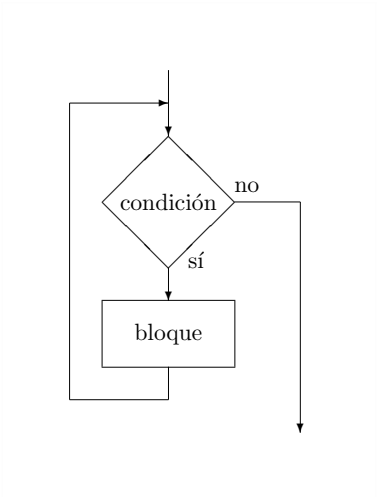
FUNDAMENTOS DE COMPUTACIÓN
Grado en ingeniería en tecnologías Industriales
Universidad de Cantabria

Práctica 7 (laboratorio)
5 de noviembre del 2019

Bucles e Iterables




```
while (condición):
    (bloque)
```

- (condición) evalúa a un booleano
- Si (condición) es Cierta, se ejecutan todas las etapas del bloque
- Comprueba (condición) de nuevo.
- Repite hasta que (condición) es Falsa.

```
i = 1
while i <= 50:
  print(i)
  i = 3*i + 1
print("Final")
```

```
i = 1
while i <= 10:
  print(i, "", end="")
```

```
a, b = 0, 1
while b < 1000:
    print(b, end = ',')
    a, b = b, a + b
```

El segundo programa es un **bucle infinito**, para interrumpirlo hay que pulsar la combinación de teclas Crtl+C.

ITERABLES (listas, tuplas, cadenas, range)

```
for (variable) in (iterable):
    (bloque)
```

- cada vez que entra en el bucle la (variable) toma un valor
- la primera vez, (variable) comienza con el primer elemento(valor) del iterable
- la siguiente, (variable) toma el siguiente elemento(valor) del iterable
- Repite hasta que el iterable no tenga más elementos(valores)

```
lista=[1, 2, 'a', True]
for x in lista:
    print(x)
print('------')
for i in range(4):
    print(lista[i])
```

```
tupla=(1, 2, 'a', True)
for x in tupla:
    print(x)
print('------')
for i in range(4):
    print(tupla[i])
```

```
cadena='12aT'
for x in cadena:
    print(x)
print('------')
for i in range(4):
    print(cadena[i])
```

La función `range` admite uno, dos o tres argumentos números enteros `range(stop)` `range(start stop (step) )`.

```
>>> range(5)                            >>> list(range(3))
range(0, 5)                            [0, 1, 2]
>>> type(range(3, 6))                    >>> list(range(1, 5))
<class 'range'>                        [1, 2, 3, 4]
>>> list(range(0, 9, 2))
[0, 2, 4, 6, 8]
>>> for i in range(2, 15, 3): print(i**2)
...
4
25
64
121
196
>>>
```

El siguiente programa la variable `misuma` almacena el entero 3, sumando 0+1+2 :

```
misuma = 0
for x in range(7,10):
    misuma += 1
print(misuma)
```

Y el siguiente programa la variable `misuma` almacena el entero 21, sumando 5+7+9 :

```
misuma = 0
for x in range(5,11,2):
    misuma += x
print(misuma)
```

EJERCICIOS

1. Escribe un programa para calcular el factorial de un número, sin utilizar la función `factorial` del módulo `math`, que se comporte como el ejemplo de la izquierda. Modifícalo después para que, como en el ejemplo de la derecha, no termine hasta que el valor que escriba el usuario sea `0`.

```
$ python factorial.py
n= 3
n!=6
$ python factorial.py
n= 6
n!=720
```

```
$ python factorial_2.py
n= 1
1!=1
n= 5
5!=120
n= 0
Adiós
```

   Asegúrate de que, después de que el usuario escriba `0`, no se muestre en la pantalla ningún mensaje como `0!=1`, sino solamente una despedida.

2. Haciendo caja:

```
$ python caja.py
Importe (0 para terminar): 10
Importe (0 para terminar): 3.25
Importe (0 para terminar): 12.14
Importe (0 para terminar): 7.9
Importe (0 para terminar): 0
TOTAL:                     33.29
```

3. Es sabido que

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \sum_{n=0}^{\infty} \frac{1}{\prod_{j=1}^{n} j}.$$

Escribe un programa para aproximar esta serie hasta un término dado, como en el ejemplo:

```
$ python exp.py
Tope: 0
1.0
$ python exp.py
Tope: 1
2.0
$ python exp.py
Tope: 2
2.5
$ python exp.py
Tope: 14
2.71828182846
```

4. Realiza un programa que calcule el desglose del menor número de billetes y monedas de una cantidad entera de euros. Hay billetes de 500, 200,100,50,20,10 y 5 euros y, monedas de 2 y 1 euro. Por ejemplo:

```
Dame una cantidad entera de euros: 903
El desglose es:
1 billete(s)/moneda(s)  de 500 euros
2 billete(s)/moneda(s)  de 200 euros
0 billete(s)/moneda(s)  de 100 euros
0 billete(s)/moneda(s)  de 50 euros
0 billete(s)/moneda(s)  de 20 euros
0 billete(s)/moneda(s)  de 10 euros
0 billete(s)/moneda(s)  de 5 euros
1 billete(s)/moneda(s)  de 2 euros
1 billete(s)/moneda(s)  de 1 euros
```

5. Diseña un programa que calcule la suma de las cifras de un número natural dado. Por ejemplo:

```
>>> numero = 1021
>>> suma_cifras
4
```

6. Ya hemos comentado que el NIF es un código corrector de errores. Se toma el resto r de la división Euclídea del DNI entre 23, y dependiendo del resto $0 \leq r < 23$, la letra que corresponde del DNI viene dada en la siguiente tabla:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
T	R	W	A	G	M	Y	F	P	D	X	B	N	J	Z	S	Q	V	H	L	C	K	E

Diseña un programa que dado el DNI, devuelva el NIF.

7. `continue`.— En un juego de dardos, se acumulan las puntuaciones obtenidas en diez tiradas. El programa del fichero `dardos_2.py` no tiene en cuenta las tiradas en las que se logran menos de 50 puntos. Esto mismo puede conseguirse (`dardos_3.py`) con la sentencia `continue`, que interrumpe la iteración en curso de un bucle (el resto del bloque deja de ejecutarse) para pasar a la siguiente iteración.

dardos_1.py

```
total = 0
for i in range(10):
    p = int(input('Puntos: '))
    total += p
print ('TOTAL:',total)
```

dardos_2.py

```
total = 0
for i in range(10):
    p = int(input('Puntos: '))
    if p >= 50:
        total += p
print ('TOTAL:',total)
```

dardos_3.py

```
total = 0
for i in range(10):
    p = int(input('Puntos: '))
    if p < 50:
        continue
    total += p
print('TOTAL:',total)
```

La sentencia **continue** interrumpe la iteración en curso de un bucle (el resto del bloque deja de ejecutarse) para pasar a la siguiente iteración.

Los siguientes dos programas descalifican aquellos jugadores que obtienen en alguna tirada menos de 1 punto, otorgándoles una puntuación total de -1 puntos.

dardos_4.py

```
total = 0; total1 = 0
for i in range(10):
   p = int(input('Puntos: '))
   if p < 1:
      total1 = -1
   total += p
if total1 < 0:
   print('TOTAL:', total1)
else:
   print('TOTAL:', total)
```

dardos_5.py

```
total = 0; i = 0
while i < 10:
   p = int(input('Puntos: '))
   if p < 1:
      total = -1
      i = 11
   total += p
   i += 1
print('TOTAL:', total)
```

Esto mismo puede conseguirse con la sentencia **break**:

dardos_6.py

```
total = 0
for i in range(10):
    p = int(input('Puntos: '))
    if p < 1:
        total = -1
        break
    total += p
print('TOTAL:', total)
```

dardos_7.py

```
total = 0; i = 0
while i < 10:
    p = int(input('Puntos: '))
    if p < 1:
        total = -1
        break
    total += p
    i += 1
print('TOTAL:', total)
```

La sentencia **break** interrumpe el bucle (y no solo la iteración en curso como **continue**).

8. Utiliza la sentencia `continue` para enumerar los cinco primeros años bisiestos que siguen a una fecha dada por el usuario.

```
$ python bisiestos.py
Año: 2014
 1: 2016
 2: 2020
 3: 2024
 4: 2028
 5: 2032
```

9. Utiliza la sentencia `break` para escribir un programa que realice la tarea siguiente: en primer lugar, preguntará al usuario en qué año nació; después, le irá preguntando años, mostrando en la pantalla, tras cada una de ellos, cuántos años cumplirá (o cumplió) entonces. Cuando el usuario escriba una fecha anterior a su nacimiento, el programa debe detenerse.

```
$ python cumple.py
¿Cuándo nació? 1990
Año: 2000
Cumple Ud. 10 años.
Año: 2050
Cumple Ud. 60 años.
Año: 1992
Cumple Ud. 2 años.
Año: 1900
```

10. El siguiente programa dibuja el calendario de un mes, tomando como dato de entrada el día de la semana en que cae el primer día del mes.

```
print('¿Cuántos días tiene el mes?', end=' ')
n = int(input())
print('El día 1.º del mes es:')
print('(1: lunes, 2: martes, etc.)', end = ' ')
s = int(input())

# Escribimos el encabezado del calendario.
print('  L   M   X   J   V   S   D ')

# Dejamos s-1 lugares vacíos. Cada uno ocupa  cuatro caracteres.
for hueco in range(s-1):
    print(3*' ', end = ' ')  # 3 espacios, y el que «deja» la coma

# La variable «dia» recorre todos los días del mes:  desde 1 hasta n. Al comenzar cada iteración,
# la variable «s» indica qué día de la semana es.
for dia in range(1,n+1):
    print(' {:2d}'.format(dia), end=' ')
    s += 1
    if s == 8:
        # Cambiamos de línea y de semana.
        print()
        s = 1
print()
```

```
$ python mes.py
¿Cuántos días tiene el mes? 31
El día 1.º del mes es:
(1: lunes, 2: martes, etc.) 6
  L   M   X   J   V   S   D
                      1   2
  3   4   5   6   7   8   9
 10  11  12  13  14  15  16
 17  18  19  20  21  22  23
 24  25  26  27  28  29  30
 31
```

```
$ python mes.py
¿Cuántos días tiene el mes? 30
El día 1.º del mes es:
(1: lunes, 2: martes, etc.) 2
  L   M   X   J   V   S   D
      1   2   3   4   5   6
  7   8   9  10  11  12  13
 14  15  16  17  18  19  20
 21  22  23  24  25  26  27
 28  29  30
```

11. Modifica el programa anterior para que tome como entrada no el número de días del mes, sino su número de orden (1: enero, 2: febrero, etc.). Además, este programa debe mostrar también el mes siguiente.

```
$ python mes_2.py
Mes (1: enero, 2: febero, etc.): 3
El día 1.º del mes es:
(1: lunes, 2: martes, etc.) 6
           MARZO
  L   M   X   J   V   S   D
                      1   2
  3   4   5   6   7   8   9
 10  11  12  13  14  15  16
 17  18  19  20  21  22  23
 24  25  26  27  28  29  30
 31

           ABRIL
  L   M   X   J   V   S   D
      1   2   3   4   5   6
  7   8   9  10  11  12  13
 14  15  16  17  18  19  20
 21  22  23  24  25  26  27
 28  29  30
```

```
$ python mes_2.py
Mes (1: enero, 2: febero, etc.): 12
El día 1.º del mes es:
(1: lunes, 2: martes, etc.) 1
         DICIEMBRE
  L   M   X   J   V   S   D
  1   2   3   4   5   6   7
  8   9  10  11  12  13  14
 15  16  17  18  19  20  21
 22  23  24  25  26  27  28
 29  30  31

           ENERO
  L   M   X   J   V   S   D
              1   2   3   4
  5   6   7   8   9  10  11
 12  13  14  15  16  17  18
 19  20  21  22  23  24  25
 26  27  28  29  30  31
```

12. **Célebre anécdota escolar de Gauß.**— Queremos calcular la suma $\sum_{i=1}^{n} i$, para lo que podemos hacer un programa acumulando la suma de los elementos función `range`. Sin embargo, analizando algo más la situación, podemos conseguir un algoritmo mucho más rápido.

    Escribe un programa que represente y por duplicado la suma $\sum_{i=1}^{n} i$, dando un resultado similar al que sigue:

```
$ python suma2_1.py
Tope: 5
 1 +  5 = @ # # # # #
 2 +  4 = @ @ # # # #
 3 +  3 = @ @ @ # # #
 4 +  2 = @ @ @ @ # #
 5 +  1 = @ @ @ @ @ #
```

    Esta suma duplicada toma la forma de un rectángulo de unidades (en el ejemplo, de dimensiones 6×5). Expresa su altura y su anchura en función de n y deduce una fórmula para $2\sum_{i=1}^{n} i$.

    Aprovecha la fórmula para escribir una versión ágil del programa que suma los primeros n números.

13. Deseamos enviar una carta tipo a varios estudiantes, pero adaptando algunos datos de la misma a los propios de cada estudiante: La carta es:

```
Estimado alumno =S:
La  calificación que has obtenido en el examen  de la asignatura
Máquinas de Turing es =C sobre un máximo de 10.
Atentamente.

El profesor
```

    Se trata de sustituir las marcas =S y =C por el nombre y por la nota respectivamente.

    Diseña un programa solicitando el nombre de cada alumno y la nota, personalizará el escrito y lo mostrará por pantalla, si lo deseamos repetirá el proceso para un nuevo alumno. Por ejemplo:

```
>>>
Nombre del alumno:  Alan Turing
Calificación del alumno: 9.9
Estimado alumno Alan Turing:
La  calificación que has obtenido en el examen  de la asignatura
Máquinas de Turing es 9.9 sobre un máximo de 10.
Atentamente.

El profesor
Si desea enviar otra carta, pulse c :
```

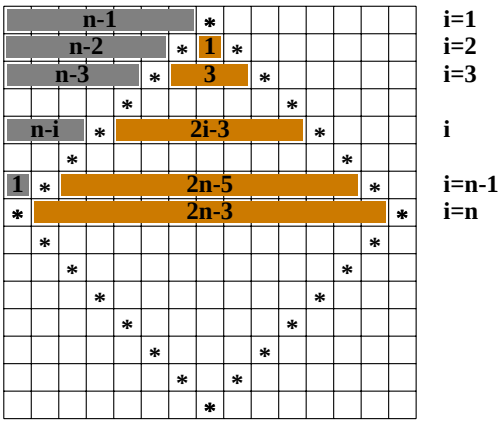
14. Hagamos ahora algunos dibujos:

```
$ python cuadrado.py
Lado: 1
*
$ python cuadrado.py
Lado: 2
* *
* *
$ python cuadrado.py
Lado: 3
* * *
*   *
* * *
$ python cuadrado.py
Lado: 6
* * * * * *
*         *
*         *
*         *
*         *
* * * * * *
```

```
$ python diamante.py
Lado: 1
*
$ python diamante.py
Lado: 2
  *
*   *
  *
$ python diamante.py
Lado: 5
        *
      *   *
    *       *
  *           *
*               *
  *           *
    *       *
      *   *
        *
```

Para dibujar los cuadrados, teniendo en cuenta que el mecanismo que utilizamos para escribir en la pantalla (`print`) escribe de arriba abajo, podemos repartir la tarea según el esquema de la izquierda.

- dibujar el lado superior
- dibujar los laterales ⟶
  - dibujar `*`
  - dibujar `*`
  - ⋮
  - dibujar `*`
- dibujar el lado inferior



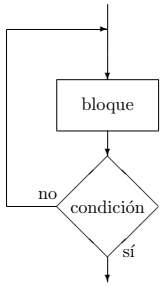
Para la otra figura, el análisis de los espacios en blanco que hay que escribir es más complejo. El gráfico de la derecha indica una solución posible.

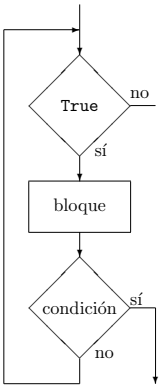
15. Escribe dos programas que se comporten como los de los ejemplos:

```
$ python aes.py
Palabra: abracadabra
Esa palabra tiene 5 aes minúsculas.
```

```
$ python letra.py
Palabra: tocomocho
Letra: o
La letra aparece 4 veces en la palabra.
```

En Python, al contrario que en otros lenguajes de programación, no contamos con una estructura de control que reproduzca el diagrama de la izquierda. Sin embargo, podemos lograr el mismo resultado combinando la sentencia **break** con un bucle **while**, en el que siempre se satisfaga la condición. La sentencia **break** interrumpe el bucle (y no solo la iteración en curso como **continue**).






```
while True:
    (bloque)
    if (condición):
        break
```