Authors Sonia Estradé
José M Gómez
Ricardo Graciani
Frank Güell
Manuel López
Xavier Luri
Josep Sabater

Control de Flux (I). Les sentències `if` i `while`

Control de flux: conceptes

Els programes que hem fet fins a ara s'executaven en una **seqüència lineal**, seguint les sentències en ordre de la primera a l'última, però en programes reals sol ser necessari **modificar el flux d'execució sobre la base de sentències condicionals basades en l'estat i valors de les dades de programa**. Bàsicament hi han dos tipus de sentències que ens permeten modificar el flux d'execució:

- La sentència condicional. Permet executar les instruccions que hi ha sota amb identació si la operació booleana és certa (`True`)
- Els bucles. Permeten executar unes determinades línies de codi, identades sota la sentència de bucle, de forma repetitiva metre la operació booleana que hi ha al bucle sigui certa.

Per a aquesta fi Python proporciona sentencies para realitzar bifurcacions i bucles que discutirem en aquesta secció.

Exemple de bifurcació:

Llença una moneda a l'aire. Si surt cara guanyes, si surt creu perds. La implementació d'aquest codi requereix de bifurcacions.

Exemple de bucle:

La realització del següent sumatori

$$\sum_{i=1}^{N} i$$

Sense sentències de control de flux no té massa sentit implementar un sumatori; el codi no és flexible, no té massa utilitat.

Bifurcacions

Anem a intentar implementar el codi que simula el llençament de la moneda.

Podem fer servir la llibreria **`random`** per tal de generar un valor aleatori. La funció **`randint(inici, final)`** genera un enter aleatori

$$inici <= EnterGenerat <= final$$

In [3]:
```
import random as rnd
n = rnd.randint(0,1)

# Fem l'associació
#    CARA = 0
#    CREU = 1

# per tant:

#    SI N ÉS IGUAL A CARA, GUANYO
#    EN CAS CONTRARI N ÉS IGUAL A CREU, PERDO
print(n)
```

```
0
```

Un altre mètode de la llibreria **`random`** és **`choice(<sequence>)`**. Ens retorna un element aleatori de una seqüencia no buida:

In [11]:
```
import random

# Seleccio aleatoria d'un element de la sequencia
sequence = 'Ab5,.9K-_m'
ch = random.choice(sequence)
print( "Random choice from '{}' is '{}'".format(sequence, ch) )
```

```
Random choice from 'Ab5,.9K-_m' is 'K'
```

Podem generar un caràcter aleatori amb **`choice(<sequence>)`** fent servir la cadena **`string.ascii_letters`** com argument:

In [5]:
```
import string
import random

print("Characters contained in 'string.ascii_letters':\n\t{}\n".format(string

# Draw a letter in string.ascii_letters
rand_letter = random.choice(string.ascii_letters)
print("Random letter generated: ", rand_letter)
```

```
Characters contained in 'string.ascii_letters':
        abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

Random letter generated:  A
```

També podem seleccionar un element aleatori d'un conjunt:

In [6]:
```
import random

elements = ('cat', 'dog', 'turtle', 'iguana', 1, 5, 8, 'pig')

# Draw an element
rand_elem = random.choice(elements)
print("Random choice: ", rand_elem)
```

```
Random choice:  1
```

La llibreria **`random`** conté altres mètodes. Per més detalls podeu consultar la seva documentació (https://docs.python.org/3.4/library/random.html).

Sentència if

Les sentències **`if`** permeten l'execució condicional d'un bloc de codi basant-se en expressions booleanes. Si l'expressió booleana dóna com a resultat **`True`** el bloc de codi s'executa, si dóna com a resultat **`False`** no s'executa.

```
if (condició):
    # bloc de codi que s'executa si la expressió és True

# El flux del codi continua
```

El bloc de codi al que s'aplica la condició es marca mitjançant la indentació.

In [7]:
```
# PRIMERA APROXIMACIÓ A LA SOL·LUCIÓ DEL PROBLEMA
CARA = 0
CREU = 1
if n == CARA:
    print("Guanyo!")
if n == CREU:
    print ("Ooooh, he Perdut")
```

```
Guanyo!
```

En aquest cas anterior, el codi executa les dues sentències **`if`**. Comprova primer si n és igual a CARA. Si aquesta condició és certa imprimeix "Guanyo!"

Tot seguit executa la següent instrucció **`if`** i comprova si n és igual a CREU. Si aquesta condició és falsa, no imprimeix "Ooooh, he Perdut".

Existeix l'opció de que un cop s'ha detectat que una condició és certa, no verifiqui la resta de condicions.

```
if (condició1):
    # Bloc de codi 1

elif (condició2):
    # Bloc de codi 2

else:
    # Bloc de codi 3

# Altre codi
```

En aquest cas el flux es el següent:

- si **`condició1`** es **`True`**: s'executa el **`bloc de codi 1`** i després continua amb **`Altre codi`**.
- si **`condició1`** es **`False`**: s'avalua **`condició2`**.
  - si **`condició2`** es **`True`**: s'executa el **`bloc de codi 2`** i després continua amb **`Altre codi`**.
  - si **`condició2`** es **`False`**: s'executa el **`bloc de codi 3`** i després continua amb **`Altre codi`**.

veiem un exemple...

In [8]:
```
n = rnd.randint(0,2)
CARA = 0
CREU = 1
if n == CARA:
    print ("surt cara")
    n = CREU
elif n == CREU:
    print ("surt creu")
    n = 2
else:
    print("cau de cant")
```

```
cau de cant
```

Noteu que independentment del valor de y, només s'executa una bifurcació, tot i que explícitament fem canviar el valor de y per donar-li altres valors. Veiem que passa si posem aquesta seqüència només amb **if**'s

In [9]:
```
n = rnd.randint(0,2)
CARA = 0
CREU = 1
if n == CARA:
    print ("surt cara")
    n = CREU
if n == CREU:
    print ("surt creu")
    n = 2
if n == 2:
    print("cau de cant")
```

```
cau de cant
```

Tenim doncs aquestes dues opcions. En certs casos necessitarem que un cop el programa entra en una de les bifurcacions, no executi la resta d'opcions. En altres casos necessitarem que es comprovin totes les opcions. Dependrà del tipus de problema amb el que ens enfrontem.

Les comprovacions poden ser tant complicades com sigui necessari. Es poden fer servir operadors booleans com **and** i **or**, i comparadors com:

- A == B A és igual que B?
- A < B A és menor que B?
- A > B A és major que B?
- A <= B A és menor o igual que B?
- A >= B A és major o igual que B?
- A != B A és diferent de B?

veiem un senzill exemple:

In [11]:
```
a = 1
b = 2
c = 3
d = 4

if a==1 and b==2 and ( c!=5 or d>4 ) :
    print ("Condició certa")
else:
    print ("Condició falsa")
```

```
Condició certa
```

Exercici:

Feu un programa que generi un número aleatori entre 1 i 10 i demani el nom del jugador. El programa preguntarà al jugador un número entre 1 i 10. Es guanya si s'encerta el valor. Es poden introduir tres valors com a màxim.

In [20]:
```
#El codi aqui
import random as r
import sys

# Ordena el codi correctament...

y = int(input("introdueix un numero entre 1 i 10 "))
if y == numeroMagic:
    print("Encert a la segona!! ")
    sys.exit(0)
x = int(input("introdueix un numero entre 1 i 10 "))
if x == numeroMagic:
    print("Encert a la primera!! ")
    sys.exit(0)
print ("no has encertat!! Torna-ho a provar ")
print ("no has encertat!! Torna-ho a provar ")
z = int(input("introdueix un numero entre 1 i 10 "))
if z == numeroMagic:
    print("Encert a la tercera!! ")
    sys.exit(0)
jugador1 = input("nom del jugador ")
numeroMagic = r.randint(1,10)
print("Ooooh! {} no has encertat el número. Era {}. Els teus valors eren {},
```

```
nom del jugador Manel
introdueix un numero entre 1 i 10 4
no has encertat!! Torna-ho a provar
introdueix un numero entre 1 i 10 3
no has encertat!! Torna-ho a provar
introdueix un numero entre 1 i 10 6
Ooooh! no has encertat el número. Era 10. Els teus valors eren 4, 3 i 6
```

Es podria millorar aquest codi fent servir `if - elif - else` ? Feu l'exercici implementant ara `if` i `else`

Bucles

Els bucles ens permeten fer operacions iteratives sempre i quan una determinada condició sigui certa. En el cas del sumatori,

$$\sum\limits_{i=1}^N i$$

la condició de iterativitat seria que **`i`** ha de ser menor o igual que **`N`**.

Sentència `while`

Repeteix un bloc de sentències mentre la condició de control associada sigui certa. La condició de control és avaluada cada vegada, abans de executar el bloc de codi. Si la condició és falsa d'entrada no s'executa mai el bloc.

Nota: compte amb els bucles infinits!

La sintaxi bàsica és:

```
while condició :
    # executa si condició és certa
    sentència 1
    ...

else:
    # executa quan condició és falsa
    sentència 2
```

La sentència **`else`** és pròpia de Python (existeix en pocs llenguatges de programació i permet executar codi quan sortim del bucle amb la condició falsa.

Exemple:

Sol·lucionem el problema del sumatori

In [10]:
```
# Assignem a i el valor incial 1 i demanem el valor de N

i = 1
N = int(input("introduir el valor final del sumatori "))
suma = 0
while i <= N:
    suma = i + suma
    i = i + 1
print("el valor del sumatori és" suma)
```

```
introduir el valor final del sumatori 45
el valor del sumatori és 1035
```

break

Dins els bucles **`while`** també es pot usar la sentència **`break`**. Aquesta sentència permet interrompre l'execució d'un bloc **`while`**. Quan s'executa aquesta comanda l'execució salta el què quedi del bloc **`while`** i segueix amb el programa. Tot i que pot anar tota sola, normalment, aquesta sentència va lligada amb la instrucció **`if`**. En aquest cas el codi després de la sentència **`else`** no se executa i la condició de control no es torna a avaluar.

La sintaxi bàsica és:

```
while (condició control):
    # bloc de codi

    if (condició de sortida):
        break

    # mes codi

else:
    # Si la condició de control s'ha avaluat False

# I mes codi
```

Exemple:

In [15]:
```
# Seguint amb l'exemple anterior, fem un sumatori però en cas que la
# suma sigui superior a un determinat valor, voldrem que el programa
# deixi de realitzar les operacions.

i = 1
N = int(input("introduir el valor final del sumatori "))
M = int(input("introduir el valor màxim que pot assolir la suma: "))
suma = 0

while i <= N:
    suma = i + suma
    i = i + 1
    if suma > M:
        print("limit assolit")
        break
if suma < M:
    print("el valor del sumatori és", suma)
else:
    print ("per al terme {}-essim hem superal el màxim {}".format(i,M))
```

```
introduir el valor final del sumatori 67
introduir el valor màxim que pot assolir la suma: 1356
limit assolit
per al terme 53-essim hem superal el màxim 1356
```

`continue`

Una altra sentència que es pot usar dins els bucles **`while`** és **`continue`**. La sentència **`continue`** permet interrompre la execució del bloc *while* en curs i saltar directament a la iteració següent. Al igual que en el cas anterior, la sentència **`continue`** pot executar-se directament, però normalment va de lligada a una instrucció **`if`**

Exemple:

Calculem els valors de y per la funció:

$$y = \frac{1}{x-a}$$

Primer pas: introduïu per consola un valor enter per a Segon pas: demaneu els valors enters de x per als quals voleu saber el valor de y

Repetiu aquest procés 10 vegades

In [ ]:
```
a = int(input("introduïu el paràmetre a: "))
MAXIM = 10
contador = 0

while contador < MAXIM:
    x = int(input("introduïu el valor de x: "))
    if x == a:
        print ("per aquest valor tenim un infinit")
        continue
    else:
        y = 1/(x-a)
        print (y)
        contador = contador + 1
```

Exercici Sentència IF + WHILE

Volem provar la nostra punteria llençant una pedra a una diana de 1 metre de diàmetre i que es troba a una alçada de 1 metre des de el centre fins al terra. Ens trobem a una distància de 10 metres. Considereu només el cas de dos dimensions i trobeu la velocitat inicial i l'angle necessari per donar-li a la diana, tenint en compte que l'alçada inicial del llençament és igual a 1 metre.

In [19]:
```
# La resposta aqui...

from math import sin, cos
PI = 3.1416
G = 9.8
dianaEixX = 10
dianaSuperiorEixY = 1.5
dianaInferiorEixY = 0.5
y0 = 1
encert = False
contador = 0

'''
    POSEU LES INSTRUCCIONS SEGUENTS EN L'ORDRE CORRECTE

y = y0+Vy0*tDiana-0.5*G*tDiana**2
tDiana = dianaEixX/Vx
if y > dianaInferiorEixY and y < dianaSuperiorEixY:
Vx = velInicial*cos(angleRadians)
Vy0 = velInicial*sin(angleRadians)
print("Objectiu assolit")
angle = float(input("introduir l'angle en graus: "))
angleRadians = angle * PI/180
encert = True
else:
print("torna-ho a provar!!")
contador = contador + 1
velInicial = float(input("Inrtroduir la velocitat m/s: "))
while encert == False:
'''

print (x,y)
print ("has encertat en {} llençaments".format(contador))

# MILLOREU EL CODI PER DIR A QUINA ALCADA DE LA DIANA HEM DONAT
```

```
Inrtroduir la velocitat m/s: 10
introduir l'angle en graus: 0
torna-ho a provar!!
Inrtroduir la velocitat m/s: 10
introduir l'angle en graus: 12
torna-ho a provar!!
Inrtroduir la velocitat m/s: 10
introduir l'angle en graus: 45
Objectiu assolit
6 1.2000007345762569
has encertat en 3 llençaments
```