Entrada y Salida

Lectura de datos de teclado. Raw_input()

Podemos leer datos desde el teclado, de forma interactiva. La forma de hacerlo es utilizar la función **raw_input()**.

Esta función hace lo siguiente: detiene la ejecución del programa y espera a que el usuario escriba un texto y pulse la tecla de retorno de carro(enter); en ese momento prosigue la ejecución y la función devuelve una cadena con el texto que tecleó el usuario.

En principio, los datos leídos forman una cadena de caracteres. Si deseas que el dato sea un valor flotante, debes transformar la cadena devuelta por raw_input utilizando la función **float**. Lo mismo con los valores enteros, utilizando la función **int**.

La función raw_input acepta un argumento: una cadena con el mensaje que debe mostrar.

Posibles llamadas:

- `x = raw_input()`
- `x = float(raw_input())`
- `x = int(raw_intput())`
- `x = raw_input('Dato = ')` (Introducimos un mensaje al usuario)

In [1]:
```
x = float(raw_input('Introduce un dato: '))
print "El dato es", x
```

```
Introduce un dato: 4.2
El dato es 4.2
```

La orden `input()` se comporta de manera similar, pero hay diferencias en la gestión de errores que hacen más adecuada la utilización de `raw_input()`.

Consider using the raw_input() function for general input from users. (https://docs.python.org/2/library/functions.html?highlight=raw_input#input)

Presentación de datos por pantalla: `print`

Para presentar algo por pantalla, basta utilizar la instrucción `print`

In [2]:
```
a = 5
print a
```

```
5
```

Podemos presentar varios datos separados por comas.

In [3]:
```
print a, 2*a, a*a
```

```
5 10 25
```

También podemos presentar mensajes de texto, entre comillas.

In [4]:
```
print a,'al cuadrado es',a*a
```

```
5 al cuadrado es 25
```

Podemos evitar el salto de línea con una coma al final.

Salida con formato

Para mejorar las presentaciones, podemos dar distintos formatos a la salida.

In [5]:
```
a = 0.5
i = 1
while i < 11:
    print a,'elevado a',i,'es',a**i
    i+=1
```

```
0.5 elevado a 1 es 0.5
0.5 elevado a 2 es 0.25
0.5 elevado a 3 es 0.125
0.5 elevado a 4 es 0.0625
0.5 elevado a 5 es 0.03125
0.5 elevado a 6 es 0.015625
0.5 elevado a 7 es 0.0078125
0.5 elevado a 8 es 0.00390625
0.5 elevado a 9 es 0.001953125
0.5 elevado a 10 es 0.0009765625
```

In [6]:

```
a = 0.5
i = 1
while i < 11:
    print '%f elevado a %d es %f' %(a,i,a**i)
    i+=1
```

```
0.500000 elevado a 1 es 0.500000
0.500000 elevado a 2 es 0.250000
0.500000 elevado a 3 es 0.125000
0.500000 elevado a 4 es 0.062500
0.500000 elevado a 5 es 0.031250
0.500000 elevado a 6 es 0.015625
0.500000 elevado a 7 es 0.007812
0.500000 elevado a 8 es 0.003906
0.500000 elevado a 9 es 0.001953
0.500000 elevado a 10 es 0.000977
```

In [7]:
```
a = 0.85
i = 1
while i < 11:
    print '%4.2f elevado a %2d es %5.3f' %(a,i,a**i)
    i+=1
```

```
0.85 elevado a  1 es 0.850
0.85 elevado a  2 es 0.722
0.85 elevado a  3 es 0.614
0.85 elevado a  4 es 0.522
0.85 elevado a  5 es 0.444
0.85 elevado a  6 es 0.377
0.85 elevado a  7 es 0.321
0.85 elevado a  8 es 0.272
0.85 elevado a  9 es 0.232
0.85 elevado a 10 es 0.197
```

Aparte de %d y %f existe la marca %s para cadenas

In [8]:
```
c = 'X'
i = 0
while i < 10:
    print 'La cadena es', c
    c = c + 'Y'
    i += 1
```

```
La cadena es X
La cadena es XY
La cadena es XYY
La cadena es XYYY
La cadena es XYYYY
La cadena es XYYYYY
La cadena es XYYYYYY
La cadena es XYYYYYYY
La cadena es XYYYYYYYY
La cadena es XYYYYYYYYY
```

In [9]:

```
c = 'X'
i = 0
while i < 10:
    print 'La cadena es %10s' %(c)
    c = c + 'Y'
    i += 1
```

```
La cadena es          X
La cadena es         XY
La cadena es        XYY
La cadena es       XYYY
La cadena es      XYYYY
La cadena es     XYYYYY
La cadena es    XYYYYYY
La cadena es   XYYYYYYY
La cadena es  XYYYYYYYY
La cadena es XYYYYYYYYY
```

Ejemplos

In [2]:

```
from math import pi
def circle_length(radius):
    return 2*pi*radius
def main():
    #SALIDA CON FORMATO
    n = int(raw_input("n = "))
    #presentamos por pantalla sin ningún tipo de formato
    for k in range(2, 11):
        print n, "elevado a", k, "es igual a", n**k
    print '='*10
    #presentamos por pantalla utilizando un formato de salida
    for k in range(2, 11):
        print '%d elevado a %d es igual a %d' % (n, k, n**k)
    print '='*10
    '''
    %<numero>d --> formato enteros
    %<numero1.numero2>f --> formato flotantes
    %s --> cadenas
    '''
    #Con un buen formato, mejoramos la presentación
    for k in range(2, 11):
        print '%d elevado a %2d es igual a %9d' % (n, k, n**k)
    print '='*10
    #Podemos mezclar reales y enteros
    for k in range(1, 10):
        print 'la circunferencia de radio %d tiene longitud %f' % (
k, circle_length(k))
    print '='*10

    for k in range(1, 10):
        print 'la circunferencia de radio %d tiene longitud %5.1f'
% (k, circle_length(k))
    print '='*10
    for k in range(1, 10):
        r = 1.0/k
        print 'la circunferencia de radio %4.2f tiene longitud %5.1
f' % (r, circle_length(r))
```

In [3]:
```
main()
```

```
n = 4
4 elevado a 2 es igual a 16
4 elevado a 3 es igual a 64
4 elevado a 4 es igual a 256
4 elevado a 5 es igual a 1024
4 elevado a 6 es igual a 4096
4 elevado a 7 es igual a 16384
4 elevado a 8 es igual a 65536
4 elevado a 9 es igual a 262144
4 elevado a 10 es igual a 1048576
==========
4 elevado a 2 es igual a 16
4 elevado a 3 es igual a 64
4 elevado a 4 es igual a 256
4 elevado a 5 es igual a 1024
4 elevado a 6 es igual a 4096
4 elevado a 7 es igual a 16384
4 elevado a 8 es igual a 65536
4 elevado a 9 es igual a 262144
4 elevado a 10 es igual a 1048576
==========
4 elevado a  2 es igual a        16
4 elevado a  3 es igual a        64
4 elevado a  4 es igual a       256
4 elevado a  5 es igual a      1024
4 elevado a  6 es igual a      4096
4 elevado a  7 es igual a     16384
4 elevado a  8 es igual a     65536
4 elevado a  9 es igual a    262144
4 elevado a 10 es igual a   1048576
==========
la circunferencia de radio 1 tiene longitud 6.283185
la circunferencia de radio 2 tiene longitud 12.566371
la circunferencia de radio 3 tiene longitud 18.849556
la circunferencia de radio 4 tiene longitud 25.132741
la circunferencia de radio 5 tiene longitud 31.415927
la circunferencia de radio 6 tiene longitud 37.699112
la circunferencia de radio 7 tiene longitud 43.982297
la circunferencia de radio 8 tiene longitud 50.265482
la circunferencia de radio 9 tiene longitud 56.548668
==========
la circunferencia de radio 1 tiene longitud   6.3
la circunferencia de radio 2 tiene longitud  12.6
la circunferencia de radio 3 tiene longitud  18.8
la circunferencia de radio 4 tiene longitud  25.1
la circunferencia de radio 5 tiene longitud  31.4
la circunferencia de radio 6 tiene longitud  37.7
la circunferencia de radio 7 tiene longitud  44.0
la circunferencia de radio 8 tiene longitud  50.3
la circunferencia de radio 9 tiene longitud  56.5
==========
la circunferencia de radio 1.00 tiene longitud   6.3
la circunferencia de radio 0.50 tiene longitud   3.1
la circunferencia de radio 0.33 tiene longitud   2.1
la circunferencia de radio 0.25 tiene longitud   1.6
```

In [4]:
```
#Lectura de una lista desde el teclado
def read_list(size):
    '''
    Read a list from keyboard
    '''
    i = 0
    ll = []
    while i < size:
        element = raw_input('Input element: ')
        ll.append(element)
        i += 1
    return ll
#Mostrar una lista en orden inverso
def inverse_output(mylist):
    '''
    Prints on screen a list in inverse order
    '''
    i = len(mylist) - 1
    while i >= 0 :
        print "Element %d is %s" %(i,mylist[i])
        i -= 1
```

In [6]:
```
n = int(raw_input('Size = '))
ll = read_list(n)
inverse_output(ll)
```

```
Size = 5
Input element: AC/DC
Input element: Kiss
Input element: Iron Maiden
Input element: Judas Priest
Input element: Queen
Element 4 is Queen
Element 3 is Judas Priest
Element 2 is Iron Maiden
Element 1 is Kiss
Element 0 is AC/DC
```

In []: