Tuplos e Listas

Carlos Tavares

October 28, 2024

Tuples

Tuplos i

Some information makes more sense together. For instance, **a point**:

- We have been representing a point using two variables:

  » x = 5
  » y = 5

- We can represent the same information in a single variable using a **tuple:**

  » point = (5,5)

Tuplos ii

Another example: a **person**

» # (Name, Height, Weight (Kg), Blood type)
» person = ("Jonh doe", 1.80, 65, "AB+")

Informally, it corresponds to the notion of **record**, i.e. the aggregation of information about an object:

Tuplos iii

Syntactically, a tuple is represented as a construction such like:

» $(\text{field}_1, \text{field}_2, \text{field}_3, \ldots, \text{field}_n)$

where n is any finite dimension.

It is a way of building composite types in python

$$Type \times Type \ldots \times Type$$

Tuple in python are immutable.

How to deal with tuples in python i

Acessing elements of the tuple in **python**:

variable [index]

Example: the variable point

» point [0]
5
» point [1]
6

How to deal with tuples in python ii

variable [i:j]

» person [1:3]
(23,42)

» person [1:]
(1.8, 65, "AB+")

» person [:3]
('Jonh Doe', 1.8, 65)

How to deal with tuples in python iii

Pattern matching using tuples:

» (**name**, **height**, **weight**, **blood_type**) = person
» **name**
Joaquim
» **height**
1.80

Tuples are iterable:

```
for i in person:
    print (i)
```

Tuples immutability i

Tuples are immutable

» person[0] = "Doe Jonh"
TypeError: 'tuple' object does not support item assignment

However it is always possible to create new tuples!

» person = ("Unknown", 1.60, 65, "0-")

A change can be done by replacing just one field:
» person = person [0] + (1,70, 80, "0-")

It is also possible to create copies of tuples
» person *= 2

Lists

Lists i

A list is syntatically represented in **python** as:

[elem_1, elem_2, . . ., ... , elem_3]

Example:

- $[12, 13, 14, 15, 16]$
- $['a', 'b', 'c', 'd']$

Tuples vs **Lists**

Tuples are immutable, i.e. once defined its size cannot be extended. Lists can be added and removed elements arbitrarily.

Lists ii

More examples

- Empty list - []
- List of lists - [[1, 2, 3, 4], [4, 3, 2, 1]]
- List of lists of lists - [[[1, 2, 3, 4], [4, 3, 2, 1]], [[5, 6, 7, 8], [8, 7, 6, 5]]]
- Lists of different types - [12, "Joaquina", [1,2,3,4]]

Accessing list elements (Same as in tuples):

list_variable [index]

Lists iii

By pattern matching:

```
» [x, y, z] = [12, "Joaquina", [1,2,3,4]]
» x
12
» y
?
» z
?
» z [3]
?
```

Lists iv

One can **select** intervals from lists:

» l = [1,2,3,4,5,6,7,8,9, 10]

» **print** (l)
[1,2,3,4,5,6,7,8,9, 10]

» **print** (l[4:8])
[5, 6, 7, 8]

» **print** (l[:7])
[1, 2, 3, 4, 5, 6, 7]

» **print** (l[3:])
[4, 5, 6, 7, 8, 9, 10]

Lists v

```
» print (l[0:-2])
[1, 2, 3, 4, 5, 6, 7, 8]
```

Lists vi

Lists are also iterable objects

```
for i in l:
        print (i)
```

Alternatively, one can use the **len** function

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

Addition of elements to a list

The **append** function:

list.**append** (element)

The **insert** function:

list.**insert** (index, element)

Example:

```
list = [1, 2, 3, 10]
list.append (11)
list.insert (0, 0)
```

Remove elements from a list i

The **remove** function:

list.remove (item)

The **pop** function:

list.pop (index)

Example:

l = [1,2,3,4]
l.remove (3)
l.pop(2) » l
[1, 2]

Remove elements from a list ii

The **clear** function

The function clear erases all elements from a list.

Example

```
» l = [1,2,3,4,5]
» l.clear ()
» l
[]
```

Useful functions for lists i

The **sort** function (Sort a list in ascending roder)

```
» l = [5,4,3,2,1]
» l.sort ()
» l
[1,2,3,4,5]
```

The **reverse** function

```
» l = [4,5,3,1,2]
» l.reverse ()
» l
[2,1,3,5,4]
```

Useful functions for lists ii

Convert a tuple into a list. The list constructor:

Example:

```
» t = ("Jonh Doe", 87, 1.80)
»l = list (t)
»l
["Jonh Doe", 87, 1.80]
```

Useful functions for lists iii

Count the instances of an element in a list. The **count** function.

Example:

```
» l = [2, 2, 3 ,4 ,4, 4, 4, 5]
» l.count(4)
4
»l.count(3)
1
```

Exercise 1: Given a list, make a function to retrieve the minimum element from it.

Exercise 2: Given a list, make a function to eliminate repeated elements from it.

Exercise 3: Consider a public service, that needs the implementation of a program to manage the queue of clients, that has the following requirements:

- Clients register their name, age and priority (True or False);
- The queue must contain the clients with priority before the ones that do not have it.
- There must be an option to show who is waiting in the queue;
- There must be an option to remove a client from the queue, from a specific position.

Questions?