Introducció a la Programació
Grau en Estadística
Universitat Autònoma de Barcelona

Sessió 4:
Llistes i Bucles for (II)

Vicenç Soler

Índex

1. Introducció

A la sessió anterior hem vist les llistes i una altra instrucció per a fer bucles iteratius: la instrucció "for". En aquesta sessió es veurà més sobre aquests temes.

2. Break i continue

En els bucles "for" i "while" podem afegir altres condicionants que milloren la versatilitat en la programació. En aquest sentit tenim 3 instruccions:

- **break**: Quan s'executa aquesta instrucció, acaba el bucle i continua l'execució del programa en les línies posteriors al bucle.
- **continue**: Quan s'executa aquesta instrucció, continua el bucle en la següent iteració que li toqui, sense executar les línies que hi ha per sota del "continue".
- **else**: Pertany al "for" o al "while" i es posa al final del bucle. El codi del "else" s'executa quan el bucle ha finalitzat de manera natural, és a dir, sense executar "**break**".

Exemple: Volem trobar quin és el primer nombre parell d'un array, i acabar quan el trobem:

```
1.  for  x in v:
2.      if x%2==0:          #L'operador % calcula el mòdul. Si el reste de dividir per 2 és 0, és parell
3.        print("Primer parell trobat:",x)
4.        break             #una vegada trobat, acabem el bucle
```

Notar que si no posem el "break", el programa llistaria tots els nombres parells de l'array.

Ara imaginem que volem fer-ho només als nombres positius, i descartar si el parell és negatiu. En aquest cas ho podem fer amb un "and" o bé a la instrucció "continue":

Fent servir "and"

```
1.  for x in v:
2.    if x>=0 and x%2==0:
3.      print("Primer parell trobat:",x)
4.      break
5.
```

Fent sevir "continue"

```
1.  for x in v:
2.    if x<=0:
3.      continue  #torna a la 1 fa la següent iteració
4.    if x%2==0:
5.      print("Primer parell trobat:",x)
6.      break
```

Si ara volguéssim col·locar un missatge en cas que no hagi trobat cap nombre parell:

```
1.  for x in v:
2.    if x<=0:
3.      continue  #torna a la 1 fa la següent iteració
4.    if x%2==0:
5.      print("Primer parell trobat:",x)
6.      break
7.  else:       #pertany al "for" i, per tant, el situem amb la mateixa indentació-
8.      print("No s'ha trobat cap parell") #Si ha trobat un parell, sortirà amb el "break" i no executa l'else
```

Exercici proposat 1: Repetir el bucle anterior amb "while".

3. Llistes

Funció enumerate

Quan volem iterar en un "for", ho podem fer de 2 maneres: agafant els elements directament, o trobar-los via els seus índexs. Però si volem obtenir també l'índex, ho podem fer via la funció **enumerate**. Un exemple de les 3 maneres:

For de valors	For d'índexs	For de valors i índexs
1. for x in v: 2. print(x)	1. for i in range(len(v)): 2. print(v[i]) 3.	1. for i,x in **enumerate**(v): 2. print("index:",i) 3. print("valor:",x)

Operador in

També hem vist com poder trobar un element a una llista, i que normalment ho hem fet amb un bucle. En aquest cas, per a simplificar, també tenim a la nostra disposició l'operador **in**. Per exemple, si volem saber si el nombre 3 està en una llista, podem fer:

```
if 3 in v:
   print("Valor 3 trobat")
```

Subllistes (subarrays)

Ens permeten agafar alguns elements de la llista i copiar-los en una llista nova. Ho mostrarem amb diferents exemples. Suposem que tenim la llista:

```
v=[10,20,30,40,50]

v2=v[1:]   -> [20,30,40,50]  #des de l'índex 1 fins al final

v3=v[1:4]   -> [20,30,40]    #des de l'índex 1 fins al 4 exclòs (des de l'1 fins al3)

v4=v[-1]    ->  50   #quan és negatiu agafa pel final (-1=últim, -2=penúltim, etc.).
                     Equival a: v[len(v)-1]

v5=v[-2]    -> 40
```

Operacions bàsiques amb llistes

Afegir elements: Tenim la possibilitat de fer anar les funcions **append** o **insert** per a incrementar el tamany de la llista afegint nous elements. Exemple:

```
v.append(60)    #afegeix el 60 al final de la llista incrementant el tamany en 1
                (ara de tamany 6).
```

Eliminar elements: Tenim la possibilitat de fer anar les funcions **pop, remove** o **del** per a decrementar el tamany de la llista eliminant elements. Exemple:

```
v.pop()         #elimina el darrer element de la llista, decrementant el tamany en 1.
```

Unir llistes: Ho podem fer amb l'operador '**+**':

```
v=[10,20,30,40,50]
w=[100,200]
z=v+w      -> [10,20,30,40,50,100,200]
```

Organització a memòria

Totes les variables es guarden a memòria. Per a simplificar, suposarem que cada dada es guarda en una casella diferent. Per tant, les variables que només guarden una dada només necessiten una casella per a guardar-la, i les llistes necessiten vàries caselles

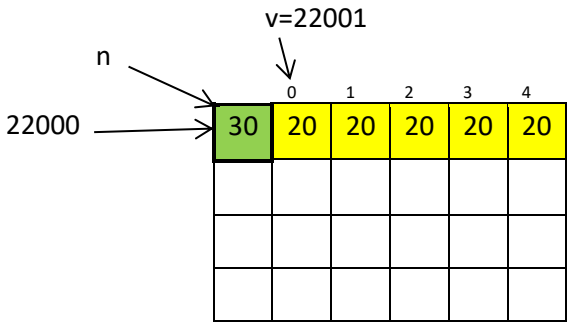
En cadascun dels casos el **manager de memòria** que Python fa servir ha reservat en un lloc de la memòria espai per a guardar el que ha necessitat. Cadascun d'aquests espais els referencia via l'adreça de memòria on estan guardats i es preocupa que cap altre variable ni res pugui tenir ni la mateixa adreça ni pugui escriure a la mateixa posició.

Anem a veure com es comporta la memòria en el cas de les llistes

Exemple 1: una variable i una llista.

```
n=30
v=[20]*5
```

el manager ha reservat espai per a guardar 6 enters, un correspon a la variable 'n' i els altres 5 a l'array 'v'. Com és el manager qui decideix quines adreces es reserven, suposem que comença la reserva per l'adreça de memòria 22000[1]. Gràficament tindríem:



On veiem que el 30 l'hem guardat a la 'n' i que la 'n' està a l'adreça de memòria 22000. La següent adreça, la 22001, és on començaria l'array 'v', de 5 posicions (de 0 a 4 marcades a dalt), que sempre tenen caselles contigües.

I així continuaríem a mesura que anem creant noves variables (al programa principal, a dins de funcions, etc.).

[1] Les adreces les escollim aleatòriament. Hem escolit la 22000 com podría ser qualsevol altra.

Exemple 2: una variable i dues llistes a la mateixa adreça de memòria.

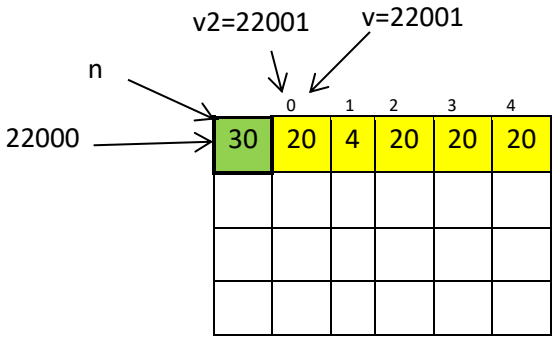
Si fem:

```
v2=v      #v2 no és una nova llista, sinó que és la mateixa llista que v, ja que amb aquesta
          operació el que hem fet és assignar l'adreça de memòria.

v[1]=4    #Hem assignat el 4 a la posició 1

print(v2) #si ara mostrem la llista v2 veurem que dona [20,4,20,20,20], cosa que demostra
          que 'v' i 'v2' són la mateixa llista, és a dir, 'v' i 'v2' apunten a la mateixa adreça
          de memòria
```

Gràficament tindríem:



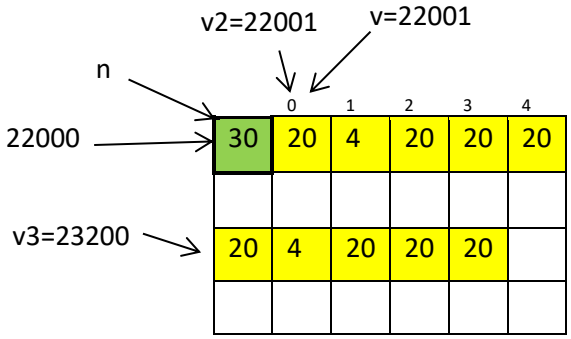
De fet, quan es fa v[1], en veritat el Python el que fa és buscar la casella de memòria on està 'v' i sumar-li 1 casella, v[4] és l'adreça de 'v' + 4 caselles, etc. Per aquesta raó l'índex comença per 0.

Exemple 3: una variable i dues llistes a diferent adreça de memòria.

Però si el que volíem fer amb "v2=v" era crear una nova còpia de 'v', el que hem de fer és crear aquesta còpia de manera explícita. Per a això, tenim, al menys, 3 maneres de fer-ho:

```
v3 = list(v)      #amb la funció list() creem una llista a partir d'una altra llista
v3 = v.copy()     #fem anar al funció copy() de les llistes
v3 = v[:]         #creem una subllista corresponent a tots els elements de 'v'.
```

Gràficament tindríem:



Funcions típiques de llistes

Per a treballar amb les llistes, tenim a la nostra disposició una sèrie de funcions. A continuació es mostra una llista:

Funció	Descripció
append()	Afegeix un element al final de la llista
clear()	Elimina tots els elements de la llista
copy()	Retorna una còpia de la llista
count()	Retorna el nombre d'elements del valor passat com a paràmetre
extend()	Afegeix els elements d'una altra llista al final de la llista.
index()	Retorna l'índex del primer element del valor passat per paràmetre.
insert()	Afegeix un element a la posició especificada
pop()	Elimina l'element del final de la llista (si no rep paràmetre) o el de la posició especificada (si l'hi passes com a paràmetre).
remove()	Elimina l'element amb el valor especificat
reverse()	Inverteix l'ordre de la llista
sort()	Ordena la llista

Referència: https://www.w3schools.com/python/python_lists.asp

Exercicis proposats per a implementar-los fent servir les funcions disponibles:

Exercici proposat 4 : Fent servir la funció "sort", ordena un array de menor a major.

Exercici proposat 5 : Fer una funció anomenada "inserta" que rebi una llista i un nombre, i saps que la llista està ordenada incrementalment, com a l'exercici anterior. La idea és inserir a la llista l'element que ens passen per paràmetre, però mantenint l'ordre. S'ha de fer servir la funció "insert" de les llistes, però no es pot fer servir de nou "sort".

Exercici proposat 6: Fer una funció que elimini un element concret d'una llista, però sense fer servir "remove". Has de fer servir "pop". La funció rebrà com a paràmetres la llista el nombre a eliminar. Si n'hi ha més d'un, només cal eliminar el primer que es trobi.

a. La clau està en identificar la posició de l'element a trobar i passar-li a "pop".

b. Fer una variació per a què elimini totes les posicions que continguin aquell valor, en cas que estigui més d'una vegada a la llista.

4. Més info sobre Python

Instrucció pass

A vegades tenim algun 'if' o algun bucle en què ara mateix no sabem què posar i volem deixar el contingut en blanc i afegir-lo més endavant. En Python aquest espai no es pot deixar en blanc i s'ha de posar alguna cosa. En aquest cas tenim la instrucció "pass" que, bàsicament, no fa res, simplement omple un espai buit. Un exemple seria:

```
1. if i>10:
2.     pass
3. else:
4.     print("i no és més gran que 10")
```

Intercanvi de variables

Imaginem que volem intercanviar el valor de 2 variables (o 2 posicions d'una llista). En aquest cas, la tècnica habitual és fer servir una tercera variable de suport que et permeti fer l'intercanvi. En el següent exemple veurem com no s'hauria de fer (esquerra) i com s'hauria de fer aquest intercanvi per a què funcionés:

Intercanvi incorrecte

```
1. A = 1   #al final volem B=1 i A=2
2. B = 2
3. A = B   #si fem això, hem perdut el valor de A, i ara
           mateix ambdues variables valen 2.
4. B = A   #aquest pas ja no serveix de res, ja que
           ambdues valen 2.
5. print(A)  #imprimirà 2 en les dues línies.
6. print(B)
```

Intercanvi fent servir una tercera variable (aux)

```
1. A = 1     #al final volem B=1 i A=2
2. B = 2
3. aux = A   #guardem el valor de A en 'aux'.
4. A = B     #com hem guardat el valor de A abans,
             ara podem posar el valor de B en A.
5. B = aux   #recuperem el valor que hi havia en A i
             el posem a B
6. print(A)  #mostrarà valors intercanviats: A=2,B=1.
7. print(B)
```

En Python se'ns permet de fer assignacions múltiples en una sola línia, Per exemple:

```
A, B=1, 2    #A=1 i B=2
```

En aquest cas, un intercanvi de 2 variables es pot fer en una sola línia, fent:

```
A, B=B, A    #A=1 i B=2
```

I si volguéssim intercanviar, per exemple, les posicions 0 i 4 d'un array de 5 posicions:

```
v=[10,20,30,40,50]
v[0],v[4]=v[4],v[1]        #intercanviem les 2 posicions
print(v)   -> [50,20,30,40,10]
```

Per tant, en Python tenim la possibilitat de fer l'intercanvi de 2 variables en una sola línia. Escolliu la més us vingui millor: la tradicional (a dalt, fent servir una variable auxiliar) o la específica de Python (a sota, en una sola línia).

Exercici proposat 7: Volem invertir una llista (donar-li la volta).Fer-ho fent la inversió en una segona llista i també que el resultat quedi a la mateixa llista.

5. Solucions als exercicis proposats

Exercici proposat 1: Repetir el bucle anterior amb "while".

```
1.  i = 0
2.  while i<len(v):
3.    if v[i]<=0:
4.       continue  #torna a la línia 2 fa la següent iteració
5.    if v[i]%2==0:
6.       print("Primer parell trobat:",v[i])
7.       break
8.  else:       #pertany al "while" i, per tant, el situem amb la mateixa indentació.
9.       print("No s'ha trobat cap parell")  #Si ha trobat un parell, sortirà amb el "break" i no executa el "else"
```

Exercici proposat 2: Crea un array de 5 posicions i posa-li valors. Busca en aquest array si el nombre 3 es troba en un dels valors. Mostra un missatge en cas afirmatiu o negatiu.

L'exercici el resoldrem de 3 maneres diferents:

1) Amb l'operador "in" de les llistes.
2) Emulant l'operador "in" amb una funció que ens faci la mateixa tasca.
3) Amb un bucle 'for' i el seu 'else'.

1) Amb l'operador "in" de les llistes.

```
1.  v=[10,20,30,40,50]
2.  if 3 in v:
3.    print("trobat")
4.  else:
5.    print("no trobat")
```

2) Emulant l'operador "in" amb una funció que ens faci la mateixa tasca.
Imaginem que desconeixem l'existència d'aquest operador 'in'. Aleshores hauríem de fer una funció que ens fes la mateixa tasca i ens retornés True o False en cas que el trobi o no el trobi:

```
1.  def troba3(v):
2.     for x in v:
3.        if x==3:
4.           return True
5.     return False
6.
7.  v=[10,20,30,40,50]
8.  if troba3(v):
9.    print("trobat")
10. else:
11.   print("no trobat")
```

3) Amb un bucle 'for' i el seu 'else'.

```
for x in v:
    if x==3:
        print("trobat")
        break
else:                   #només executa el "else" si no ha sortit per "break", i surt per "break" si el troba.
    print("no trobat")
```

Exercici proposat 3: Ara no volem saber si el nombre 3 s'hi troba, sinó que ens mostri per pantalla l'índex que ocupa el valor 3 dins de l'array. Si no es troba, que mostri un missatge "No es troba".

1) Si només volem mostrar el 1er trobat

```
for i in range(len(v)):
    if v[i]==3:
        print("Trobat a la pos:",i)
        break                   #quan en trobem un 3, sortim
else:
    print("No trobat")          #només executa aquesta línia si no ha sortit per break
```

2) Si volem mostrar-los tots

En aquest exercici la dificultat rau en el fet de mostrar el "No trobat" si i només si no en troba cap, és a dir, al final del bucle. Com hem tret el "break" perquè no volem parar quan en trobi un, no podem fer servir el "else" del bucle. En aquest cas, farem servir una variable booleana (valors True i False) que ens permetrà saber si hem trobat, al menys, un 3.

Inicialitzem la variable "haEntrat" a False abans d'iniciar el bucle i, tant si troba un 3 com si en troba 100, la variable canvia a valor True. Quan el bucle acaba pregunto quan val aquesta variable i sabré si ha entrat o no (valdrà False si no ha trobat cap 3).

Per tant, la variable "haEntrat" ens serveix com a marca per a saber si ha entrat dins del "if v[i]==3:" o no.

```
haEntrat=False
for i in range(len(v)):
    if v[i]==3:
        print("Trobat a la pos:",i)
        haEntrat=True

if haEntrat==False:
    print("No trobat")
```

Exercici proposat 4: Fent servir la funció "sort", ordena un array de menor a major.

Nota: Per a executar una funció d'una llista, s'escriu **llista.funció(paràmetres)**

```
1. v=[20,40,50,10,30]
2. v.sort()
3. print(v)
```

Exercici proposat 5: Fer una funció anomenada "inserta" que rebi una llista i un nombre, i saps que la llista està ordenada incrementalment, com a l'exercici anterior. La idea és inserir a la llista l'element que ens passen per paràmetre, però mantenint l'ordre. S'ha de fer servir la funció "insert" de les llistes, però no es pot fer servir de nou "sort".

```
1. def inserta (llista,num):
2.     for i,val in enumerate(llista):
3.         if val>num:
4.             llista.insert(i,num)
5.             break
6.     else:
7.         llista.append(num)        #també funcionaria:   llista.insert(len(llista),num)
8.
9.     return llista
```

Nota: en aquest exercici fem un "return llista", però veurem més endavant que no caldrà fer un "return" per a obtenir la llista des d'on cridem a la funció.

Exercici proposat 6: Fer una funció que elimini un element concret d'una llista, però sense fer servir "remove". Has de fer servir "pop". La funció rebrà com a paràmetres la llista el nombre a eliminar. Si n'hi ha més d'un, només cal eliminar el primer que es trobi.

a. La clau està en identificar la posició de l'element a trobar i passar-li a "pop".

```
1. def elimina(v,num):
2.     ind = v.index(40)                    #compte!: La funció "index" dona error si no troba el valor!
3.     v.pop(ind)        #si "pop" rep un índex, elimina l'element que està en aquest índex. Si no, elimina l'últim.
4.     return v
5.
6. v=[10,20,30,40,50]
7. v=elimina(v,40)
8. print(v)   -> [10, 20, 30, 50]
```

b. Fer una variació per a què elimini totes les posicions que continguin aquell valor, en cas que estigui més d'una vegada a la llista.

   Com "index" pot donar error si no troba un valor, i encara no sabem controlar els errors, proposarem fer-ho per 3 vies:

   1) For amb range (veurem que no funciona)
   2) Amb while
   3) Farem servir la funció count per a saber si ens queda algun valor per a eliminar i index per a saber on està:

1) For amb range (veurem que no funciona)

```
1.  def elimina(v,num):
2.      for i in range(len(v)):
3.          if v[i]==40:
4.              v.pop(i)
5.      return v
6.
7.  v=[10,20,30,40,50]
8.  v=elimina(v,40)
9.  print(v)
```

El problema d'aquesta implementació és que la funció "range" només s'executa a l'inici del bucle i no es recalcula a mida que el bucle avança. En aquest cas "range" retorna (0,1,2,3,4), però com la llista l'anem modificant a mesura que el bucle avança, és possible que els darrers índexs ja no es trobin a l'array. Per exemple, si eliminem un element, el darrer índex haurà canviat a 3 (hem reduït en 1 el tamany), però el "for" voldrà anar a veure si la posició 4 existeix, ja que (0,1,2,3,4) es va generar a l'inici i donarà error.

2) Amb while: sí que funciona ja que ofereix suficient versatilitat com per a evitar el problema de la implementació amb "for" anterior:

```
1.  def elimina(v,num):
2.      i=0
3.      while i<len(v) :    #com recalculem "len" cada vegada, ens adaptem als canvis de tamany fets per la "pop"
4.          if v[i]==num:
5.              v.pop(i)
6.          else:
7.              i+=1        #Si eliminem un no cal incrementar i, ja que els de la dreta es mouen 1 posició a l'esquerra
8.
9.  v=[10,20,40,40,50]
10. v=elimina(v,40)
11. print(v)   ->   [10,20,50]
```

3) Farem servir la funció **count** per a saber si ens queda algun valor per a eliminar i **index** per a saber on està:

```
1.  def elimina(v,num):
2.      while True:         #bucle infiinit, que trencarem quan no trobem més vegades "num" (amb el break)
3.          n=v.count(num)          #mirem si en queda algun a la llista per a eliminar
4.          if n>0:                 #si en queda algun, busquem l'índex i l'eliminem
5.              i=v.index(num)
6.              v.pop(i)
7.          else:                   # sinó en trobem cap, ja hem acabat i sotim del bucle
8.              break
9.
10. v=[10,20,40,40,50]
11. v=elimina(v,40)
12. print(v)   ->   [10,20,50]
```

En aquest cas hem muntat un bucle infinit ("while True:") ja que la condició d'aturada del bucle la preferim mirar enmig del bucle, no en el moment del "while".

Exercici proposat 7: Volem invertir una llista (donar-li la volta). Fer-ho fent la inversió en una segona llista i també que el resultat quedi a la mateixa llista.

Intercanvi amb variable auxiliar

```
1. v=[10,20,30,40,50]
2. j=len(v)-1  #índex per a decrementar
3. for i in range(len(v)//2):
4.     aux=v[i]    #fem l'intercanvi
5.     v[i]=v[j]
6.     v[j]=aux
7.     j=j-1
8. print(v)
```

Intercanvi directe amb Python

```
1. v=[10,20,30,40,50]
2. j=len(v)-1  #índex per a decrementar
3. for i in range(len(v)//2):
4.     v[i],v[j]=v[j],v[i]  #intercanvi en 1 línia
5.     j=j-1
6. print(v)
7.
```

Només hem de fer l'intercanvi fins a la meitat d'índexs. Sinó, estarem fent l'intercanvi 2 vegades i la llista ens quedaria igual. Per això fem el 'for' fins a la meitat. Com el 'range' necessita un enter, fem la divisió entera amb '//' . Així, si 'v' tingués tamany 5, només hauríem d'intercanviar els 2 primers amb els 2 darrers i el del mig no caldria intercanviar-lo amb res, ja que es quedaria al mateix lloc.