Tuplos

Programação I **2016.2017**

Teresa Gonçalves
tcg@uevora.pt

Departamento de Informática, ECT-UÉ

Reminder...

Como aprender?

Estudar, estudar, estudar...

Praticar, praticar, praticar…

Cometer erros, cometer erros, cometer erros…

Aprender com os erros, aprender com os erros, aprender com os erros ...

Sumário

Propriedades, operadores e indexação

Iteração

Funções pré-definidas

Tuplos e definição de funções

Tuplos vs. Listas

Propriedades, operadores e indexação

Tuplo

Sequência de itens

- Os itens podem ser de tipos diferentes
- Usualmente entre ()
  - Mas não é obrigatório!

Tamanho da sequência

- Função len()

Exemplo

```
>>> coordenadas=(1,7,-1)
>>> type(coordenadas)
<class 'tuple'>
```

Propriedades

Sequencial

Relação de ordem entre elementos

Indexável

Acesso direto a cada elemento através do índice

Imutável

Não é possível alterar o valor de um item do tuplo

```
>>> coordenadas=(4,7,-1)
>>> coordenadas[0]=5
traceback(most recent call)
TypeError: 'tuple' object does not support item
assignment
```

Geração de tuplos

A atribuição de uma sequência de valores separados por vírgula gera um tuplo

Exemplo

```
>>> tup1 = 2011,
>>>  type(tup1)
<class ‘tuple’>
>>>  tup2=9,8,7
>>>  type(tup2)
<class ‘tuple’>
>>> tup3=(1,’dois’,[3],(4),(5,))
>>>  type(tup3[3])
<class ‘int’>
>>>  class(tup3[4]
<class ‘tuple’>
```

Operadores básicos

Concatenação

```
>>> a=(2,1); b=('ola',5,1)
>>> print(a+b)
(2,1,'ola',5,1)
```

Repetição

```
>>> tuplo=b*2
>>> tuplo
('ola,5,1,'ola,5,1)
```

Tuplos vazios, índices e segmentos

Tuplo vazio

Tuplo com zero itens

Indicado por ()

Índices e segmentos

Têm a mesma semântica que os outros tipos (strings, listas)

Os segmentos de tuplos geram tuplos!

Exemplo

```
>>> t=('a', 'b', 'c', 'd', 'e', 'f')
>>> t[-1]
'f'
>>>t[0::2]
('a', 'c', 'e')
>>> t[-2:]
('e', 'f')
```

Operador ==

Verificação de equivalência

Operador binário booleano

Retorna True se os tuplos tiverem os mesmos elementos

A ordem tem importância

Exemplo

```
>>> a=(1,2,3); b=(); c=(1,3,2); d=(1,2,3)
>>> a==b
False
>>> a==c
False
>>> a==d
True
```

Operador is

Verificação de identidade

Operador binário booleano

Retorna True as variaveis referem o mesmo tuplo

Exemplo

```
>>> a=(1,2); b=a; c=(1,2)
>>> a==b
True
>>> a is b
True
>>> a==c
True
>>> a is c
False
```

Operador in

expr in tuplo

Operador binário booleano

Operação entre um item e um tuplo

Retorna True se o resultado da expressão é um item do tuplo

Exemplo

```
>>> tuplo=('pao','queijo,'fiambre','manteiga')
>>> 'pao' in tuplo
True
>>> 'azeite' in tuplo
False
```

Operadores <, >

Comparação

- Comparam o 1º item
  - Se maior/menor, os restantes itens não são considerados
  - Se igual compara-se o seguinte…

Exemplo

```
>>> a=(1,2,3); b=(1,3,3); c=(0,3)
>>> a<b
True
>>> a>b
False
>>> a>c
True
```

Iteração

Iteração sobre tuplos

Iterar na sequência

```
for item in lista:
    print( item )
```

Iterar sobre os índices (for)

```
for indice in range(len(tuplo)):
    print( tuplo[indice] )
```

Iterar sobre os índices (while)

```
i=0
while  i<len(tuplo):
    print( tuplo[i] )
    i = i+1
```

Packing & Unpacking

Packing

Criar um tuplo a partir de uma sequência de valores

```
>>> t=1,2,3
>>> type(t)
<class 'tuple'>
```

Unpacking

Atribuir valores de um tuplo a variáveis

```
>>> a,b,c = t
>>> b
2
```

Packing & Unpacking

Packing e unpacking

```
>>> a,b=b,a      # troca o valor das variáveis
```

Funcionamento

- Primeiro são avaliadas as expressões do lado direito
- Depois o respetivo alor é atribuido a cada variável do lado esquerdo

Nota

- A maioria das linguagens de programação não permite esta construção!!!

Funções

tuple()

Gera um tuplo cujos elementos são os elementos do argumento

Argumento é valor iterável

Exemplo

```
>>> a=tuple()
>>> b=tuple('exemplo')
>>> b
('e', 'x', 'e', 'm', 'p', 'l', 'o')
>>> c=tuple([1,2,3,4,5])
>>> c
(1,2,3,4,5)
```

zip()

Cria uma sequência de tuplos a partir de cada uma das sequências dos argumentos

- Cada tuplo i inclui o i-ésimo elemento de cada sequência dos argumentos
- O comprimento é o menor comprimento dos argumentos
- Devolve um tipo sequencial, iterável mas não indexável

Exemplo

```
>>> a=(1,2); b=(2,4)
>>> c=zip(a,b)
>>> for i in c
        print(i)
(1,2)
(2,4)
```

Exemplo: s1[i]==2*s2[i]?

Verificar se cada valor de uma sequência é o dobro de cada valor da outra

Versão 1

```
i=0
while i<len(s1):
  if s1[i]!=2*s2[i]:
    return False
  i= i+1
return True
```

Versão 2

```
for a,b in zip(s1,s2):
  if a!=2*b:
    return False
return True
```

Tuplos e definição de funções

Definição de funções

Tuplos tornam possível

Devolver mais que um valor como resultado da função (através de packing)

Passar um número variável de valores como argumentos

```
>>> max(5,6)
6
>>> max(5,6,1)
6
>>> max(5,6,1,9)
9
```

Devolução de mais de um valor

Dividir 2 números e devolver quociente e resto

```
>>> def divide(a,b):
        q=0; r=0
        while a>b:
            a=a-b
            q=q+1
        r=a
        return(q,r)
>>> quo,resto = divide(15,4)
>>> quociente
3
>>> resto
3
```

Definição de um nº variável de argumentos

Marcar o parâmetro com o *

Identifica-o como sendo um tuplo

```
>>> def maximo(*valores):
        if len(valores)==0:
          return None
        m=valores[0]
        for v in valores[1:]
          if v>m:
            m=v
        return  m
```

Passagem de tuplo como argumento

Marcação do argumento com *

Permite passar um tuplo a funções que não aceitam tuplos

```
>>> range((1,6))
Traceback (most recent call last): TypeError: range()
integer end argument expected, got tuple.
>>> range(*(1,6))
range(1,6)
```

Tuplos vs. Listas

Tuplos vs. Listas

Tuplos

- ()
- Imutavel
- Convenção
  - Heterogéneo: sequencia de diferentes tipos de “coisas”
  - Constitui uma unidade coerente

Listas

- []
- Mutável
- Convenção
  - Homogéneo: sequencia do mesmo tipo de “coisas”
  - Cada elemento é tratado individualmente