Llistes a python

Author: David Marquez i Ferran Godoy
Date: 13/02/2018

Llista

La llista és un tipus de dada molt versàtil que es pot escriure com a una llista de valors separats per coma (els *items*) dins de brackets.

Crear una llista és ben senzill, simplement cal posar varis valors separats per coma entre brackets.

Per exemple:

```
llista1 = ["info", "mates", "física"]
llista2 = [1, 2, 3, 4, 5]
llista3 = ["hola", 1]
```

Accedint a valors en llistes

Per accedir a valors en llistes, cal fer servir els brackets amb un/dos indexs per obtenir el valor/valors en aquell índex. El primer índex és 0.

Per exemple:

```
llista1 = ["info", "mates", "física"]
llista2 = [1, 2, 3, 4, 5]
print "llista1[0]: ", llista1[0]
print "llista2[0:2]: ", llista2[1:4]
```

Quan s'executa aquest codi es produirà el següent resultat:

```
llista1[0]: info
llista[0:2]: [2, 3, 4]
```

Canviant els valors d'una llista

Per canviar el valor d'un o varis valors d'una llista cal accedir-hi de la manera mostrada anteriorment i assignar un valor o conjunt de valors.

Per exemple:

```
llista1 = ["info", "mates", "física"]
llista2 = [1, 2, 3, 4, 5]
llista1[2] = "hola"
llista2[0:2] = [2, 3]
```

D'aquesta manera fariem que les llistes siguessin així:

```
llista1 = ["info", "mates", "hola"]
llista2 = [2, 3, 3, 4, 5]
```

Mètodes de les llistes

Les llistes tenen molts mètodes predefinits per facilitar-ne el seu ús. Uns d'ells sería:

1. list.append(obj): Afegiex l'objecte obj a la llista
2. list.count(obj): Retorna les vegades que apareix l'objecte obj en la llista
3. list.remove(obj): Borra l'objecte obj de la llista
4. list.reverse(): Inverteix l'ordre de la llista
5. ...

Aquests són alguns dels mètodes que la llista conté. Per més informació visita la pàgina web oficial de Python.

Iterar en una llista

Iterar en una *llista* és un procediment molt típic i molt necessàri per aprofitar-se de tot el seu potencial.

Es pot fer de dues maneres diferents:

1. Loop while
2. Loop for

1. Loop while

El loop *while* s'utilitza per repetir seccions de codi mentre es cumpleixi una condició (*booleà*).

Per aquesta raó, podriem iterar tantes vegades com elements en una llista i canviar-ne els seus valors si fos necessari utilitzant un comptador.

```
comptador = 0
llista2 = [2, 3, 3, 4, 5]
while(comptador < len(llista2)):
    llista2[comptador] += 1
    comptador +=1
```

Aquest codi generaria la següent llista:

```
llist2 = [3, 4, 4, 5, 6]
```

També és pot fer servir *break*, que s'utilitza per sortir del loop while independentment de si la condició es cumpleix o no.

Així, quan trobessim un element que cumplís quelcom, podriem fer *break* i sortir del while, evitant passar per els altres valors de la llista innecessariament.

Exemple:

```
comptador = 0
llista2 = [2, 3, 3, 4, 5]
while(comptador < len(llista2)):
    if llista2[comptador] == 4:
```

```
        break
    llista2[comptador] += 1
    comptador +=1
```

Per tant, la llista quedaria així:

```
llista2 = [2, 4, 4, 4, 5]
```

---

2. Loop for

El loop *for* s'utilitza quan es vol repetir una secció de code un nombre n de vegades.

Exemple:

```
llista = [1,2,3,4]
for valor in list:
    valor+=1
```

El resultat seria que sumaria 1 a tots els elements de la llista.

Una altre manera de fer-ho seria la següent:

```
llista = [1,2,3,4]
for i in range(len(llista)):
    llista[i]+=1
```

Aquesta exemple fa el mateix que l'anterior però utilitzant *range()*, que farà que i incrementi de un en un fins arribar l'últim index de la llista.