Introducció a la Programació
Grau en Estadística
Universitat Autònoma de Barcelona

Sessió 3:
Llistes i Bucles for

Vicenç Soler

Índex

1. Introducció

A la sessió anterior hem vist les instruccions per a fer bucles iteratius "while". En aquesta sessió veurem la instrucció "for". A més, també veurem els llistes.

2. Llistes (arrays)

Fins ara hem vist que les variables ens ajuden a guardar valors. Però i si necessitéssim guardar 30 valors, o 300? El que hauríem de fer és declarar les 30 o 300 variables per a guardar aquests valors. Per a no haver de fer-ho, existeixen les llistes[1].

Les **llistes** (en informàtica fem servir el terme "arrays") permeten guardar un nombre determinat de dades en una sola variable. La seva sintaxi és englobant les dades entre corchetes [...] i separant els elements per comes.

Les llistes han de tenir un tamany predefinit, que l'establim quan les creem. La sintaxi és:

```
Nom_llista=[valor_inicial]*tamany
```

Per exemple, si volem crear una llista de 50 valors, seria:

```
v=[None]*50    #creem una llista de 50 elements, de moment buits.
v=[0]*50       #creem una llista de 50 zeros.
```

També podem crear una llista amb diferents valors. Un exemple d'una llista dels 5 primers nombres primers:

```
v=[2,3,5,7,11]
```

En tots els casos, sempre podem preguntar a una llista quin és el seu tamany fent servir la funció **len(llista)**. En l'exemple de l'esquerra "len(v)" retornaria un valor de 5.

La instrucció anterior crea 5 espais de memòria per a guardar 5 nombres (la llista és de tamany 5), i s'accedeix a cadascun d'ells mitjançant un **nombre índex**. L'índex del primer element és el 0.

Per exemple:

```
v[3] = 7
```

Guarda el nombre 7 a la posició 3 de l'índex.

[1] En Python existeixen altres estructures per a guardar diversos valors, com les tupes, els conjunts o els diccionaris, que veurem més endavant. En aquesta secció ens centrarem en les llistes.

En Python, l'índex del **primer element és el 0** i el **darrer és tamany-1** (4, en aquest darrer cas; 49, en el cas de la llista anterior de 50 elements). Per tant, si volguéssim guardar els 5 enters, fent servir l'ìndex, ho faríem així:

v[0] = 2
v[1] = 3
v[2] = 5
v[3] = 7
v[4] = 11

Si volem crear una llista amb molts elements (per exemple, 50), la solució ideal seria fent servir una instrucció iterativa (while) que ens repeteixi aquesta acció 50 vegades, canviant l'índex a cada iteració.

Per exemple, suposem que volem guardar els nombres del 0 al 49 dins d'una llista de 50 posicions:

```
1. v=[None]*50          #declarem que v és una llista de tamany 50
2. i=0
3. while  i<50:
4.     v[i] = i
5.     i = i + 1        #no oblidar-se d'incrementar la variable i
```

Si ara volguéssim omplir la llista amb els nombres de l'1 al 50, seria:

```
1. v = [None]*50
2. i = 0
3. while  i<50:
4.     v[i] = i + 1     #en lloc de guardar 'i', guardem 'i+1'
5.     i = i + 1
```

ja que encara podem aprofitar l'índex "i" per a 2 tasques: com a índex i com a base per a proporcionar-nos el valor a guardar desitjat (en aquest cas coincideix que volem guardar exactament el valor de l'índex incrementat en 1, ja que en les posicions 0 a 49, volem guardar els valors 1 a 50).

Si volguéssim guardar els 50 primers nombres parells positius, incloent el 0:

```
1. v=[None]*50
2. i=0
3. while  i<50:
4.     v[i] = i * 2     #generem els parells
5.     i = i + 1
```

A la posició 0, hi guardarem el 0, a la posició 1 hi guardarem el 2,..., a la posició 30 hi guardarem el 60,... i a la darrera posició, la 49, hi guardarem 49*2.

Exercici proposat 1: Volem que en cada posició d'una llista es guardi el valor de l'índex anterior, incrementat en 1. El primer valor de la llista el posem a 1.

```
1. v=[None]*50
2. v[0]=1                #necessitem un valor inicial
3. i = ?
4. while i < ?:
5.     v[i+1] = v[i] + 1
```

Quins valors hem de posar en lloc dels dos interrogants vermells **?**.

Exercici proposat 2: I si la llista de l'exercici proposat anterior fos de tamany 100?

Ara suposem que ja tenim una llista a la qual ja hem introduït totes les dades. Per exemple, si volguéssim calcular la suma de tots els elements, seria:

```
1. #considerem que la llista és de tamany 5
2. suma = 0
3. i = 0
4. while i<len(v):              #podem fer servir 5 o len(v)
5.     suma = suma + v[i]
6.     i = i + 1
7. print("La suma és:",suma)
```

Fixem-nos que en tots els exemples proposats de llistes, pràcticament ens dóna igual si el tamany de la llista és de 5, 50 o 500, ja que ens consta d'escriure el mateix, amb el mateix nombre de línies i instruccions. És aquí on veiem la potència dels iteradors (la variable 'i', en aquest cas). Per exemple, en l'exercici anterior, si la llista fos de 500 posicions, no hauríem de canviar res de la línia 4, i ens calcularia la suma dels 500 nombres.

Exercici proposat 3: Omple una llista de 5 posicions amb 5 nombres enters entrats per teclat i mostra la seva suma.

Exercici proposat 4: Omple una llista de tamany 5 amb 5 nombres de tipus "float" (nombres amb decimals) entrats per teclat i mostra la seva suma. Recorda la funció "float" en l'input.

Exercici proposat 5: Omple una llista de tamany 5 amb l'invers dels nombres 1 a 5. És a dir, que la llista contindrà els nombres 1, 1/2, 1/3, 1/4 i 1/5. Quan ho hagis fet, mostra per pantalla cadascun dels nombres i la seva suma.

Exercici proposat 6: Crear una llista de 5 nombres (els que vulguis) i mostrar-la en ordre invers, sense canviar el contingut de la llista.

Exercici proposat 7: Sèrie de Fibonacci. En la sèrie de Fibonacci, cada nombre és la suma dels 2 anteriors. Els 2 primers nombres de la sèrie son 1. L'exercici consisteix en entrar un nombre enter per teclat "n" i mostrar per pantalla els nombres de la sèrie fins al "n"-èssim valor. Per exemple, si s'entra un 6 per teclat, ha de mostrar els 6 primers nombres de la sèrie: 1,1,2,3,5,8. (8 és 3+5, 5 és 2+3, etc.). Feu-ho fent servir una llista per a guardar tots els nombres i també sense guardar-los en cap llista (sense fer servir cap llista).

3. Bucles "for"

La instrucció "while" està pensada per a executar una tasca un nombre indeterminat de vegades, mentre una condició sigui certa.

En canvi, si pensem en executar alguna tasca un nombre concret de vegades, és millor fer servir la instrucció **for**, combinat amb la funció **range**, especialment si fem anar llistes. El format de la instrucció "for" és:

for <variable> in <llista> :

En aquest cas, la instrucció "for" passa per cadascun dels elements de la llista, i a cada iteració posa cada element a la variable que nosaltres especifiquem, i que només servirà per la tasca pròpia del "for".

Un exemple seria:

```
1. v = [10 , 20 , 30]
2. for numero in v:
3.     print(numero)
```

Aquest exemple mostra per pantalla els nombres 10 , 20 i 30. A cada iteració del "for", la variable "numero" pren el valor de cadascun dels elements de la llista.

Si ho volgués fer amb "while":

```
1. v = [10 , 20 , 30]
2. i = 0
3. while i < len(v) :
4.     numero = v[i]
5.     print(numero)          #alternativament, podríem haver fet print(v[i])
6.     i = i + 1
```

La diferència d'aquest exemple anterior entre fer servir el bucle "for" i el "while" és que el "for" no necessita fer anar una variable per a l'index.

En resum:

Bucle "for"	**Bucle "while"**
1. v = [10 , 20 , 30] 2. for numero in v: 3. print(numero)	1. v = [10 , 20 , 30] 2. i = 0 3. while i < len(v) : 4. numero = v[i] 5. print(numero) 6. i = i + 1

Si ara volguéssim mostrar els nombres decrementalment, necessitaríem accedir als índexos i fer-ho com ho fèiem amb el "while". Com un "for" necessita recórrer una llista, hauríem de primer generar una llista amb els índexos. Per a fer això, tenim a la nostra disposició la funció **range**, que genera aquesta llista. Si fem:

range(<tamany>)

genera **una llista d'enters de 0 a tamany -1**, just els índexos d'un array d'aquest tamany.

A continuació es mostren exemples de com s'implementaria incrementalment i decrementalment, fent servir índexos:

1.Incrementalment:

For amb range

```
for i in range(5):
    print(v[i])
```

For amb range i len

```
for i in range(len(v)):    #aprofitem "len"
    print(v[i])
```

Amb while

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

2.Decrementalment:

For amb range

```
for i in range(5):
    print( v[5 - i] )
```

For amb range i len

```
for i in range(len(v)):
    print( v[5-i] )
```

Amb while (completar)

```
i = len(v)
while i >= ? :
    print(v[ ? ])
    i = i - 1
```

Exercici proposat 8: Quins valors haurien d'haver en lloc del ? en l'exercici anterior de la dreta?

Exercici proposat 9: Entreu un nombre enter per teclat. Mostreu per pantalla els nombres des d'aquest nombre entrat fins al 5 fent servir "for". Per exemple, si entrem el 3, hauria de mostrar els nombres 3, 4 i 5. Si entre 6 o superior, no hauria de mostrar res.

Exercici proposat 10: El mateix d'abans però decrementalment fins a 1.

Pista:

range amb 3 paràmetres és**: range(<inicial>,<final>,<step>)**

i "step" servix per a dir-li com ha de generar la seqüència de nombres. Exemples:

- range (1 ,5, 1) -> 1,2,3,4 #igual que fer range(1,5), ja que l'estep per defecte és 1
- range (1 ,9, 2) -> 1,3,5,7 #nombres senars: de 2 en 2 i no arriba al 9

4. Variables comptadores i acumuladores

En general, les variables les podem fer servir per al que nosaltres necessitem. Però d'aquestes, en podem destacar alguns tipus, segons l'ús que se'n faci al programa. Dos d'aquests tipus són:

- **Comptadors**: Fins ara hem treballat amb variables comptadores, que les hem fet servir als bucles "while" per a fer d'índexos, o bé per a obtenir una sèrie de nombres. Aquestes variables les anem incrementant o decrementant (i=i+1 ,o bé i=i-1)
- **Acumuladors**: Les variables acumuladores ens permetran anar emmagatzemant un càlcul parcial. Per exemple, ens ajudaran a calcular una suma.

Posem que necessitem calcular la suma dels valors d'una llista. El següent programa fa aquest càlcul:

```
1. #suposem una llista de 5 elements
2. suma = 0
3. i = 0
4. while i<len(v) :      #Podem posar 5 en lloc de len(v), però sempre és millor fer-ho general
5.             suma = suma + v[i]
6.             i = i + 1
7. print (suma)
```

el que hem vist és que hem fet servir 2 variables: la variable 'i' és un comptador i la variable 'suma' és un acumulador, ja que hi anem acumulant els resultats parcials.

5. Aprofitar les funcions fetes

Quan es programa, acostumem a separar les tasques a fer en diversos mòduls que permetin organitzar millor un programa. Cada mòdul s'encarrega d'una feina en concret i, en el nostre cas, cada mòdul serà una funció.

Imaginem que no tenim a la nostra disposició la funció "sum" de Python (permet sumar els elements d'una llista) y que hem de crear-la nosaltres mateixos, com passa en altres llenguatges de programació. Com suposem que voldrem sumar una llista més d'una vegada, lo ideal és ficar el codi anterior en una funció que permeti sumar una llista (ho farem amb while i for):

```
1. def sumarLlista (v) :
2.   suma = 0
3.   i = 0
4.   while i<len(v) :
5.     suma = suma + v[i]
6.     i = i + 1
7.   return suma
```

```
1. def sumarLlista (v) :
2.   suma = 0
3.   for n in v:
4.     suma = suma + n
5.   return suma
```

Veiem que fent-ho amb el 'for' ens permet tenir un codi més petit, ja que volem recórrer tota la llista i no necessitem fer anar índexos.

Si ara volguéssim fer una funció per a calcular la mitja aritmètica dels nombres d'una llista, no necessitaríem fer un bucle per a sumar-los de nou, ja que podem aprofitar la funció anterior:

```
1.  def sumarLlista (v) :
2.    suma = 0
3.    for n in v:
4.        suma = suma + n
5.    return suma
6.
7.  def mitja (v):
8.     return sumarLlista(v) / len(v)
9.
10. #Programa principal
11. llista = [1,2,3,10,11,12]
12. avg = mitja(llista)          #crido a la meva funció mitja, que cridarà a la meva funció sumarLlista
13. print("La mitja de ",llista," és: ",avg)
```

6. Solucions als exercicis proposats

Exercicis proposats 1 & 2: Volem que en cada posició d'una llista es guardi el valor de l'índex anterior, incrementat en 1. El primer valor de la llista el posem a 1.

```
1.  v=[None]*50
2.  v[0]=1                  #necessitem un valor inicial
3.  i = 0
4.  while i < len(v)-1:
5.      v[i+1] = v[i] + 1
```

Quins valors hem de posar en lloc dels dos interrogants vermells **?**.

Exercici proposat 2: I si la llista de l'exercici proposat anterior fos de tamany 100?

En aquest cas, ho referenciem tot a "len(v)" i no cal pensar si la llista té 50, 100 o un nombre diferent d'elements.

Exercici proposat 3: Omple una llista de 5 posicions amb 5 nombres enters entrats per teclat i mostra la seva suma.

```
1.  v=[None]*5
2.  i = 0
3.  while i < len(v):
4.      v[i] = int(input("entra num ",(i+1),":"))
5.      i = i +1
6.  suma = 0
7.  i = 0                   #tornem a posar el 'i' a 0 per a començar la llista des del principi
8.  while i < len(v):
9.          suma = suma + v[i]
10.         i = i +1
11.
12. print("Suma:",suma)
```

Exercici proposat 4: Omple una llista de tamany 5 amb 5 nombres de tipus "float" (nombres amb decimals) entrats per teclat i mostra la seva suma. Recorda la funció "float" en l'input.

Igual que l'anterior però canviant la funció "int" per "float"

Exercici proposat 5: Omple una llista de tamany 5 amb l'invers dels nombres 1 a 5. És a dir, que la llista contindrà els nombres 1, 1/2, 1/3, 1/4 i 1/5. Quan ho hagis fet, mostra per pantalla cadascun dels nombres i la seva suma.

```
1.  v=[None]*5
2.  i = 0
3.  suma = 0
4.  while i < len(v):
5.      v[i] = 1/(i+1)
6.      sum = suma + v[i]       #ara aprofitem el mateix bucle per a anar sumant
7.      i = i + 1
8.  print("Llista: ",v)
9.  print("Suma:",suma)
```

Exercici proposat 6: Crear una llista de 5 nombres (els que vulguis) i mostrar-la en ordre invers, sense canviar el contingut de la llista.

```
v = [1,3,5,7,9]
i = len(v)-1
while i >= 0:
    print (v[i])
    i = i - 1
```

Exercici proposat 7: Sèrie de Fibonacci. En la sèrie de Fibonacci, cada nombre és la suma dels 2 anteriors. Els 2 primers nombres de la sèrie son 1. L'exercici consisteix en entrar un nombre enter per teclat "n" i mostrar per pantalla els nombres de la sèrie fins al "n"-èssim valor. Per exemple, si s'entra un 6 per teclat, ha de mostrar els 6 primers nombres de la sèrie: 1,1,2,3,5,8. (8 és 3+5, 5 és 2+3, etc.). Feu-ho fent servir una llista per a guardar tots els nombres i també sense guardar-los en cap llista (sense fer servir cap llista).

a) Solució guardant en array:

En aquest cas tenim el dilema de saber quin tamany d'array fem, ja que ha de ser un nombre fixe (fins que no aprenguem a fer arrays amb un tamany variable). El que usualment es fa és pensar en un tamany màxim i només fer servir les posicions necessàries. Lo important és no fer servir més posicions de les que hem reservat, ja que sinó incorrerem en error.

```
Fib=[None]*50  #reservem un tamany 50 com a màxim

maxim=int(input("Quants nombres mostrem?"))

#els 2 primers nombres de la sèrie són fixos
fib[0]=1
fib[1]=1

#Primer omplim la sèrie i després la mostrarem.
#Comencem amb l'índex 2 ja que el 0 i l'1 ja els tenim.
i = 2
while i<màxim :
  v[i] = v[i-1] + v[i-2]
  i = i + 1


#ara els mostrem per pantalla
print("Els ",maxim," primers nombres de la sèrie de Fibonacci són:")
print(v)
```

Questió: Com ho faries per a comprovar que el nombre "màxim" que t'entren no sigui superior a 50? En aquest cas, si fos superior a 50, no hauríem de mostrar la sèrie i sí un missatge del tipus "El nombre que has entrat és erroni ja que és superior a 50".

b) Solució sense fer servir un array:

En aquest cas, com no fem servir array, no posarem límit al nombre d'elements de la sèrie a mostrar. Ara farem servir 2 variables auxiliars que ens serviran per a emmagatzemar el 2 nombres anteriors de la sèrie ("aux_1" per l'anterior i "aux_2" per a 2 anteriors). Per exemple, en cas que es volgués generar el 6è element de la sèrie (8), és la suma dels 2 anteriors: 3 i 5. En aquest cas "aux_1" valdrà 5 i "aux_2" valdrà 3.

Variables que farem servir:

- i , maxim
- aux_1, int aux_2 //2 variables que emmagatzemen els 2 darrers valors de la sèrie.
- aux_actual //servirà per a emmagatzemar l'actual

```
1.  maxim=int(input("Quants nombres mostrem?"))

    # els 2 primers nombres de la sèrie són fixos
2.  print("1 ")
3.  print("1 ")
4.  aux_1 =1
5.  aux_2 = 1   #aux_1 i aux_2, 2 variables que emmagatzemen els 2 darrers valors de la sèrie.

    # Ara mostrarem directament la sèrie.
    # Comencem amb l'índex 3 ja que l'1 i el 2 ja els tenim, com abans.
    # Veiem que ara hem decidit començar per l'índex 1, per a comoditat nostra
    # i acabem amb i<=màxim ja que ara comencem per l'1.
6.  i = 3
7.  while i<=màxim:
8.      aux_actual=aux_1 + aux_2
9.      print (aux_actual)
10.     aux_2=aux_1
11.     aux_1=aux_actual
12.     i = i +1
13.
```

Fixem-nos en el mètode utilitzat, ja que acabem de pujar un nivell en quant a programació. La línia 21 està clara, ja que calculem el proper element de la sèrie, que després mostrem per pantalla en la 22. Però la clau que funcioni la iteració està en les línies 23 i 24. Fem-ho amb un exemple. Suposem que ara ens trobem que mostrem l'element 3er de la sèrie (2), que és la suma dels dos primers (1 i 1). En la següent iteració voldrem mostrar la suma del 2 i de l'1 (3) i això només ho podem fer si actualitzem les variables "aux_1" i "aux_2" als valors corresponents. Per tant, una vegada calculats el valor actual (2), hem de fer que a la següent iteració "aux_1" valgui 2 i "aux_2" valgui 1, per a poder generar el 4rt nombre de la sèrie (3).

Veiem-ho en un gràfic:

Iteració	aux_2	aux_1	aux_actual
i=3	**1**	**1**	**2**
i=4	**1**	**2**	**3**
i=5	**2**	**3**	**5**
i=6	**3**	**5**	**8**
...	...	...	...

Fixem-nos que, a cada iteració (fila), per a calcular el valor actual, necessitem que l'actual anterior sigui el "aux_1" i l'anterior "aux_1" sigui el "aux_2", ja que la sèrie va avançant. Per a implementar-ho, hem proposat:

```
aux_2=aux_1
aux_1=aux_actual
```

És a dir, que després de calcular i mostrar per pantalla el valor actual de la sèrie, actualitzi els 2 valors anteriors de la sèrie: un el que acaba de calcular i l'altre el desplaci una posició cap a enrere.

Qüestió: I si canviem d'ordre aquestes 2 instruccions, funcionaria? Doncs la resposta és NO. Posem el cas que estem en la transició entre i=5 i i=6. Després de mostrar "5" per pantalla, anem a actualitzar els valors de "aux_1" i "aux_2" fent servir:

```
aux_1=aux_actual
aux_2=aux_1
```

Si fem això "aux_1" valdrà 5 (lo que acabem de mostrar per pantalla, el valor de aux_actual) i "aux_2" valdrà també 5, ja que acabem d'assignar 5 a "aux_1". Això passa perquè abans de modificar el valor de "aux1" necessitem moure'l a "aux_2". Si ho fem a l'inrevés, el valor de "aux_1" queda modificat i el programa ja no fa el que desitgem.

Exercici proposat 8: Quins valors haurien d'haver en lloc del **?** en l'exercici anterior de la dreta?

```
1. i = len(v)
2. while i >= 1 :
3.     print(v[ i-1 ])
4.     i = i - 1
```

Exercici proposat 9: Entreu un nombre enter per teclat. Mostreu per pantalla els nombres des d'aquest nombre entrat fins al 5 fent servir "for". Per exemple, si entrem el 3, hauria de mostrar els nombres 3, 4 i 5. Si entre 6 o superior, no hauria de mostrar res.

```
1. numero = int(input("Entra nombre:"))
2. for i in range(numero,5+1) :          #range amb 2 paràmetres és: range(<inicial>,<final>)
3.     print(i)
```

Exercici proposat 10: El mateix d'abans però decremental fins a 1.

```
1. numero = int(input("Entra nombre:"))
2. for i in range(5,numero-1,-1) :  #range amb 3 paràmetres és: range(<inicial>,<final>,<step>)
3.     print(i)
```

En aquest cas, amb el "range" de 3 paràmetres li podem especificar que generi els nombres decreixentment, amb el tercer paràmetre.