Python trae consigo ciertas funciones, llamadas funciones integradas, esto es, no es necesario cargarlas desde ningún módulo, abs(). Dichas funciones dependen de la versión Python que tengamos instalada en nuestro sistema.Las encontramos en: [http://docs.python.org/library/functions.html](https://docs.python.org/3.3/library/functions.html)

* *float()*: conversión a flotante. Si recibe un número entero como argumento, devuelve el mismo número convertido en un flotante equivalente.

>>> float(3)

3.0

La función *float* también acepta argumentos de tipo cadena. Cuando se le pasa una cadena, *float* la convierte en el número flotante que ésta representa:

>>> float(’3.2’)

3.2

>>> float(’3.2e10’)

32000000000.0

Pero si la cadena no representa un flotante, se produce un error de tipo ValueError, es decir, ˂˂error de valor˃˃:

>>> float(’un texto’)

Traceback (innermost last):

File "<stdin>", line 1, in ?

ValueError: invalid literal for float(): un texto

Si *float* recibe un argumento flotante, devuelve el mismo valor que se suministra como argumento.

* *int()*: conversión a entero. Si recibe un número flotante como argumento, devuelve el entero que se obtiene eliminando la parte fraccionaria.

>>> int(2.1)

2

>>> int(-2.9)

-2

También la función *int* acepta como argumento una cadena:

>>> int(’2’)

2

Si *int* recibe un argumento entero, devuelve el argumento tal cual.

* *str()*: conversión a cadena. Recibe un número y devuelve una representación de éste como cadena.

>>> str(2.1)

’2.1’

>>> str(234E47)

’2.34e+47’

La función *str()* también puede recibir como argumento una cadena, pero en ese caso devuelve como resultado la misma cadena.

* *round()*: redondeo. Puede usarse con uno o dos argumentos. Si se usa con un sólo argumento, redondea el número al flotante más próximo cuya parte decimal sea nula.

>>> round(2.1)

2.0

>>> round(2.9)

3.0

>>> round(-2.9)

-3.0

>>> round(2)

2.0

El redondeo de *int()* puede ser al alza o a la baja según el ordenador en que lo ejecutes. Si quieres un comportamiento homogéneo del redondeo, utiliza las funciones *round*, *floor* o *ceil*,.

(¡Observa que el resultado siempre es de tipo flotante!) Si *round* recibe dos argumentos, ´estos deben ir separados por una coma y el segundo indica el número de decimales que deseamos conservar tras el redondeo.

>>> round(2.1451, 2)

2.15

>>> round(2.1451, 3)

2.145

>>> round(2.1451, 0)

2.0

Estas funciones y otras pueden formar parte de expresiones y sus argumentos pueden, a su vez, ser expresiones. Observa los siguientes ejemplos:

>>> abs(-23) % int(7.3)

2

>>> abs(round(-34.2765,1))

34.3

>>> str(float(str(2) \* 3 + ’.123’)) + ’321’

222.123321

1. Pedir un número con decimales y calcula con una única expresión el valor absoluto del redondeo del número.

Teclea un número:-3.2

Número: -3.2 Valor absoluto del redondeo es: 3.0

Teclea un número:-5.6

Número: -5.6 Valor absoluto del redondeo es: 6.0

1. Leer dos números enteros y convierte (en una única expresión) a una cadena el resultado de la división 5011/10000 redondeado con 3 decimales.

Dividendo: 5011

Divisor: 10000

5011 / 10000 = 0.501

1. ¿Qué resulta de evaluar estas expresiones? Haz un módulo que lo demuestre

str(2.1) + str(1.2)

int(str(2) + str(3))

str(int(12.3)) + ’0’

int(’2’+’3’)

str(2 + 3)

str(int(2.1) + float(3))