Laboratório de Programação 1

Aula 10

Mário Hozano
professor@hozano.com

Ciência da Computação
UFAL - Arapiraca

Relembrando a aula anterior...

- O que são listas?
- Como acessar os items de uma lista?
- Para que serve a função *len() ?*
- O que são *sublistas*? Como obter?
- Como remover items de uma lista?
- Para que servem as funções *append()*, *extend()* e *sort()*?
- Quais as relações entre listas e *strings* ?

Roteiro da aula

- Tuplas
- Acessando os itens de uma tupla
- A função *len()* e o operador *slice*
- Características de Mutação
- Atribuição de Tuplas
- Tuplas como valores de retorno
- Tuplas como argumentos de funções

Tuplas

- Vimos anteirormente dois tipos compostos: *strings* e listas
- *Strings* são compostas por caracteres e listas por valores de qualquer tipo
- *Strings* são imutáveis e listas são mutáveis
- Tuplas são coleções de valores de qualquer tipo assim como listas, entretanto **tuplas são imutáveis**
- Assim, tuplas são coleções de valores separados por vírgulas

```
>>> tupla = 'Maria', True, 34
>>> tupla
('Maria', True, 34)
```

Tuplas

- Embora não seja necessário é convencional colocar os valores das tuplas entre parênteses

```
>>> tupla = 'Maria', True, 34
>>> tupla
('Maria', True, 34)
>>> tupla = ('Maria', True, 34)
>>> tupla
('Maria', True, 34)
```

- Uma tupla que não contém elementos é uma tupla vazia e expressa por *()*

```
>>> tupla = ()
```

Tuplas

- Tuplas de um elemento devem ter uma vírgula no final

```
>>> a = ('Maria')
>>> type(a)
<type 'str'>
>>> a = ('Maria',)
>>> type(a)
<type 'tuple'>
```

- Os elementos de uma tupla podem ser acessados com índices, assim como listas

```
>>> tupla = 'Maria', True, 34
>>> tupla[0]
'Maria'
```

Tuplas – A função *len()*

- Assim como em listas a função *len()* pode ser aplicada em tuplas
- Neste caso, a função retorna o número de elementos (tamanho) da tupla dada

```
>>> tupla = (10, 'joao', 13.3, False)
>>> len(tupla)
4
>>> len(('A', 22, True))
3
```

Tuplas – Índices Negativos

- Os elementos das tuplas também podem ser acessados através de índices com valores negativos

- O índice -1 representa o último item, o índice -2 o penúltimo e assim sucessivamente

```
>>> tupla = (10, 'joao', 13, False)
>>> tupla[-1]
False
>>> tupla[-2]
13
```

Tuplas – O operador *slice* (:)

- Assim como *strings* e listas, o operador *slice* (:) pode ser utilizado com tuplas

```
>>> tupla = (10, 'joao', 13, False)
>>> tupla[1:3]
('joao', 13)
>>> tupla[:2]
(10, 'joao')
>>> tupla[1:]
('joao', 13, False)
>>> tupla[:]
(10, 'joao', 13, False)
```

Tuplas – Características de mutação

- Assim como *strings*, tuplas são imutáveis
- Uma vez criada não é possível alterar o valor de um determinado elemento da tupla
- Caso o programador tente alterar um item de uma tupla, o interpretador irá gerar um erro

```
>>> tupla = 'Maria', True, 34
>>> tupla[0] = 'Mario'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>>
```

Atribuição de tuplas

- O mecanismo de atribuição de tuplas permite atribuir valores a diversas variáveis com um único comando
- Para isso, as variáveis devem ser escritas entre virgulas e seus respectivos valores também

```
>>> a, b, c, d = 10, 20, 30, 40
>>> c
30
```

- Caso o número de variáveis seja diferente do número de valores, o interpretador irá reproduzir um erro

```
>>> a, b, c = 13, 20, 13, 40
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: too many values to unpack
```

Atribuição de tuplas

- Em alguns casos é necessário trocar os valores de duas variáveis
- Com atribuições convencionais, isto seria feito com o uso de uma variável auxiliar como a seguir

```
>>> aux = a
>>> a = b
>>> b = aux
```

- A atribuição de tuplas pode ajudar neste processo como a seguir

```
>>> a, b = b, a
```

Tuplas como valores de retorno

- Em alguns casos é necessário retornar múltiplos valores a partir de uma função
- Tuplas podem ser utilizadas para este fim como na função a seguir que retorna os valores dados como argumentos com a ordem inversa

```
def troca(a,b):
    return b,a
```

- A função anterior deveria ser chamada assim:

```
x, y = troca(x, y)
```

Tuplas como argumentos de uma função

- Uma função recebe um determinado número de argumentos

```
def soma(a, b):
    return a + b
```

```
def soma(a, b, c):
    return a + b + c
```

- Tuplas oferecem um mecanismo que permite que uma função receba um número indeterminado de argumentos
- Para isso, o parâmetro da função deve ser precedido com um asterisco (*)
- Este parâmetro será definido como uma tupla contendo como elementos os argumentos passados para a função

Tuplas como argumentos de uma função

- A seguir uma função que recebe um número indeterminado de argumentos

```
def soma(*argumentos):
    resultado = 0
    for item in argumentos:
        resultado += item

    return resultado
```

- A função acima retorna a soma de todos os valores dados como argumentos

Exercícios

1. Faça um programa que receba uma *string* e retorne uma tupla com suas letras

2. Escreva uma função que receba uma lista e retorne uma tupla com seus elementos

3. Escreva uma função que retorne uma tupla com os itens invertidos de uma outra tupla passada como argumento

4. Faça uma função que receba um número indeterminado de números como argumentos e retorne os itens multiplicados