3. Libreri?as

May 5, 2014

1 Librerías / Bibliotecas / Paquetes / Módulos

Una biblioteca es una colección de funciones ya hechas.

En Python, es literalmente el caso: una biblioteca es una colección de funciones definidas al estilo de Python (con `def`) en un archivo con terminación `.py`

```
In [14]: %%file raices.py
         # escribe a un archivo
         def raiz(y):
             """
             Calcula la raiz cuadrada de un numero con el algoritmo Babilonico
             """
             x0 = y
             x = x0
             eps = 1e-10

             while abs(x*x - y) > eps:
                 x = 0.5 * (x + y/x)

             return x

         def cuad(y):
             return y*y


         if __name__=="__main__":
             # codigo que se ejecuta solo si corro el programa como programa principal
             # con 'python raices.py'
             print(cuad(10))
```

```
Overwriting raices.py
```

Los comandos internos de IPython (comandos *mágicos*) comienzan con `%`:

- `%algo`: mágicos de línea; operan sobre de el contenido sólo de una línea
- `%%algo`: mágicos de celda; operan sobre el contenido de una celda completa

```
In [2]: %who
        # muestra los nombres actualmente presentes
```

```
Interactive namespace is empty.
```

```
In [3]: %whos
        # mas informacion
```

```
Interactive namespace is empty.
```

Cómo jalar el contenido de una biblioteca:

```
In [4]: import raices
```

Qué hace este comando? Cuál es su efecto?

```
In [5]: %who

raices

In [6]: import raices

In [7]: %who

raices

In [8]: raices.


          File "<ipython-input-8-0f28cde6bb6f>", line 1
        raices.
               ^
    SyntaxError: invalid syntax


In [9]: raices.raiz(10)

Out[9]: 3.162277660168379

In [10]: raices.cuad(10)

Out[10]: 100
```

Para llamar a un programa desde la terminal (el shell), uso `!`

```
In [15]: !python raices.py

100
```

Para correr desde adentro de IPython

```
In [16]: %run raices.py

100
```

1.1 Bibliotecas pre-definidas

Hay miles. Algunos útiles:

```
In [17]: import math
         # biblio de la biblio estándar

In [18]: %who

cuad        math        raices        raiz

In [20]: dir(math)  # directorio
```

```
Out[20]: ['__doc__',
          '__file__',
          '__name__',
          '__package__',
          'acos',
          'acosh',
          'asin',
          'asinh',
          'atan',
          'atan2',
          'atanh',
          'ceil',
          'copysign',
          'cos',
          'cosh',
          'degrees',
          'e',
          'erf',
          'erfc',
          'exp',
          'expm1',
          'fabs',
          'factorial',
          'floor',
          'fmod',
          'frexp',
          'fsum',
          'gamma',
          'hypot',
          'isinf',
          'isnan',
          'ldexp',
          'lgamma',
          'log',
          'log10',
          'log1p',
          'modf',
          'pi',
          'pow',
          'radians',
          'sin',
          'sinh',
          'sqrt',
          'tan',
          'tanh',
          'trunc']
```

`math.<TAB>` en IPython:
`math` es una copia / envoltura ('wrapper') de la biblioteca `math.h` de C

```
In [21]: math.sin(1)
```

```
Out[21]: 0.8414709848078965
```

Si necesitamos usar muchas veces estas funciones, tal vez no queramos teclear `math` todo el tiempo.

```
In [22]: sin

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)

<ipython-input-22-fd33a234f1c4> in <module>()
----> 1 sin


NameError: name 'sin' is not defined


In [23]: sin = math.sin

In [24]: %who

cuad        math        raices        raiz        sin

In [25]: sin(1)

Out[25]: 0.8414709848078965

In [26]: m = math

In [27]: m

Out[27]: <module 'math' from '/usr/local/Cellar/python/2.7.5/Frameworks/Python.framework/Versions/2.7/li

In [28]: from math import sin, cos, exp

In [29]: %who

cos        cuad        exp        m        math        raices        raiz        sin
```

Es posible hacer:

```
In [30]: from math import *

In [31]: %who

acos        acosh        asin        asinh        atan        atan2        atanh        ceil        copy
cos        cosh        cuad        degrees        e        erf        erfc        exp        expm1
fabs        factorial        floor        fmod        frexp        fsum        gamma        hypot
isnan        ldexp        lgamma        log        log10        log1p        m        math        modf
pi        pow        radians        raices        raiz        sin        sinh        sqrt        tan
tanh        trunc

In [33]: import math as m
         # equivalente:    import math; m=math
```

Para "ir haciendo" algo en Python:

```
In [34]: tabla = []  # empezar con lista vacia
         for i in xrange(10):  # adentro de un bucle
             tabla.append(raices.raiz(i))   # ir agregando cosas a la lista

In [35]: tabla
```

Out[35]: [0,
 1,
 1.4142135623746899,
 1.7320508075688772,
 2.000000000000002,
 2.236067977499978,
 2.4494897427875517,
 2.6457513110645907,
 2.82842712474619,
 3.0]

In []: