

Ch12-ModulesAndPackages

September 22, 2021

1 Modules and Packages

<http://openbookproject.net/thinkcs/python/english3e/modules.html> - module is a file containing Python definitions and statements intended for use in other Python programs - standard library is an example of Python language provided modules

1.1 Various ways to import names into the current namespace

```
[1]: # import math module into the global namespace  
import math  
x = math.sqrt(100)  
print(x)
```

10.0

```
[2]: import random  
print(random.choice(list(range(1, 21))))
```

20

```
[3]: from random import choice
```

```
[4]: print(choice([1, 2, 3, 4]))
```

2

```
[5]: help(math)
```

Help on module math:

NAME

math

MODULE REFERENCE

<https://docs.python.org/3.8/library/math>

The following documentation is automatically generated from the Python source files. It may be incomplete, incorrect or include features that are considered implementation detail and may vary between Python

implementations. When in doubt, consult the module reference at the location listed above.

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 $\geq x$.

`comb(n, k, /)`

Number of ways to choose k items from n items without repetition and without order.

Evaluates to $n! / (k! * (n - k)!)$ when $k \leq n$ 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 evaluation 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 $\leq x$.

`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 $0.5 \leq \text{abs}(m) < 1.0$.

`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 $k \leq n$ 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

```

/Users/rbasnet/miniconda3/lib/python3.8/lib-
dynload/math.cpython-38-darwin.so

```

```

[6]: from math import * # Import all the identifiers from math
print(sqrt(100))
print(pi)

```

```

10.0
3.141592653589793

```

```

[7]: from math import radians, sin
rad = radians(90)
print(rad)
print(sin(rad))

```

```

1.5707963267948966
1.0

```

1.2 names can be imported in to local namespace

```
[8]: def isUpper(letter):  
      import string # string name is local  
      return letter in string.ascii_uppercase
```

```
[9]: print(isUpper('a'))
```

False

```
[10]: # can we use string module outside isUpper function?  
      print(string.digits)
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-10-1f51304bf154> in <module>  
      1 # can we use string module outside isUpper function?  
----> 2 print(string.digits)  
  
NameError: name 'string' is not defined
```

1.3 scope and lookup rules

The scope of an identifier is the region of program code in which the identifier can be accessed, or used.

Three important scopes in Python: - Local scope refers to identifiers declared within a function - Global scope refers to all the identifiers declared within the current module, or file - Built-in scope refers to all the identifiers built into Python – those like range and min that are (almost) always available

Precedence rule:

innermost or local scope

global scope

built-in scope

```
[11]: def testLocalScope():  
      k = 5  
      for i in range(2):  
          for j in range(2):  
              if i == j:  
                  k = 3  
                  print('inside=',k)  
      print('outside=',k)  
      print(i, j, k)
```



```
[12]: testLocalScope()
```

```
inside= 3
inside= 3
outside= 3
1 1 3
```

```
[13]: # can't access k outside the testLocalScope function
print(k)
```

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-13-3abf04720919> in <module>
      1 # can't access k outside the testLocalScope function
----> 2 print(k)

NameError: name 'k' is not defined
```

1.4 User-defined modules

- see `modules` folder
- see `main.py` and `module2.py` inside `modules` folder
- demonstrates user defined modules and importance of import guard
- run each module, but `main.py` depends on `module2.py`

```
if __name__ == '__main__':
    ...
```

2 Packages

- folder with module(s)
- must define `__init__.py` empty module to initialize as package
- can't import package itself (in a useful way) but only module(s) or identifiers in the modules
- <https://docs.python.org/3/tutorial/modules.html#packages>

2.1 fibos package

- the folder `fibos` in this repository is an example of Python package
- take a look inside the package and observe the files
- see demo script `demos/package_demo.py` that uses `fibos` package
- the following code snippets demonstrate using user-defined package

```
[14]: # change current working directory to demos
%cd demos
```

```
/Users/rbasnet/CMU/projects/Python-Fundamentals/demos
```

```
[15]: import fibos
```

```
[16]: help(fibos)
```

Help on package fibos:

NAME

fibos

PACKAGE CONTENTS

fibonacci

FILE

/Users/rbasnet/CMU/projects/Python-Fundamentals/demos/fibonacci/__init__.py

```
[17]: # can't use the imported package to access its modules!
fibonacci.fibonacci.fib(10)
```

```
-----
AttributeError                                Traceback (most recent call last)
<ipython-input-17-a690191fd105> in <module>
      1 # can't use the imported package to access its modules!
----> 2 fibonacci.fibonacci.fib(10)

AttributeError: module 'fibonacci' has no attribute 'fibonacci'
```

```
[18]: # must import the modules or identifiers defined in the package
import fibonacci.fibonacci as f
f.fib(10)
```

0 1 1 2 3 5 8 13 21 34

```
[19]: from fibonacci import fibonacci
fibonacci.fibonacci(10)
```

```
[19]: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

```
[ ]:
```