1003546

Introducció a la Programació
Grau en Estadística

Sessió2: Bucles while i Funcions

Vicenç Soler

Índex

1. Introducció

A la darrera sessió vàrem iniciar-nos en la programació en Python. Vàrem veure com és la estructura bàsica d'un programa en Python i com executar-lo per a veure el seu resultat.

També vàrem veure com es declaraven variables per a guardar, en aquest cas, nombres enters, i com guardàvem valors o resultats de càlculs. També vàrem veure com es mostrava text per pantalla.

2. Operadors

A continuació llistem els operadors aritmètics, de comparació i aritmètics que té C:

Operadors de comparació

Nom de l'operador	Sintaxi
Menor que	`a < b`
Menor o igual que	`a <= b`
Major que	`a > b`
Major o igual que	`a >= b`
Diferent que	`a != b`
Igual que	`a == b`

Operadors lògics

Nom de l'operador	Sintaxi
Negació	`not a`
AND	`a and b`
OR	`a or b`

Operador d'assignació

Nom de l'operador	Sintaxi
Assignació de valor	=

Operadors aritmètics

Nom de l'operador	Sintaxi
Suma, resta, producte, divisió	+, -, *, /
Divisió entera	//
Acumulació	+= , -= ,*= , /=
Mòdul	%

Anem a aplicar els operadors lògics. Ho farem amb la instrucció if.

Fins ara, tenim clar que la instrucció:

```
if  numero>6 :
    print(numero)
```

Fa que només es mostri el "numero" per pantalla si aquest "numero" és més gran que 6 (recordeu que la condició de la instrucció "if" sempre acaba amb ":"). Però, i si volem preguntar per més d'una condició? Per exemple, voldríem preguntar si "numero"=>3 i "numero"<=5, hem de fer:

```
if numero>=3 and numero<=5 :
```

L'operador AND: fa que només executi el codi que hi ha dins de l'"if" (el que està indentat a dins de l'"if") si és cert que "numero">=3 i "numero"<=5. Per tant, només entrarà dins del "if" si numero té els valors 3, 4 o bé 5 (suposant que "numero" és enter). Si volguéssim aplicar més condicions, posarem més "and" a la pregunta de l'"if".

Compte a vegades amb algunes expressions amb AND. Per exemple, si volem executar un codi si numero no està entre 3 i 5, podem incórrer en l'error d'expressions com:

```
if numero<=3 and numero>=5 :
```

que mai seran certes, ja que no hi ha cap nombre que sigui a la vegada menor a 3 i major a 5. En aquest cas, hauríem d'haver fet servir un operador OR en lloc de l'AND.

Operador OR: Si ara volem preguntar si una de les 2 condicions és certa, farem servir **l'operador OR:**

```
if  numero>=3 or numero<=5 :
```

En aquest cas, entrarà a executar el codi de l'"if" (l'indentat) si "numero">=3 o "numero"<=5. Fixem-nos que, en aquest cas, sempre serà certa, ja que tots els nombres enters compleixen una

de les 2 condicions. Per exemple, si "numero"=6, la 1ª condició serà certa, i si "numero"=-20, la 2ª condició és certa.

Un exemple diferent podria ser:

```
if  numero<=0 or numero >=5 :
```

En aquest cas, si "numero" val 1, 2, 3 o 4 no es compleix la condició. En altre cas, sí.

També tenim altres operadors de comparació:

Operador de comparació d'igualtat (==):

```
if  numero==3 :
```

En aquest cas, compara si "numero" té el valor 3. Anem amb compte de no confondre'ns amb:

~~If numero=3 :~~

Ja que l'operador "=" no és de comparació, sinó d'assignació. El compilador donarà error. Per a comparar fem servir l'operador "==".

Operador de comparació de diferent (!=):

```
if  numero!=3 :
```

En aquest cas, compara si el valor de "numero" és diferent de 3.

Operador NOT : L'operador NOT nega la condició que necessitis.

Per exemple, imaginem que volem comparar si "numero" és 3:

```
if  numero==3 :
```

Que diu que si és cert que "numero" és 3... Però si li col·loquem un NOT al davant:

```
if  not numero==3 :      - >  equival a   ->     if numero != 3 :
```

El que diu és 'si no és cert que "numero" sigui 3 ... '. El que faria el programa, primer, seria comprovar si "numero"==3, i després "negaria" el resultat d'aquesta avaluació. És a dir, que si és CERT que "numero"==3, el resultat amb el NOT, seria que la condició és falsa, i viceversa. Per tant, en aquest cas concret, tenim la casualitat que la expressió de dalt es podria expressar d'aquesta altra manera:

```
if  numero != 3 :          (com hem comentat a dalt)
```

ja que ambdues expressions donen el mateix resultat: només és certa la condició del "if" si "numero" no és 3.

Quina de les dues és millor? La que t'agradi més. En aquest cas, hi ha expressions que es poden formular de maneres diferents i donen el mateix resultat.

Exercici proposat 1: Entrar 2 nombres enters per teclat i mostrar per pantalla si són iguals o diferents.

Exercici proposat 2: Entrar 2 nombres enters per teclat i mostrar per pantalla si els dos són 3.

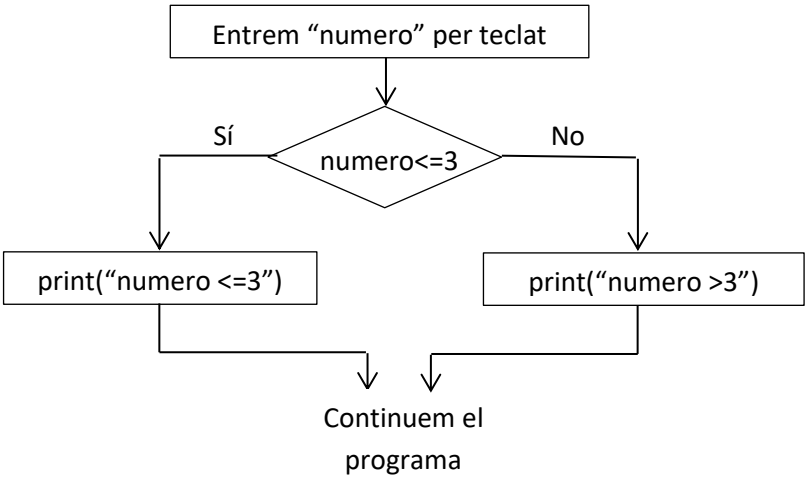
Exercici proposat 3: Entrar 2 nombres enters per teclat i mostrar per pantalla si, al menys un dels 2 nombres és 3.

Exercici proposat 4: Entrar 1 nombre enters per teclat i mostrar per pantalla si és o no diferent de 3.

3. Diagrames de Fluxe

Abans de continuar, fem una parèntesi i expliquem el que són els diagrames de fluxe. És una forma de representar programes que va molt bé per a mostrar gràficament el comportament del mateix. En resum, el **rombe** simbolitza una pregunta i el **rectangle** una ordre. Per tant, el diagrama de fluxe del següent programa podria ser:

```
1. numero = int(input("Entra un num:"))
2. if  numero<=3 :
3.    print ("El nombre ",numero," és més petit o igual que 3")
4. else :
5.    print ("El nombre ",numero," NO és més petit o igual que 3")
```



4. Bucles while

Fins ara, hem vist i practicat estructures de programes lineals. En tot cas, hem pogut decidir si executàvem unes ordres o altres via instruccions "if".

Els bucles ens permeten crear estructures repetitives, on podem executar unes mateixes accions vàries vegades. En aquesta sessió, veurem la instrucció **while** en Python per a executar bucles.

El seu format és:

while condició:

INSTRUCCIONS DEL WHILE

Vol dir que, ***mentre*** *es compleixi una condició, executarà les instruccions que hi hagi dins del "while"*. Veiem uns exemples.

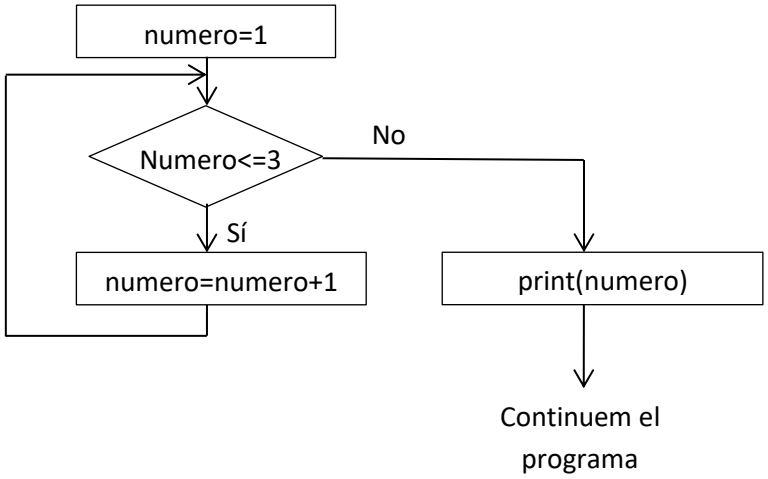
```
1. numero=1
2. while  numero<=3 :
3.     numero = numero +1
4. print (numero)
```

Primer declarem una variable "numero" de tipus entera i després l'hi assignem el valor 1.

A continuació, a la línia 2 comprova si aquesta condició és certa. Com és certa, executa la línia 3, que el que fa és sumar 1 al valor actual de "numero"(1) i el resultat (2) l'assigna a "numero". Per tant, després de la línia 3, el valor de "numero" val 2. La línia 3 és a única que pertany al bucle "while" (en podria haver les que volguéssim) i, quan ha acabat d'executar-la, el que fa el programa és tornar a la línia 2, per a tornar a fer el mateix. Primer comprova si la condició és certa (sí que és certa) i, com ho és, torna a executar la línia 3, incrementant de nou el valor de "numero". Ara, "numero" val 3. De la línia 3 tornem a la 2, comprovem que la condició torna a ser certa i executem la línia 3. Ara "numero" val 4. De la línia 3 tornem a la 2 i ara sí que veiem que la condició del "while" ja no és certa, amb lo que el programa salta al final del bucle (línia 4) i continua executant el programa fins al final. En aquest cas mostrarà per pantalla el valor de "numero", amb lo que mostrarà "4".

Finalment podem afegir que les instruccions de dins del bucle s'ha executat 3 vegades.

Ara anem a expressar aquest bucle en forma de diagrama de fluxe:



Si volguéssim mostrar els números de l'1 al 3:

```
1. numero=1
2. while numero <=3 :
3.     print(numero)
4.     numero = numero +1
5.
```

Fixem-nos que, sense l'increment de "numero", el bucle seria infinit, imprimint sempre "1" per pantalla, ja que "numero" sempre valdria 1.

Exercici proposat 5: Què passaria si intercanviéssim les línies 3 i 4 del programa previ? Què sortiria per pantalla? Intenta fer-ho sense ordinador i, després, pots provar el resultat a l'ordinador.

Nota: La comprovació en un bucle de si continuem les iteracions o no es fa a cada pas que el programa passa per la línia que així ho comprova. Per exemple, suposem aquest codi:

```
1. i = 1
2. while  i<5 :
3.     i = i + 24
4.     print(i)
5.     i = i – 24
6.     i = i +1
7.
```

Mostrarà per pantalla el texte: "25 26 27 28 ". Fixem-nos que inicialment inicialitzem la variable "i" a 1. Això fa que la condició "i<5" sigui certa i, per tant, entrem al bucle i anem a executar la línia 3. En aquesta línia "i" canvia a 25, i la línia 4 mostra aquest "25 " per pantalla. Fixem-nos que, després d'executar la línia 3, la "i" val 25 i, per tant, incompleix la condició del bucle "i<5", però com no es comprova en aquest precís moment el programa continuarà executant la línia 4

com si no passés res. La línia 5 fa que el valor de "i" torni a 1. La línia 6 (incrementem "i") és la darrera del bucle i, després, el programa torni a la línia 2, on comprovem de nou la condició "i<5", la qual és certa i fa que l'ordinador continuï executant per la línia 3, després la 4 i així successivament.

Exercici proposat 6: Mostreu per pantalla els nombres de l'1 al 5, fent servir "while" .

Exercici proposat 7: Mostreu per pantalla els nombres de l'5 al 1, fent servir "while" .

Exercici proposat 8: Entre un nombre enter per pantalla. Mostreu per pantalla els nombres des d'aquest nombre entrat fins al 5, fent servir "while" . Per exemple, si entrem el 3, hauria de mostrar els nombres 3, 4 i 5.

Exercici proposat 9: El mateix d'abans però decremental fins a 1.

5. Funcions

Una funció o mètode és un tros de codi que fa una tasca concreta. La diferència bàsica entre funció i mètode és que la funció retorna un resultat i el mètode no, però la seva definició i ús a Python és quasi idèntica. En la Teoria de la Programació sí que hauríem de distingir entre els 2 termes, però en aquesta assignatura parlarem amb el terme "funció" genèric, i direm funció que retorna valori que no en retorna.

Un exemple de funció podria ser la funció "log" que calcula el logaritme d'un nombre. Aquesta funció rep un nombre i en retorna un altre. Els càlculs que es fan per a proveir aquest resultat es fan dins d'aquella funció que algú ha programat prèviament.

Un exemple de funció que no retona valor (mètode) seria el "print". Aquesta funció, quan se la crida, mostra algun missatge per pantalla, però no retorna cap resultat. Com abans, algú ha declarat el seu codi Python intern que fa que mostri el missatge desitjat per pantalla.

A més de mostrar text per pantalla també podem llegir dades per teclat, que entrem nosaltres al programa. Si "print" és una funció que es fa servir per a poder mostrar informació per pantalla, la funció "input" és la funció que es fa servir per a llegir dades per pantalla.

Ara el que farem és dissenyar la nostra pròpia funció. Per a començar, farem una que mostri per pantalla si un nombre és més gran que 9 o no ho és. Per a fer-ho, l'hem de declarar (definir la seva capçalera i el codi que executarà. La sintaxi és:

def nom_funció (paràmetres) :

codi indentat de la funció ...

return valor

```
1.  def esMajorQue9(num):       # això és la capçalera
2.      if  num>9 :
3.          print("És major que 9")
4.      else :
5.          print("No és major que 9")
6.
```

La línia 1 és la capçalera, i conté:

- El **nom de la funció** és "esMajorQue9", igual que abans hem comentat noms com "print", "input" i "log".
- La funció rep un **paràmetre**, que en aquest cas és un nombre, que guardem a la variable "num". Quan cridarem a aquesta funció hem de passar-li aquest nombre.
- En aquest cas, no ens interessa retornar res. Així que no apliquem **return**.

De les línies 2 a 5 hi ha el codi que executa.

Per tant, quan des d'un lloc del nostre programa volem cridar la nostra funció, escriurem:

```
esMajorQue9(5)
```

i mostrarà el missatge "No és major que 9".

Per tant, el programa sencer, inclosa la funció nova, s'escriuria de la següent manera:

```
1.  def esMajorQue9(num):       # això és la capçalera
2.      if num>9 :
3.          print("És major que 9")
4.      else :
5.          print("No és major que 9")
6.
7.  #inici del programa principal
8.  Numero = int(input("Entra un num.:"))
9.  esMajorQue9(numero)        #cridem a la nostra funció, passant-li un paràmetre
10.
```

Aquest programa espera a que se l'entri un nombre i simplement mostra un missatge de si és >9 o no.

Fixem-nos que ara tenim una funció definida per nosaltres: la "esMajorQue9" , que s'uneix a totes les funcions ja existents al sistema (com print, input, int, etc.) . Recordem que el programa no s'inicia al "def", sinó que la primera instrucció que executa és la línia 8.

Una altre detall important és que la declaració de les funcions han d'estar a dalt de tot, i que al final posem el programa principal.

6. Per què funcions?

Però, per què hauríem de tenir necessitat de posar el if...else dins d'una funció? Si el programa és tan simple com aquest, és evident que no val la pena. Posant el if...else després del "input" ja seria suficient.

Però si volguéssim llegir 2 nombres per pantalla, sí que seria beneficiós el fet de tenir una funció. Veiem la comparativa de fer servir una funció o no:

Sense funció

```
1.  #LLEGIM EL PRIMER NOMBRE
2.  numero = int(input())
3.  if numero>9 :
4.      print("És major que 9")
5.  else :
6.      print("No és major que 9")
7.
8.  #LLEGIM EL SEGON NOMBRE
9.  numero = int(input())
10. if numero>9 :
11.     print("És major que 9")
12. else  :
13.     print("No és major que 9")
14.
```

Amb funció

```
1.  def esMajorQue9(num) :
2.      if num>9 :
3.          print ("És major que 9")
4.      else  :
5.          print("No és major que 9")
6.
7.  #Programa principal
8.  #LLEGIM EL PRIMER NOMBRE
9.  numero = int(input())
10. esMajorQue9(numero)
11.
12. #LLEGIM EL SEGON NOMBRE
13. numero = int(input())
14. esMajorQue9(numero)
15.
```

Ambdós codis faran exactament el mateix i pràcticament faran servir el mateix temps en executar el programa, però veiem en el codi de l'esquerra dupliquem el codi del "if " (el segon és un copy&paste del primer), **cosa que sempre hauríem d'evitar**.

En canvi, el de la dreta és molt més net i clar. En aquest cas ens ha estat útil definir una funció ja que la tasca de comparar amb 9 és una tasca que hem de fer més d'una vegada. I si, en lloc de llegir 2 nombres, són 10 nombres els que llegir, més raó encara per a fer servir una funció.

Per tant, quan veiem que hi ha alguna tasca concreta, que nosaltres percebem que la podríem aïllar en una funció ja que és previsible que la podem fer servir vàries vegades durant la execució d'un programa, és convenient que ho fem. Això significa crear la funció i definir la capçalera (valor de retorn, nom i paràmetres que necessita per a treballar). En el cas de "esMajorQue9", per a fer la seva tasca l'únic que necessita saber és el nombre que ha de comparar.

7. Tipus de funcions/mètodes

Segons la existència de paràmetres o de retorn de dades podem tenir funcions o mètodes que rebin o no paràmetres. Anem a veure exemples dels 4 casos possibles de capçaleres:

a) Funció que no retorni res (mètode) i que rebi valors: def metode1 (valor)
b) Funció que no retorni res (mètode) i que no rebi cap valor: def metode2 ()
c) Funció que rep valors i en retorna un: def multiplicaPer2(valor)
d) Funció que no rep cap valor i en retorna un: def aleatori()

Anem a explicar-los amb detall:

a) **Un mètode que rep paràmetres**. Un exemple serien "print" i "int", que reben paràmetres (els hem de dir què volem mostrar o quina dada convertir a enter, respectivament). Un exemple senzill podria ser el d'un mètode que mostrés per pantalla un valor multiplicat per 2:

```
1. def mostraPer2(num) :
2.   print("El doble de ",num," és ", num*2)
3.
```

b) **Un mètode que no rep paràmetres**. En aquest cas posem no posem res a dins del parèntesi de la capçalera indicant que no es rebran paràmetres. Un exemple podria ser un mètode que implementa els passos per a fer una tasca concreta, que no necessita informació que se li passi i que no té necessitat de retornar res. Per exemple, l'acció de caminar, que podria ser resumida en accions concretes, on crides al mètode corresponent:

```
1. def caminar() :
2.   aixecarPeuEsquerre()
3.   inclinarEndavantElCos()
4.   baixarPeuEsquerre()
5.   aixecarPeuDret()
6.   moureEndavantPeuDret()
7.   baixarPeuDret()
8.
```

c) **Una funció que rep paràmetres**. Exemples són les funcions matemàtiques clàssiques com log, sin, cos, etc. Un exemple podria ser una funció que multiplica per 2 un nombre:

```
1. def multiplicaPer2(num) :
2.   resultat = num*2
3.   return resultat
4.
```

En aquest cas la instrucció "return" col·loca el valor desitjat en el que retorna la funció, que en aquest cas és el contingut de la variable "resultat".

d) **Una funció que no rep paràmetres**. En aquest cas no posem res a dins del parèntesi de la capçalera indicant que no es rebran paràmetres. Un exemple seria una funció que retorna un nombre aleatori:

```
1. def aleatori() :
2.     resultat = random.random()   #ja veurem més endavant com funciona random().
3.     return resultat
4.
```

Exercici proposat 10: Fer una funció que retorni 1 si un nombre és positiu i 0 en cas contrari.

Exercici proposat 11: Fer un mètode que escrigui per pantalla si un nombre és positiu , negatiu o zero.

Exercici proposat 12: Combinar els exercicis proposats 10 i 11 per tal que el mètode que mostra per pantalla si és positiu o no, ho faci cridant la funció de l'exercici 10.

Exercici proposat 13: Fer una funció que mostri per pantalla si el nombre passat per paràmetre multiplicat per 3 i dividit per 2 és més gran que 9.

8. Solucions als exercicis proposats

Solució Exercicis proposats del 1 al 4: Veiem totes les solucions al mateix codi, per a simplificar.

- Exercici proposat 1: Entrar 2 nombres enters per teclat i mostrar per pantalla si són iguals o diferents.
- Exercici proposat 2: Entrar 2 nombres enters per teclat i mostrar per pantalla si els dos són 3.
- Exercici proposat 3: Entrar 2 nombres enters per teclat i mostrar per pantalla si, al menys un dels 2 nombres és 3.
- Exercici proposat 4: Entrar 1 nombre enters per teclat i mostrar per pantalla si és o no diferent de 3.

```
1.  numero1 = int(input("Entri el 1er numero:"))
2.  numero2 = int(input("Entri el 1er numero:"))
3.
4.  #Solució exercici 1
5.  if numero1==numero2 :
6.    print("són iguals")
7.  else :
8.    print("són diferents")
9.
10. #Solució exercici 2
11. if numero1==3 or numero2==3 :
12.   print("al menys un és =3")
13. else :
14.   print("cap és =3")
15.
16. #Solució exercici 3
17. if numero1==3 and numero2==3 :
18.   print("ambós son =3")
19. else:
20.   print("els dos no són =3")
21.
22. #Solució exercici 4
23. if numero1 != 3 :
24.   print ("el numero1 es != 3")
25.
```

Exercici proposat 5: Què passaria si intercanviéssim les línies 3 i 4 del programa previ? Què sortiria per pantalla? Intenta fer-ho sense ordinador i, després, pots provar el resultat a l'ordinador.

Tenim el següent programa:

```
1. numero=1
2. while numero<=3 :
3.       numero = numero +1
4.       print(numero)
5.
```

Per pantalla obtenim: "2 3 "

Exercici proposat 6: Mostreu per pantalla els nombres de l'1 al 5, fent servir "while" .

```
1. numero=1
2. while numero<=5 :
3.   print(numero)
4.   numero = numero +1
```

Fixeu-vos que, si intercanviéssim les línies 3 i 4 no faria la tasca proposada. Proveu-ho.

Exercici proposat 7: Mostreu per pantalla els nombres de l'5 al 1, fent servir "while" .

```
1. numero=5
2. while numero>=1 :
3.   print(numero)
4.   numero = numero - 1
```

Exercici proposat 8: Entre un nombre enter per pantalla. Mostreu per pantalla els nombres des d'aquest nombre entrat fins al 5, fent servir "while" . Per exemple, si entrem el 3, hauria de mostrar els nombres 3, 4 i 5.

```
1. print("Entra un nombre:")
2. numero = int(input())        #ara hem decidir mostrar el missatge en el print anterior
3. while numero<=5 :
4.   print(numero)
5.   numero = numero +1
```

Com abans, si intercanviéssim les línies 4 i 5 no faria la tasca proposada. Proveu-ho.

Fixeu-vos, també, que si el nombre és 6 o superior no es mostra res per pantalla. Proveu-ho.

Exercici proposat 9: El mateix d'abans però decremental fins a 1.

```
1. print("Entra un nombre:")
2. numero = int(input())
3. while numero>=1 :
4.   print(numero)
5.   numero = numero - 1
```

Exercici proposat 10: Fer una funció que retorni 1 si un nombre és positiu i 0 en cas contrari.

```
def esPositiu (num) :
    if  num>0 :
        return True
    else:
        return False

```

Exercici proposat 11: Fer un mètode que escrigui per pantalla si un nombre és positiu , negatiu o zero.

```
def EsPositiuNegatiuZero(num) :
  if num>0 :
      print("El nombre és positiu.“)
  else :
      if num<0 :
          print("El nombre és negatiu.“)
      else :
          print("El nombre és zero.“)

```

Exercici proposat 12: Combinar els exercicis proposats 10 i 11 per tal que el mètode que mostra per pantalla si és positiu o no ho faci cridant la funció de l'exercici 10.

```
def esPositiu (num) :
    if  num>0 :
        return True
    else:
        return False

def EsPositiuNegatiuZero(num) :
    if esPositiu(num) :
        print("El nombre és positiu.“)
    else :
        if num<0 :
            print("El nombre és negatiu.“)
        else :
            print("El nombre és zero.“)

```

Exercici proposat 13: Fer una funció que mostri per pantalla si el nombre passat per paràmetre multiplicat per 3 i dividit per 2 és més gran que 9.

```
def esMesGranQue9(num) :
  if num*3/2>9 :
      print("És >9“)
  else :
      print("No és >9“)
```