>> tel['Paul'] = 101010
>>> tel
{'John': 456789, 'Peter': 784785, 'Vini' : 987584, 'James': 142365, '
                                    Mary': 456987, 'Paul': 101010}
>>>
>>> # Verificar se existe uma chave no dicionário:
. . .
>>> 'Paul' in tel
True
>>>
>>> # Retirar um telefone da agenda:
. . .
>>> del tel['James']
>>> tel
{'John': 456789, 'Peter': 784785, 'Vini' : 987584, 'Mary': 456987, '
                                    Paul': 101010}
```

Dicionários

Continuação:

```
>>>
>>> # Método que mostra os itens com suas chaves de um dicionário:
. . .
>>> tel.items()
dict_items([('John', 456789), ('Peter', 784785), ('Vini', 987584), ('
                                    Mary', 456987), ('Paul', 101010)])
>>>
>>> # Método que mostra todas as chaves de um dicionário:
. . .
>>> tel.keys()
dict_keys(['John', 'Peter', 'Vini', 'Mary', 'Paul'])
>>>
>>> # Método que mostra todos os valores presentes em um dicionário:
. . .
>>> tel.values()
dict_values([456789, 784785, 456987, 101010])
>>>
>>> # A função builtin "len()" retorna o tamanho da lista (neste caso,
                                    o número de chaves):
. . .
>>> print(len(tel))
4
```

Matrizes

Em Python, não existe uma função padrão que trabalhe com matrizes. Entretanto, é relativamente fácil e intuitivo a criação destas.

É possível implementar uma matriz através da criação de uma lista, cujos elementos são outras listas.

```
mat = []
linhas = 5
colunas = 5
for i in range(linhas):
    lin = []
    for j in range(colunas):
        lin.append(int(i))
    mat.append(lin)
for i in mat: print(i)
```

```
## RESULTADO:
[0, 0, 0, 0, 0]
[1, 1, 1, 1, 1]
[2, 2, 2, 2, 2]
[3, 3, 3, 3, 3]
[4, 4, 4, 4, 4]
```

Exercícios:

1 – Dada a tabela a seguir, crie um dicionário que a represente:

Lanchonete	
Produtos	**Preços R$**
Salgado	R$ 4.50
Lanche	R$ 6.50
Suco	R$ 3.00
Refrigerante	R$ 3.50
Doce	R$ 1.00

2 – Considere um dicionário com 5 nomes de alunos e suas notas. Escreva um programa que calcule a média dessas notas.

Exercícios:

3 – Faça um dicionário com as 5 pessoas mais perto de você, tendo o nome como chave e a cor da camisa que está usando como valor.

4 – Crie um dicionário vazio semana = {} e o complete com uma chave para cada dia da semana, tendo como seu valor uma lista com as aulas que você tem nesse dia (sábado e domingo recebem listas vazias).