

UNIVERSIDAD DE ALCALÁ

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN



LABORATORIO FUNDAMENTOS DE LA PROGRAMACIÓN 3ª PARTE TEMA 1

M^a Ángeles Fdez de Sevilla Vellón

3. CONTINUACIÓN INSTRUCCIONES.

En Python encontramos dos tipos de bucles:

- Bucle for ... in.
- Bucles while. (Dependen de una condición, lo utilizaremos de dos formas diferentes, tal y como se vio en la parte teórica)

3.1. BUCLE FOR... IN.

Esta instrucción ejecuta el cuerpo de instrucciones del bucle un número de veces determinado. La utilizaremos cuando sabemos exactamente el número de veces que queremos que se ejecuten las instrucciones

- **Sintaxis:**

```
for variable in serie de valores:  
    instrucciones
```

Donde:

Nuevamente, las instrucciones deberán estar tabuladas respecto a la sentencia for, sangrado de 4 espacios (**indentación**).

NOTA: Es muy importante el sangrado, ya que si no lo hacemos correctamente el intérprete nos dará error a la hora de ejecutar el programa.

El bucle se ejecutará tantas veces como elementos tenga la series de valores.

Ejemplos:

1- Recorrer una cadena de texto usando for.

```
# Archivo: ejemplo17.py  
# Autor: XXXXX XXXXXXXX (nombre y apellidos)  
# Fecha: DD/MM/AAAA  
# Descripción: Recorrer una cadena de texto.  
  
# Programa principal
```

```
for letra in 'Hola!':  
    print ('Estamos en la letra :', letra)
```

```
# Salida por pantalla:  
Estamos en la letra : H  
Estamos en la letra : o  
Estamos en la letra : l  
Estamos en la letra : a  
Estamos en la letra : !
```

2- Realizar un programa que escriba por pantalla los 5 números siguientes al número 20.

```
# Archivo: ejemplo18.py  
# Autor: XXXXX XXXXXXXX (nombre y apellidos)  
# Fecha: DD/MM/AAAA  
# Descripción: Realizar un programa que escriba por  
# pantalla los 5 números siguientes al 20.  
  
#Constantes  
Maximo = 5  
Dato = 20  
  
#Programa principal  
print('Escribe los 5 números siguientes a 20: ')  
  
for i in range (1, Maximo+1):  
    print('El que va después es')  
    print(Dato + i)
```

3.2. BUCLE WHILE.

Este bucle repite un conjunto de instrucciones mientras se cumple una determinada condición que se evalúa al principio de cada ejecución. Las instrucciones del cuerpo del bucle, se ejecutará cero una o varias veces y dentro del bucle debe haber alguna instrucción que haga variar la condición para que en algún momento la condición deje de cumplirse.

➤ **Sintaxis:**

```
While condicion:  
  
    instruccion
```

Ejemplo: Calcula el factorial de un número introducido por teclado.

```
# Archivo: ejemplo15.py
# Autor: XXXXX XXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Calcula el factorial de un número
# introducido por teclado.

# Programa principal
numero = int(input('Dime un número para que te calcule el
factorial: '))
fac = 1
i = 1
while i < numero:
    i = i + 1
    fac = fac * i

print(fac)
```

3.3. BUCLE WHILE TRUE.

Este bucle se utiliza cuando al menos queremos que se ejecute una vez las instrucciones del cuerpo del bucle. La condición, es el equivalente al bucle DO WHILE en C++ o al bucle REPEAT... UNTIL en Pascal. Al igual que en el otro bucle condicional en el cuerpo debe existir alguna instrucción que varíe la condición para que en algún momento se pueda salir del bucle.

➤ **Sintaxis:**

```
while True:

    instrucción

    if condición:
        break
```

NOTA IMPORTANTE: Prohibido utilizar el break en lenguajes estructurados. En este curso solamente se utilizará para simular esta instrucción, no existente en Python.

Ejemplo: Imprime por pantalla sólo los números positivos.

```
# Archivo: ejemplo16.py
# Autor: XXXXX XXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Imprime por pantalla sólo los números
```

```
# positivos.  
  
# Programa principal  
while True:  
    entero = int(input('Introduzca entero positivo => '))  
    if entero >= 0:  
        print(entero)  
    else:  
        break
```

ENUNCIADOS EJERCICIOS



3ª PARTE TEMA 1:

CONTINUACIÓN INSTRUCCIONES: BUCLES

Mª Ángeles Fernández de Sevilla Vellón

3ª PARTE. TEMA 1:

ENUNCIADOS

1. Escribir un programa que calcule el producto de dos números enteros leídos desde el teclado sin utilizar el operador de multiplicación *.
2. Escribir un programa que lea un número real y un número entero y calcule el valor del primero elevado al segundo.
3. Escribir un programa que, después de preguntar cuantos números se van a introducir, pida esos números y devuelva su media aritmética, el mayor y el menor.
4. La sucesión de Fibonacci, $\{u_n\}$, está definida por $u_1 = 1$, $u_2 = 2$, $u_{n+2} = u_{n+1} + u_n$. Escribir un programa que halle los 20 primeros números de la sucesión y su suma.
5. Escribir un programa que lea de teclado tres caracteres hasta que se introduzca la palabra FIN (tanto con letras mayúsculas como con minúsculas).
6. Escribir un programa que lea un número entero positivo y:
 - a) determine cuántos dígitos se necesitan para escribir su valor.
 - b) calcule cuántas veces se puede dividir ese número por 2.
7. Escribir un programa que lea una serie de números enteros hasta que se introduzca el número -9999, y cuente el total de números introducidos, el total de valores positivos y el total de valores negativos.
8. Escribir un programa que pida un número *límite* y calcule cuántos términos de la serie armónica son necesarios para que su suma supere dicho límite, es decir, determine el menor número n tal que:

$$1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} > \text{límite}$$

9. Escribir un programa que vaya pidiendo por teclado una serie de números reales y calcule para cada uno de ellos su raíz cuadrada, indicando el final de datos mediante la introducción del número 0. Para calcular la raíz cuadrada se usará, en vez de la función estándar *sqrt*, el método de Newton, que establece:

Si *aprox* es una aproximación de $\sqrt{n}$,
entonces $(n/\text{aprox} + \text{aprox})/2$ es una aproximación mejor.

Detener el cálculo cuando $|n/\text{aprox}^2 - 1| < 10^{-6}$ o al sobrepasar 20 iteraciones. Comparar el resultado obtenido con el valor dado por la función *sqrt* de la librería *math*.

10. Escribir un programa que lea un entero positivo n y genere:
- a) una tabla con las n primeras potencias de 2.
 - b) una tabla con las potencias de 2 que son menores o iguales que n .
11. Una pelota cae inicialmente desde una altura de dos metros. Si en cada bote la pelota sube hasta una altura igual al 95% de la que cayó en el bote anterior, hallar la distancia vertical recorrida en un total de 10 botes. ¿Cuántos botes deberá dar para que su altura sea menor de cinco centímetros?

Funciones predefinidas necesarias para afrontar los ejercicios 3ª Parte Tema1

Mayor y menor entero en Python: Para obtener el mayor número de tipo entero que Python admite hay que 1º invocar a la librería sys y posteriormente el mayor entero se encontrará en la constante sys.maxsize

```
>>> import sys
>>> a = sys.maxsize
>>> print(a) : 2147483647 # Mayor Entero Python
>>> print (-a) : -2147483647 # Menor Entero Python
```

abs(n) → Sea un número real ó entero devuelve su valor absoluto.

n1 / n2 → / es un operador que devuelve la división entera entre los números n1 y n2.

ex** → Sea x un número real, devuelve otro real correspondiente a la exponenciación de x (el número e se encuentra disponible en librería math e^x).

log(n) → Sea n un número real log(n) devuelve el logaritmo neperiano de n (disponible en librería math)

log10(n) → Sea n un número real log10(n) devuelve el logaritmo en base 10 de n (disponible en librería math)

upper() → Sea c una cadena c.upper() devuelve la cadena c introducida en mayúsculas. >>> cad = 'asdas'
>>> print(cad.upper())
ASDAS

lower() → Sea c una cadena c.lower() devuelve la cadena c introducida en minúsculas. >>> c= 'SOY EL MEJOR'
>>> print(c.lower())
soy el mejor