Bibliothèques externes sous Python

Source :

- Revue Coding, "Apprenez à travailler avec des bibliothèques externes sous Python", Hors série n°16, p.50-53.

Exemple : le module math (bibliothèque standard)

In [1]:

```
import math
print("Le module math a un type {}".format(type(math)))
```

```
Le module math a un type <class 'module'>
```

Math est un module. Un module est juste une collection de variables définies par une personne qui l'a développé.

Fonction `dir()`

In [3]:

```
print(dir(math))
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh',
'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copysign', 'cos', 'cos
h', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'f
loor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfini
te', 'isinf', 'isnan', 'isqrt', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log
2', 'modf', 'nan', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'si
nh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
```

Exemple :

In [4]:

```
# valeur simple
math.pi
```

Out[4]:

```
3.141592653589793
```

In [5]:

```
# fonction
math.log(32, 2)
```

Out[5]:

```
5.0
```

Fonction `help()`

In [6]:

```
# détaille la fonction indiquée
help(math.log)
```

```
Help on built-in function log in module math:

log(...)
    log(x, [base=math.e])
    Return the logarithm of x to the given base.

    If the base not specified, returns the natural logarithm (base e) of x.
```

In [8]:

```
# détaille le module lui-même
help(math)
```

```
Help on built-in module math:

NAME
    math

DESCRIPTION
    This module provides access to the mathematical functions
    defined by the C standard.

FUNCTIONS
    acos(x, /)
        Return the arc cosine (measured in radians) of x.

    acosh(x, /)
        Return the inverse hyperbolic cosine of x.

    asin(x, /)
        Return the arc sine (measured in radians) of x.

    asinh(x, /)
        Return the inverse hyperbolic sine of x.

    atan(x, /)
        Return the arc tangent (measured in radians) of x.

    atan2(y, x, /)
        Return the arc tangent (measured in radians) of y/x.

        Unlike atan(y/x), the signs of both x and y are considered.

    atanh(x, /)
        Return the inverse hyperbolic tangent of x.

    ceil(x, /)
        Return the ceiling of x as an Integral.

        This is the smallest integer >= x.

    comb(n, k, /)
        Number of ways to choose k items from n items without repetition and witho
ut order.

        Evaluates to n! / (k! * (n - k)!) when k <= n and evaluates
        to zero when k > n.

        Also called the binomial coefficient because it is equivalent
        to the coefficient of k-th term in polynomial expansion of the
        expression (1 + x)**n.

        Raises TypeError if either of the arguments are not integers.
        Raises ValueError if either of the arguments are negative.

    copysign(x, y, /)
        Return a float with the magnitude (absolute value) of x but the sign of y.

        On platforms that support signed zeros, copysign(1.0, -0.0)
        returns -1.0.

    cos(x, /)
        Return the cosine of x (measured in radians).

    cosh(x, /)
        Return the hyperbolic cosine of x.
    degrees(x, /)
        Convert angle x from radians to degrees.

    dist(p, q, /)
        Return the Euclidean distance between two points p and q.

        The points should be specified as sequences (or iterables) of
        coordinates.  Both inputs must have the same dimension.

        Roughly equivalent to:
            sqrt(sum((px - qx) ** 2.0 for px, qx in zip(p, q)))

    erf(x, /)
        Error function at x.

    erfc(x, /)
        Complementary error function at x.

    exp(x, /)
        Return e raised to the power of x.

    expm1(x, /)
        Return exp(x)-1.

        This function avoids the loss of precision involved in the direct evaluati
on of exp(x)-1 for small x.

    fabs(x, /)
        Return the absolute value of the float x.

    factorial(x, /)
        Find x!.

        Raise a ValueError if x is negative or non-integral.

    floor(x, /)
        Return the floor of x as an Integral.

        This is the largest integer <= x.

    fmod(x, y, /)
        Return fmod(x, y), according to platform C.

        x % y may differ.

    frexp(x, /)
        Return the mantissa and exponent of x, as pair (m, e).

        m is a float and e is an int, such that x = m * 2.**e.
        If x is 0, m and e are both 0.  Else 0.5 <= abs(m) < 1.0.

    fsum(seq, /)
        Return an accurate floating point sum of values in the iterable seq.

        Assumes IEEE-754 floating point arithmetic.

    gamma(x, /)
        Gamma function at x.

    gcd(x, y, /)
        greatest common divisor of x and y

    hypot(...)
        hypot(*coordinates) -> value
        Multidimensional Euclidean distance from the origin to a point.

        Roughly equivalent to:
            sqrt(sum(x**2 for x in coordinates))

        For a two dimensional point (x, y), gives the hypotenuse
        using the Pythagorean theorem:  sqrt(x*x + y*y).

        For example, the hypotenuse of a 3/4/5 right triangle is:

            >>> hypot(3.0, 4.0)
            5.0

    isclose(a, b, *, rel_tol=1e-09, abs_tol=0.0)
        Determine whether two floating point numbers are close in value.

          rel_tol
            maximum difference for being considered "close", relative to the
            magnitude of the input values
          abs_tol
            maximum difference for being considered "close", regardless of the
            magnitude of the input values

        Return True if a is close in value to b, and False otherwise.

        For the values to be considered close, the difference between them
        must be smaller than at least one of the tolerances.

        -inf, inf and NaN behave similarly to the IEEE 754 Standard.  That
        is, NaN is not close to anything, even itself.  inf and -inf are
        only close to themselves.

    isfinite(x, /)
        Return True if x is neither an infinity nor a NaN, and False otherwise.

    isinf(x, /)
        Return True if x is a positive or negative infinity, and False otherwise.

    isnan(x, /)
        Return True if x is a NaN (not a number), and False otherwise.

    isqrt(n, /)
        Return the integer part of the square root of the input.

    ldexp(x, i, /)
        Return x * (2**i).

        This is essentially the inverse of frexp().

    lgamma(x, /)
        Natural logarithm of absolute value of Gamma function at x.

    log(...)
        log(x, [base=math.e])
        Return the logarithm of x to the given base.

        If the base not specified, returns the natural logarithm (base e) of x.

    log10(x, /)
        Return the base 10 logarithm of x.

    log1p(x, /)
        Return the natural logarithm of 1+x (base e).
        The result is computed in a way which is accurate for x near zero.

    log2(x, /)
        Return the base 2 logarithm of x.

    modf(x, /)
        Return the fractional and integer parts of x.

        Both results carry the sign of x and are floats.

    perm(n, k=None, /)
        Number of ways to choose k items from n items without repetition and with
order.

        Evaluates to n! / (n - k)! when k <= n and evaluates
        to zero when k > n.

        If k is not specified or is None, then k defaults to n
        and the function returns n!.

        Raises TypeError if either of the arguments are not integers.
        Raises ValueError if either of the arguments are negative.

    pow(x, y, /)
        Return x**y (x to the power of y).

    prod(iterable, /, *, start=1)
        Calculate the product of all the elements in the input iterable.

        The default start value for the product is 1.

        When the iterable is empty, return the start value.  This function is
        intended specifically for use with numeric values and may reject
        non-numeric types.

    radians(x, /)
        Convert angle x from degrees to radians.

    remainder(x, y, /)
        Difference between x and the closest integer multiple of y.

        Return x - n*y where n*y is the closest integer multiple of y.
        In the case where x is exactly halfway between two multiples of
        y, the nearest even value of n is used. The result is always exact.

    sin(x, /)
        Return the sine of x (measured in radians).

    sinh(x, /)
        Return the hyperbolic sine of x.

    sqrt(x, /)
        Return the square root of x.

    tan(x, /)
        Return the tangent of x (measured in radians).

    tanh(x, /)
        Return the hyperbolic tangent of x.

    trunc(x, /)
        Truncates the Real x to the nearest Integral toward 0.
        Uses the __trunc__ magic method.

DATA
    e = 2.718281828459045
    inf = inf
    nan = nan
    pi = 3.141592653589793
    tau = 6.283185307179586

FILE
    (built-in)
```

Abréviations lors d'importation

In [9]:

```
import math as mt
import pandas as pd
import numpy as np
```

In [10]:

```
mt.pi
```

Out[10]:

```
3.141592653589793
```

Tout importer

In [12]:

```
from math import *

print(pi, log(32, 2))
```

```
3.141592653589793 5.0
```

A éviter : risque d'erreur si deux modules ont le même nom de fonction. Exemple numpy et math avec la fonction **log**.

Importer uniquement ce qui est nécessaire

In [13]:

```
from math import log, pi
from numpy import asarray
```

Sous-modules

In [24]:

```
import numpy

print(numpy.random.randint)
print(type(numpy.random))
print(type(numpy.random.randint))
```

```
<built-in method randint of numpy.random.mtrand.RandomState object at 0x0000022537
29A540>
<class 'module'>
<class 'builtin_function_or_method'>
```

In [22]:

```
a = numpy.random.randint(low=1, high=6, size=10)
a
```

Out[22]:

```
array([4, 4, 1, 3, 5, 4, 2, 5, 3, 5])
```

In [25]:

```
# savoir ce dont il s'agit
type(a)
```

Out[25]:

```
numpy.ndarray
```

In [27]:

```
# savoir ce qu'on peut faire avec
print(dir(a))
```

```
['T', '__abs__', '__add__', '__and__', '__array__', '__array_finalize__', '__array
_function__', '__array_interface__', '__array_prepare__', '__array_priority__', '_
_array_struct__', '__array_ufunc__', '__array_wrap__', '__bool__', '__class__', '_
_complex__', '__contains__', '__copy__', '__deepcopy__', '__delattr__', '__delitem
__', '__dir__', '__divmod__', '__doc__', '__eq__', '__float__', '__floordiv__', '_
_format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__
iadd__', '__iand__', '__ifloordiv__', '__ilshift__', '__imatmul__', '__imod__', '_
_imul__', '__index__', '__init__', '__init_subclass__', '__int__', '__invert__',
'__ior__', '__ipow__', '__irshift__', '__isub__', '__iter__', '__itruediv__', '__i
xor__', '__le__', '__len__', '__lshift__', '__lt__', '__matmul__', '__mod__', '__m
ul__', '__ne__', '__neg__', '__new__', '__or__', '__pos__', '__pow__', '__radd__',
'__rand__', '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__', '__rfloordiv
__', '__rlshift__', '__rmatmul__', '__rmod__', '__rmul__', '__ror__', '__rpow__',
'__rrshift__', '__rshift__', '__rsub__', '__rtruediv__', '__rxor__', '__setattr_
_', '__setitem__', '__setstate__', '__sizeof__', '__str__', '__sub__', '__subclass
hook__', '__truediv__', '__xor__', 'all', 'any', 'argmax', 'argmin', 'argpartitio
n', 'argsort', 'astype', 'base', 'byteswap', 'choose', 'clip', 'compress', 'conj',
'conjugate', 'copy', 'ctypes', 'cumprod', 'cumsum', 'data', 'diagonal', 'dot', 'dt
ype', 'dump', 'dumps', 'fill', 'flags', 'flat', 'flatten', 'getfield', 'imag', 'it
em', 'itemset', 'itemsize', 'max', 'mean', 'min', 'nbytes', 'ndim', 'newbyteorde
r', 'nonzero', 'partition', 'prod', 'ptp', 'put', 'ravel', 'real', 'repeat', 'resh
ape', 'resize', 'round', 'searchsorted', 'setfield', 'setflags', 'shape', 'size',
'sort', 'squeeze', 'std', 'strides', 'sum', 'swapaxes', 'take', 'tobytes', 'tofil
e', 'tolist', 'tostring', 'trace', 'transpose', 'var', 'view']
```

In [31]:

```
# afficher l'aide en ligne
help(a.ravel)

# help(a) = aide très longue
# ... mieux vaut la consulter en ligne.
```

```
Help on built-in function ravel:

ravel(...) method of numpy.ndarray instance
    a.ravel([order])

    Return a flattened array.

    Refer to `numpy.ravel` for full documentation.

    See Also
    --------
    numpy.ravel : equivalent function

    ndarray.flat : a flat iterator on the array.
```

Surcharge de l'opérateur

In [32]:

```
a
```

Out[32]:

```
array([4, 4, 1, 3, 5, 4, 2, 5, 3, 5])
```

In [33]:

```
a + 10
```

Out[33]:

```
array([14, 14, 11, 13, 15, 14, 12, 15, 13, 15])
```

On ne peux pas faire `liste + 10` (erreur) mais on peut faire `tableau numpy + 10` (ajoute le nombre à chaque élément du tableau).

In [34]:

```
liste = [x for x in range(10)]
liste
```

Out[34]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [36]:

```
# idems avec d'autres opérateurs
a <= 3
```

Out[36]:

```
array([False, False,  True,  True, False, False,  True, False,  True,
       False])
```

Créer un tableau à deux dimensions

In [42]:

```
# type : list
xlist = [[1, 2, 3],[2, 4, 6]]

# transforme en array :
x = numpy.asarray(xlist)
```

In [43]:

```
type(x)
```

Out[43]:

```
numpy.ndarray
```

In [48]:

```
# liste
print(xlist)
```

```
[[1, 2, 3], [2, 4, 6]]
```

In [49]:

```
# tableau numpy
print(x)
```

```
[[1 2 3]
 [2 4 6]]
```

Afficher le dernier élément de la deuxième ligne du tableau

In [50]:

```
x[1, -1]
```

Out[50]:

```
6
```

Afficher la dernière ligne

In [51]:

```
x[-1]
```

Out[51]:

```
array([2, 4, 6])
```