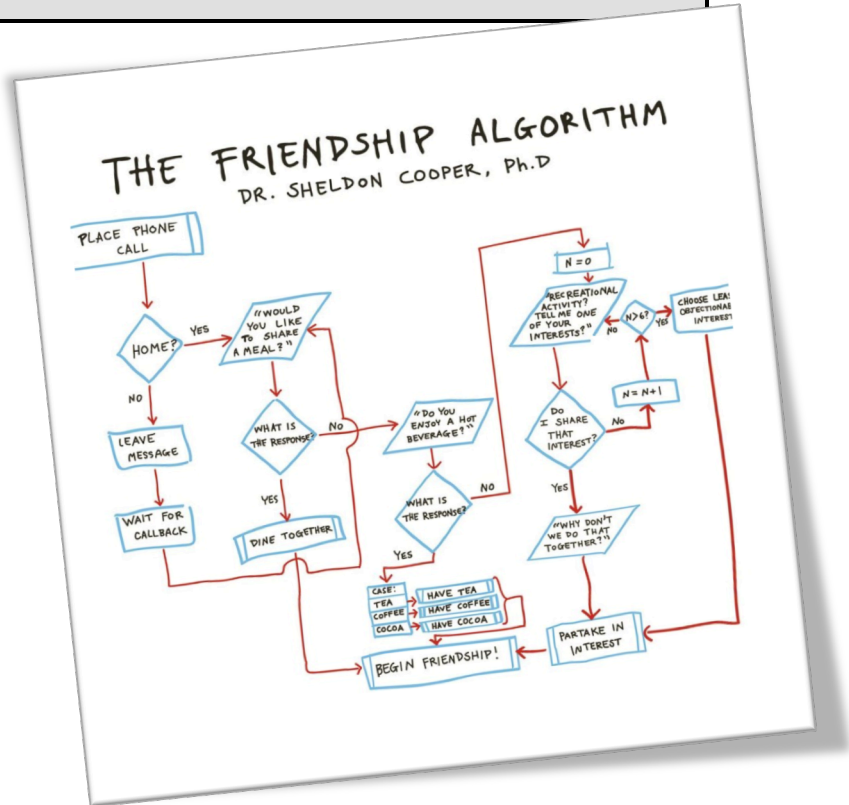


# Tema 4

## Estructures de control en Python

1. Sentències
2. Bifurcacions:
  - 2.1. Simple: *if*
  - 2.2. Doble: *if-else*
  - 2.3. Múltiple: *if-elif...-else*
  - 2.4. Altres consideracions
    - 2.4.1. *ifs* niuats
    - 2.4.2. Condicions compostes
3. Bucles
  - 3.1. Condicionals: *while*
  - 3.2. Incondicionals: *for*
  - 3.3. Alteració dels bucles: *break* i *continue*
4. La sentència *pass* de Python



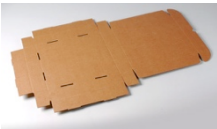
# 1. Sentències

Les sentències són cadascuna de les ordres que es donen a un programa. Hi ha de diferent tipus, independentment del llenguatge de programació:



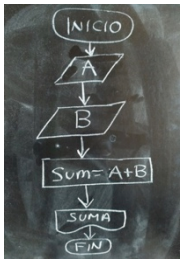
- **D'importació.** Indiquen quines llibreries del programa van a fer-se servir. Com ja hem vist, les llibreries són un conjunt de funcions que ja estan fetes. Les sentències d'importació van al principi del programa.

```
import math  
  
print("L'arrel de 25 és", math.sqrt(25))
```



- **Declaratives.** No és el cas de Python, però en C i Java vorem que per a usar una variable primer cal declarar-la o definir-la (indicar el tipus). Això es fa en sentències declaratives. Ara bé, també entren en este grup les declaracions de funcions (ja entrarem en detall).

```
def saluda(nom):  
    print("Hola, " + nom + "!")  
  
saluda("Pep")  
saluda("Maria")
```



- **Instruccions seqüencials.** Bàsicament són les instruccions d'assignació, de mostrar per pantalla i llegir de teclat. Encara que hi ha d'altres com: emetre un so, esborrar un fitxer... o invocar (cridar) a una funció.

```
edat = int(input("Edat:"))  
any_naix = 2020 - edat  
print("Nasqueres en", any_naix)
```



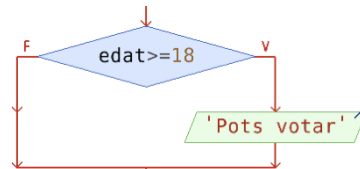
- **De control.** Són les que vorem en este tema. Permeten, depenent d'una condició, executar unes instruccions o altres (**bifurcacions**) o repetir un conjunt d'instruccions (**bucles**).

```
edat = int(input("Edat:"))  
if (edat >= 18):  
    print("Vota")  
else:  
    print("No vota")
```

## 2. Bifurcacions

Les instruccions de bifurcació (o selecció) serveixen per a quan volem executar un conjunt d'ordres només si es compleix alguna condició determinada.

La instrucció que s'usa en tots els llenguatges és *if*, però cada llenguatge té la seua sintaxi. Veiem la de Python.



### 2.1. Bifurcació simple: *if*

S'executen unes instruccions només si es compleix una condició. Si no es compleix la condició no s'executa res en particular. Usarem

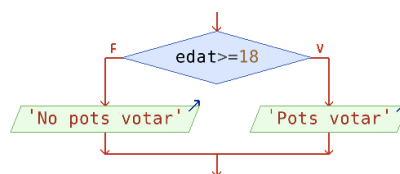
```
edat = int(input("Quants anys tens?"))

if edat > 10:
    print("Com ja eres majoret puc dir-te que...")
    print("... els reis són els pares")

print("Bon Nadal")
```

Veiem que la sintaxi és:

```
if condició :
    acció__1
    acció2
    ...
    acció_N
```



### 2.2. Bifurcació doble: *if-else*

Si es compleix una condició s'executen unes instruccions. Si no es compleix, s'executen unes altres. Usarem *if* i *else*:

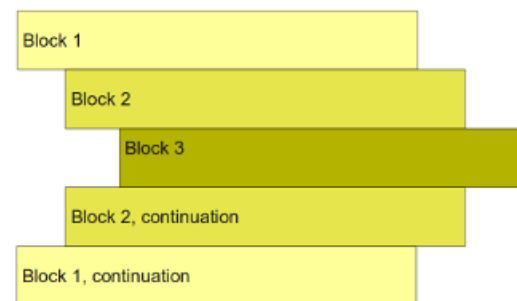
```
a = int(input("Dis-me un número:"))
b = int(input("Dis-me un altre número:"))

if a > b:
    major = a
else:
    major = b

print("El més gran és: ", major)
```

## Sagnat obligatori en Python

Les instruccions que depenen d'una condició es diu que formen un "bloc".



Python: els blocs van **sagnats obligatòriament**, respecte la condició del bloc.

```
edat = int(input("Quants anys tens?"))

if edat < 18:
    {print("No pots votar")
     {print("No deus beure alcohol")
     {print("No deus fumar")
else:
    {print("Pots fer de tot")
print("Ja ho saps")
```

Açò és un bloc.

Açò és un altre bloc.

C, Java (i majoria de llenguatges): els blocs van tancats **entre claus** { ... } i **preferiblement sagnats**.

```
...
if (condició) {
    ...
    ...
}
else {
    ...
    ...
}
```

## Exercicis sobre bifurcacions

1. Fes un programa que calcule les solucions reals de l'equació de 2n grau

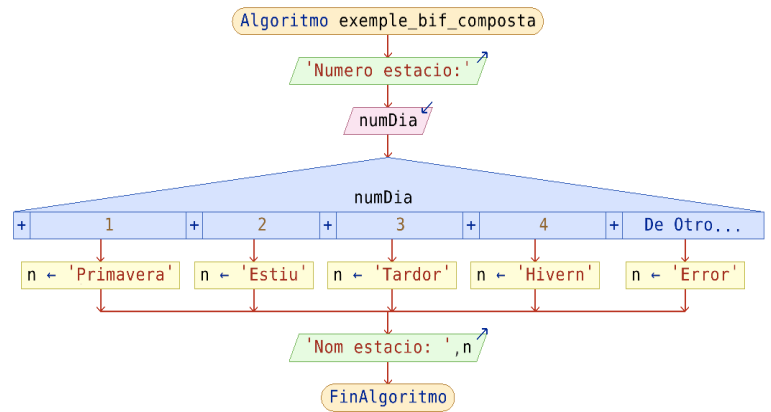
$ax^2 + bx + c = 0$  mitjançant la fórmula: 
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- S'ha de demanar per teclat quins valors tenen  $a$ ,  $b$  i  $c$ .
- Cal comprovar si té solució (allò que està dins l'arrel ha de ser positiu).
- Ha de mostrar les 2 solucions de la  $x$  (una amb el + i altra amb el -). O bé, només 1 solució si les 2 són la mateixa.
- Per a calcular l'arrel usarem la funció `math.sqrt()`, de la llibreria `math`.

### 2.3. Bifurcació múltiple: *if-elif-...-else*

Podem prendre diferents accions en base a diferents condicions.

Usarem 1 o diversos *elif*, que és una contracció d'*else if*.



```
estacio = int(input("Dis-me estació (1 a 4)"))

if estacio == 1:
    print("Primavera")
elif estacio == 2:
    print("Estiu")
elif estacio == 3:
    print("Tardor")
elif estacio == 4:
    print("Hivern")
else:
    print("Error")

print("Adéu")
```

Este és un exemple complet de la sentència *if*. Com hem vist, les parts *elif* i *else* són opcionals. Per tant, la sintaxi completa de l'*if* és:

```
if condició_1 :
    accions_1
elif condició_2 :
    accions_2
...
elif condició_N :
    accions_N
else
    accionsElse
```

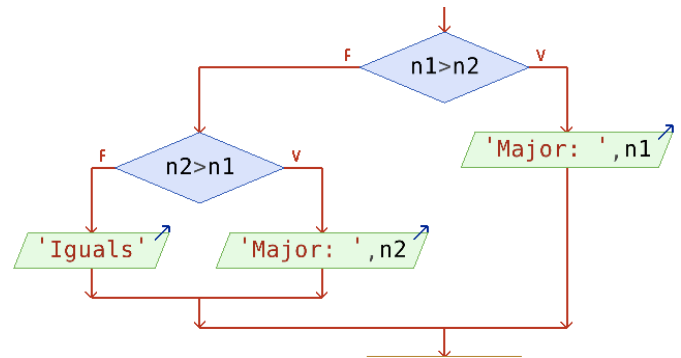
#### Exercicis sobre bifurcació múltiple

2. Demana una nota amb decimals i mostra el text corresponent: "ins", "suf", "bé", "not" o "exc". O bé "error" si la nota no està entre 0 i 10.
3. Calculadora. Fes un programa que llija de teclat 2 números i una operació aritmètica (S/R/P/D). El programa farà el càlcul i imprimirà el resultat.

## 2.4. Altres consideracions sobre les bifurcacions

### 2.4.1. Ifs niuats

Podem posar un *if* dins d'un altre, tant en la part de l'*if*, o d'algun *elif* o de l'*else*.



Vegem un exemple:

```
edat = int(input("Quants anys tens?"))
if edat < 0:
    print("Error")
else:
    print("Edat correcta")
    if edat < 12:
        print("No has fet ESO")
        if edat < 6:
            print("No has fet primària")
        else:
            print("Estàs en primària")
        print("Has de fer ESO")
    elif edat < 16:
        print("Estàs en ESO")
    elif edat < 65:
        print("Estàs en edat de treballar")
    else:
        print("Xe, jubila't!")
    print("T'he dit coses segons l'edat")
print("Adéu")
```

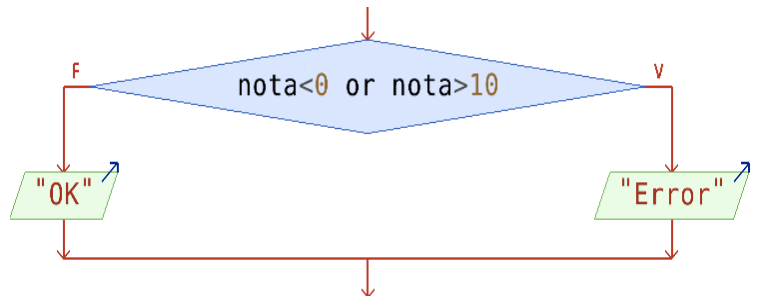
Fixa't que els **sagnats** són molt importants. Estos 3 últims *print* s'executaran en situacions diferents.

### Exercici sobre ifs niuats

4. Per a entendre el flux d'execució del programa anterior, fes l'algorisme corresponent amb un ordinograma. Millor si ho fas amb la ferramenta *Pseint*, per veure si el funcionament és l'esperat.

## 2.4.2. Condicions compostes

Cada condició que es posa en els *if* (també en els *while*, com vorem després) pot ser composta. És a dir: diverses condicions unides amb **or** i/o **and**, a més de poder tindre l'operador **not**.



```
if edat >= 14 and edat <= 30:
    print("Pots traure't en carnet jove")
else:
    print("No pots traure't el carnet jove")
```

O bé, podem invertir els missatges si posem la condició contrària:

```
if not(edat >= 14 and edat <= 30):
    print("No pots traure't en carnet jove")
else:
    print("Pots traure't el carnet jove")
```

O bé, aplicant *De Morgan* a la condició:

```
if edat < 14 or edat > 30:
    print("No pots traure't en carnet jove")
else:
    print("Pots traure't el carnet jove")
```

### Exercicis sobre bifurcacions usant condicions compostes

Fes els exercicis següents usant condicions compostes (encara que es poden fer amb condicions simples).

5. Fes un programa que mostre el màxim de 3 números introduïts per teclat.
6. Demana una nota sense decimals i mostra el text corresponent: "ins", "suf", "be", "not" o "exc". O bé "error" si la nota no està entre 0 i 10.
7. Demana per teclat el pes d'una persona, en quilos. Tant si el pes és menor de 10 kg, com si és > 200 kg, ha de mostrar error. En cas contrari mostrarà el pes en grams i també un missatge: "flac" si és menor de 50 kg, "normal" si està entre 50 i 100 kg, o "sobrepés" en altre cas.

### 3. Bucles

A voltes necessitem que un conjunt d'instruccions s'executen diverses vegades. Quantes vegades? Potser se sàpiga a priori o potser no. Per tant, tindrem 2 tipus de bucles:

- **Bucles condicionals:** es repetirà mentre es complisca una condició
- **Bucles incondicionals:** es repetirà un número determinat de vegades.



#### 3.1. Bucles condicionals: *while*

Suposem que volem mostrar l'arrel quadrada d'un número introduït per teclat:

```
import math
num = int(input("Dis-me un número:"))
print("L'arrel quadrada de %d és %f" %(num, math.sqrt(num)))
```

Funcionarà bé a no ser que introduïm un número negatiu (ja que l'arrel quadrada d'un número negatiu no té solució): *ValueError: math domain error*

Podríem solucionar-ho amb un *if*:

```
import math
num = int(input("Dis-me un número:"))

if num<0:
    print("No es pot calcular l'arrel quadrada d'un número negatiu")
    num = int(input("Torna'm a dir un número:"))

print("L'arrel quadrada de %d és %f" %(num, math.sqrt(num)))
```

Funcionarà bé si en el 2n intent donem un número positiu. Però i si tornem a donar un número negatiu en el 2n intent? Quants *ifs* hauríem de posar? No ho sabem. Per tant, voldrem repetir el fet de demanar un número MENTRE el número introduït siga incorrecte:



```
import math
num = int(input("Dis-me un número:"))

while num<0:
    print("No es pot calcular l'arrel quadrada d'un número negatiu")
    num = int(input("Torna'm a dir un número:"))

print("L'arrel quadrada de %d és %f" %(num, math.sqrt(num)))
```

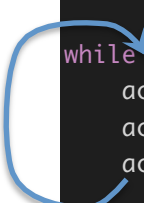


És a dir: **mentre** es complisca la **condició** de si el valor de *num* és negatiu, s'executaran les instruccions de dins del bucle (les sagnades) i es tornarà a comprovar si *num* és negatiu. Quan ja no siga negatiu, eixirà del bucle i el programa continuarà fent les instruccions de fora del bucle.

A la vista d'este exemple, veiem que:

- Usarem un bucle condicional *while* quan el programador **no sap quantes** vegades s'haurà de repetir un conjunt d'instruccions.
- La variable (o variables) de la condició hauran de tindre **un valor abans** d'entrar al bucle.
- Dins del bucle ha d'haver alguna instrucció que permeti **canviar el valor** d'eixa variable (o variables) de la condició.

Sintaxi del *while* en Python:



```
inicialització variables de la condició
while condició :
    acció_1
    acció_2
    acció_N
accions fora del bucle
```

És a dir: quan el flux del programa arriba al *while*, es comprova si la condició s'avalua a *True* o a *False*.

- Si és *True* s'executaran les accions 1 a N i el flux tornarà al *while*.
- Si és *False* el flux eixirà del bucle i s'executaran les accions de fora.

### Exercicis sobre bucles condicionals

8. Programa que, **repetidament**, mostre un menú amb 4 opcions (Demandar temperatura / Pujar 1 grau / Baixar 1 grau / Eixir), que demane per teclat una opció i l'execute. Cada vegada que s'augmente o disminuïska, també es mostrarà la nova temperatura. Després del bucle es mostrarà quantes vegades s'ha canviat la temperatura.

### 3.2. Bucles incondicionals: *for*

Este tipus de bucles es fa quan el programador ja sap **quantes vegades** s'han de repetir un conjunt d'instruccions (o bé eixa quantitat està guardada en una variable).



Exemples:

- Si volem mostrar una taula de multiplicar, sabem que s'haurà de fer un *print* **10 vegades**.
- Si es demana per teclat de quants alumnes volem demanar l'edat, si eixe valor està en la variable *q\_alumnes*, el bucle de demanar edats s'haurà de repetir ***q\_alumnes* vegades**.

En estos bucles hem d'anar comptant les iteracions fins arribar a la quantitat que volem. Per a això ens farà falta una variable, que farà de comptador o índex.

El bucle *for* en Python es bastant diferent al de C i Java.

Exemple: mostrar els números de l'1 al 9 (un en cada línia):

**Python:**

```
for i in range(1, 10):  
    print(i)
```

**C i Java:**

```
int i;  
for (i=1 ; i<10 ; i++) {  
    System.out.println( i );  
}
```

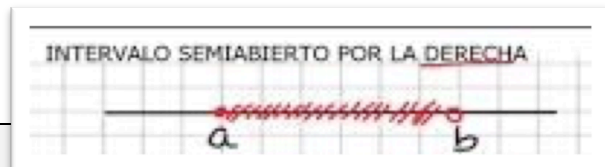
Anem a explicar detalladament el *for* de Python. Al tema següent ja explicarem el *for* de Java.

## Bucle *for* en Python

Recordem l'exemple que hem vist de *for* en Python:

```
for i in range(1, 10):  
    print(i)
```

Ací la variable *i* anirà agafant tots els valors que hi ha en la llista de valors representats en *range(1, 10)*. Per a entendre-ho, veiem com funciona el *range*.



### La funció *range* de Python

Amb el *range* obtenim un conjunt de valors anomenat *llista* (ja parlarem més endavant de les llistes).

La funció admet 1, 2 o 3 paràmetres. Veiem 3 exemples:

<code>range(10)</code>	-->	<code>[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]</code>
<code>range(4, 10)</code>	-->	<code>[4, 5, 6, 7, 8, 9]</code>
<code>range(4, 10, 2)</code>	-->	<code>[4, 6, 8]</code>
<code>range(10, 1, -1)</code>	-->	<code>[10, 9, 8, 7, 6, 5, 4, 3, 2]</code>

### Exemples de *for* de Python amb *range*:

```
for i in range(10, 31, 5):  
    print(i)
```

Això mostra els números: 10, 15, 20, 25, 30 (del 10 al 30, de 5 en 5).

```
for i in range(10, 31):  
    if i % 5 == 0:  
        print(i)
```

Este trauria el mateix resultat, però ja hem dit abans que és menys recomanable.

```
for i in range(10, 5):  
    print(i)
```

Açò no mostraria res, ja que no es pot anar del 10 al 5 incrementant 1. Caldria fer:

```
for i in range(10, 4, -1):  
    print(i)
```

Este sí que mostra els números: 10, 9, 8, 7, 6, 5 (del 10 al 5, de -1 en -1).

## Exemples de *for* de Python sense *range*:



Per a cada passada del *for* s'agafarà un element de la **llista** especificada:

```
for num in [3, 5, 7, 11, 13, 17]:  
    print(i, "és un número primer")
```

```
3 és un número primer  
5 és un número primer  
7 és un número primer  
11 és un número primer  
13 és un número primer  
17 és un número primer
```

```
alumnes = ["Pep", "Pepa", "Pepet"]  
  
for nom in alumnes:  
    print("Benvingut,", nom)
```

```
Benvingut, Pep  
Benvingut, Pepa  
Benvingut, Pepet
```

### Exercicis resolts en Python i Java

No vos preocupeu ara pels *fors* de Java.  
Ja ho vorem al tema següent.

9. Mostra els números del 0 al 10.

```
for i in range(11):  
    print(i)
```

```
for (int i=0 ; i<=10 ; i++) {  
    System.out.println(i);  
}
```

10. Mostra la taula de multiplicar del 3.

```
for i in range(1, 11):  
    print("3 x", i, "=", 3*i)
```

```
for (int i=1 ; i<=10 ; i++) {  
    System.out.println("3 x " + i + " = " + (3*i));  
}
```

11. Mostra els números parells del 40 fins al 0 (inclòs).

```
for i in range(40, -1, -2):  
    print(i)
```

```
for (int i=40 ; i>=0 ; i-=2) {  
    System.out.println(i);  
}
```

El rang de Python mai arriba al valor final. Per això no hem posat (40, 0, -2), ja que així no haguera mostrat el 0.

### Exercicis sobre bucles amb *for*

12. Programa que demane una taula de multiplicar i la mostre.

13. Programa que calcule el màxim de 10 números introduïts per teclat.

14. Programa que calcule el màxim, mínim i mitjana de 10 números entrats per teclat.

15. Programa que mostre les taules de multiplicar del 2 al 9.

16. Programa que calcule el factorial d'un número introduït per teclat (  $n!$  ) tenint en compte que:

$$0! = 1$$

$$n! = n * (n-1) * (n-2) * \dots * 2 * 1 \quad (\text{sent } n > 1)$$

Feu-ho amb diferents solucions:

- a) Amb un *for* recorrent els números des de l'1 fins n
- b) Amb un *for* recorrent els números des d'n fins a 1
- c) Amb un *while* recorrent els números des de l'1 fins n
- d) Amb un *while* recorrent els números des d'n fins a 1

### 3.3. Alteració de bucles: *break* i *continue*

Dins dels bucles (tant *while* com *for*), podem fer que, si ocorre alguna determinada condició, alterar el funcionament normal del bucle. Això es pot fer amb les sentències *break* i *continue*:



```
...
# Inici del bucle -----
for i in range(1,10):
    ...
    if ... :
        continue
    ...
    if ... :
        break
    ...
# fi del bucle -----
...
```

Si passa pel *continue*, el flux de control seguirà a l'inici del bucle. Si la *i* valia 7, ara valdrà 8.

Si passa pel *break*, el flux de control seguirà a la següent instrucció de fora del bucle.

És a dir:

- ***break*** interromp l'execució del bucle i seguim per la instrucció següent de fora del bucle.
- ***continue*** fa que el programa comence altra iteració, encara que no s'haja acabat l'actual. Per tant, les línies que hi ha dins d'un bucle a continuació del *continue* no s'executen en eixa iteració. Si el bucle és un *for*, anirem a l'increment de la variable comptadora o al següent element de la llista. Si és un *while*, es torna a comprovar la condició per a entrar altra vegada al bucle.

Nota: en cas de tindre estes instruccions dins de bucles niuats, només afecten al bucle on estan directament. Per exemple, si tenim un *break* en un bucle (pare) i este està dins d'un altre bucle (iaio), eixiríem del bucle pare però no del iaio.

### Exercici resolt de bucles amb *break* i *continue*:

17. Programa en Python que demane les notes dels alumnes (números enters i positius entre 0 i 10) i que després mostre quantes notes s'han introduït i la nota mitjana. Per a parar d'introduir notes caldrà posar la nota 11, que no es tindrà en compte per als càlculs.

Nota: El programa es podria fer sense *break* ni *continue*, però ho farem així per a veure el seu funcionament.

```
# Inicialització de comptador i acumulador
```

```
notes = 0
```

```
suma = 0
```

```
# Demanar dades i fer càlculs
```

```
while True:
```

```
    # Introducció de la nota per teclat
```

```
    nota = input("Dis-me una nota (11 per a eixir):")
```

```
    # Comprovacions de nota OK
```

```
    if not nota.isnumeric():
```

```
        print("Han de ser números enters positius. Torna a provar.")
```

```
        continue
```

```
    nota = int(nota)
```

```
    if nota > 11:
```

```
        print("La nota màxima és 10. Torna a provar.")
```

```
        continue
```

```
    # Si volem acabar
```

```
    if nota == 11:
```

```
        print("Ja no et demane més notes. Vaig a eixir del bucle.")
```

```
        break
```

```
    # Si tot ha anat bé, farem els càlculs
```

```
    print("Nota introduïda ok:", nota)
```

```
    notes += 1
```

```
    suma += nota
```

```
## Fi del bucle
```

```
# Mostrem els resultats
```

```
print(notes, "notes introduïdes.")
```

```
print("Nota mitja:", suma/notes)
```

Amb *while True* aconseguim fer un bucle "infinit". Només podrem eixir si fem un *break* dins.

*isnumeric* és *True* si tots els caràcters són dígit.

Si passa per aquí, torna altra vegada a l'inici del bucle.

Si passa per aquí, torna altra vegada a l'inici del bucle.

Si passa per aquí, eixim ja fora del bucle.

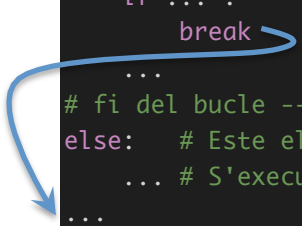
### Exercici de bucles amb *break* i *continue*:

18. Programa que demane 10 números (positius i/o negatius) i mostre la mitjana només dels positius. Fes-ho a l'estil de l'exercici resolt anterior, per a fer ús de *break* i *continue*.

## Ús de l'else de bucles amb *break*

Ja hem vist que l'else s'utilitza en sentències *if*. Però Python (no C ni Java) permeten posar-lo després d'un bucle en el qual s'haja fet algun *break*:

```
...
# Inici del bucle -----
while ... :
    ...
    if ... :
        break
    ...
# fi del bucle -----
else: # Este else és del bucle, no de l'if.
    ... # S'executaran estes accions si NO hem passat pel break.
...
```



La part de l'else s'executarà només quan el bucle ha acabat de manera "normal":

- En un while quan ha acabat el bucle perquè no es complia la condició.
- En un for quan ha acabat el bucle perquè ja s'han recorregut els elements de la llista.

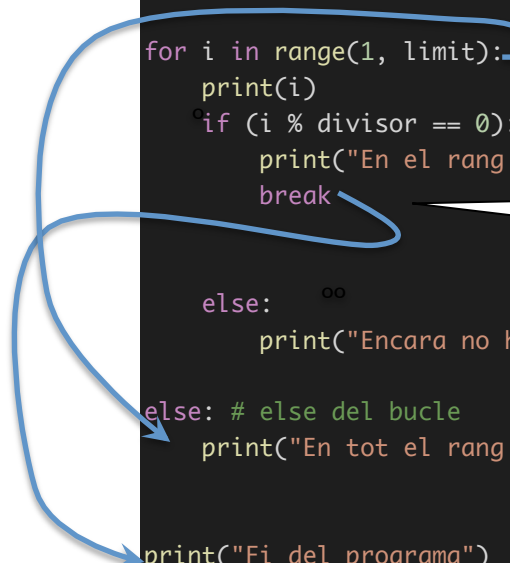
### Exercici resolt de bucles amb *break* i *continue*:

19. Programa en Python que ens diga si ha trobat o no algun múltiple d'un número determinat en un cert rang de números.

```
limit = int(input("Fins quin número vols buscar múltiples?"))
divisor = int(input("De quin número vols buscar múltiples?"))

for i in range(1, limit):
    print(i)
    if (i % divisor == 0):
        print("En el rang sí que hi ha almenys 1 múltiple de", divisor)
        break
    else:
        print("Encara no he trobat múltiple de", divisor)
else: # else del bucle
    print("En tot el rang no hi ha cap múltiple de", divisor)

print("Fi del programa")
```



Si el bucle acaba de forma "normal", sí que s'executarà l'else de fora del bucle.

Si el bucle acaba per un *break*, no s'executarà l'else de fora del bucle.



## Exercici de bucles amb *break* i *else*

20. Fes un programa en Python que simule un caixer automàtic. Es demana per teclat la clau contínuament mentre no siga correcta (1234). Però com a molt són 5 intents. Després del bucle caldrà indicar si s'han superat els intents o si s'ha encertat la clau.

- a) Fes el programa usant un *while* amb *break* i l'*else* del *while*.
- b) Fes el programa usant un *for*
- c) Fes el programa usant un *while* sense *break* (ni l'*else* del *while*, clar).

## 4. La sentència *pass* de Python

La sentència *pass* no fa **res**.

Simplement està per a quan el programador vol deixar alguna part del programa temporalment sense indicar el codi, per alguna raó:



- a) Perquè està en les fases inicials del programa i ho deixa per a després:

```
if nota >= 0 and nota <= 10:
    pass # He d'acabar el programa amb càlculs amb la nota.
else:
    print("Error en nota")
```

Si no es posa el *pass*, donaria error sintàctic.

```
def mostrarResultats(a, b, c):
    pass # He d'implementar esta funció encara
```

No et preocupes ara per les funcions. Ja les vorem.

- b) Per a millorar la llegibilitat, etc.:

```
if edat >= 18:
    pass # No fem res especial si té 18 anys o més
else:
    print("No pots votar")
```