INFORMÁTICA, GRUPO D

Asignaciones con operador

Podemos simplificar las operaciones cuando hay una única variable

In [50]:
```
a = 10
a = a + 1
a += 1
a
```

Out[50]: 12

Se utiliza para todos los operadores

In [52]:
```
b = 10
b *= 5 # es lo mismo que b = b * 5
b
```

Out[52]: 50

In [53]:
```
c = 10
c **= 6
c
```

Out[53]: 1000000

El tipo de datos cadena

En muchos lenguajes se le llama *string*

Permite guardar cadenas de símbolos

In [54]:
```
a = 'hola'
a
```

Out[54]: 'hola'

Las operaciones básicas con la concatenación (+) y la repetición (*)

In [55]:
```
a = a + a
```

In [56]:
```
a
```

Out[56]: 'holahola'

In [58]:
```
b = '-'*10
b
```

Out[58]: '----------'

In [59]:
```
#cuidado con los espacios
b + ' '*6 + b
```

Out[59]:
```
'----------      ----------'
```

CUIDADO: Una cadena no es un identificador

In [60]:
```
hola = 3.14
```

In [61]:
```
hola == 'hola'
```

Out[61]:
```
False
```

In [62]:
```
hola + hola
```

Out[62]:
```
6.28
```

In [63]:
```
'hola' + 'hola'
```

Out[63]:
```
'holahola'
```

In [64]:
```
hola + 'hola'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/home/jesus/Dropbox/docencia13-14/ipython/<ipython-input-64-db7e7bbc7041> in <module>()
----> 1 hola + 'hola'

TypeError: unsupported operand type(s) for +: 'float' and 'str'
```

In [65]:
```
'12'+'12'
```

Out[65]:
```
'1212'
```

A los caracteres individuales a veces se les llama de tipo *char*. Tienen operadores propios.

In [66]:
```
a = 'a'
ord(a)
```

Out[66]:
```
97
```

In [67]:
```
chr(98)
```

Out[67]:
```
'b'
```

In [68]:
```
#Se respeta el orden alfabético
'adios'>'burro'
```

Out[68]:
```
False
```

Funciones predefinidas

Python tiene una serie de funciones predefinidas.

abs: valor absoluto

In [69]:
```
abs(-4)
```

Out[69]: 4

float: conversión a flotante. Acepta enteros y cadenas.

In [70]:
```
float(3)
```

Out[70]: 3.0

In [71]:
```
float('3.34')
```

Out[71]: 3.34

In [72]:
```
float('3.4e11')
```

Out[72]: 340000000000.0

In [74]:
```
float('3k10')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/home/jesus/Dropbox/docencia13-14/ipython/<ipython-input-74-9f7d72827442> in <module>()
----> 1 float('3k10')

ValueError: invalid literal for float(): 3k10
```

int: conversión a entero. Acepta flotantes y cadenas.

In [75]:
```
int('29')
```

Out[75]: 29

In [76]:
```
int(3.1)
```

Out[76]: 3

In [77]:
```
int(3.9)
```

Out[77]: 3

str: conversión a cadena. Recibe un número y devuelve una representación como cadena.

In [78]:
```
str(10)
```

Out[78]: '10'

In [79]:
```
str(3.1e4)
```

Out[79]: '31000.0'

round: redondeo. Puede usarse con uno o dos argumentos. Si se usa con un argumento, redondea el número al flotante más próximo o cuya

parte decimal sea nula. (¡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 queremos conservar tras el redondeo.

In [80]:
```
round(10.3)
```

Out[80]: 10.0

In [81]:
```
round(10.8)
```

Out[81]: 11.0

In [82]:
```
a = 45.99893843959393
a **= 2
```

In [83]:
```
a
```

Out[83]: 2115.902337569552

In [84]:
```
round(a,4)
```

Out[84]: 2115.9023

El módulo math

Podemos utilizar numerosas funciones matemáticas utilizando, importándolas del módulo math.

In [85]:
```
from math import sin, cos
from math import pi
```

In [86]:
```
sin(pi)
```

Out[86]: 1.2246063538223773e-16

In [88]:
```
cos(pi/2) #esto es cero, no?
```

Out[88]: 6.123031769111886e-17

Más sencillo: podemos importar todo

In [89]:
```
from math import *
```

- sin(x), Seno de x, expresado en radianes.
- cos(x), Coseno de x, expresado en radianes.
- tan(x), Tangente de x, expresado en radianes.
- exp(x), el número e elevado a x.
- ceil(x), Redondeo hacia arriba de x.
- floor(x), Redondeo hacia abajo de x.
- log(x), Logaritmo en base e de x.
- log10(x), Logaritmo en base 10 de x.
- sqrt(x), Raíz cuadrada de x.

También se definen las constantes pi y e.

In [46]:
```
pi
```

Out[46]: 3.141592653589793

In [42]:
```
e
```

Out[42]: 2.718281828459045

In [43]:
```
floor(pi)
```

Out[43]: 3.0

In [44]:
```
ceil(pi)
```

Out[44]: 4.0

Cómo crear mis propias funciones

In [90]:
```
def cuadrado(x):
    return x**2
```

In [91]:
```
cuadrado(5)
```

Out[91]: 25

In [92]:
```
def esCuadradoPerfecto(n):
    m = int(sqrt(n))
    return m*m == n
```

In [93]:
```
esCuadradoPerfecto(30)
```

Out[93]: False

In [94]:
```
esCuadradoPerfecto(25)
```

Out[94]: True

In [95]:
```
def cuadradoPrevio(n):
    m = int(sqrt(n))
    return m**2
```

In [96]:
```
cuadradoPrevio(30)
```

Out[96]: 25

In [97]:
```
cuadradoPrevio(25)
```

Out[97]: 25

In [97]:

In [ ]: