Math module in python

In [130…
```
import math module
```

```
  Cell In[130], line 1
    import math module
                  ^
SyntaxError: invalid syntax
```

In [ ]:
```
help()
```

Welcome to Python 3.12's help utility! If this is your first time using Python, you should definitely check out the tutorial at https://docs.python.org/3.12/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing Python programs and using Python modules. To get a list of available modules, keywords, symbols, or topics, enter "modules", "keywords", "symbols", or "topics".

Each module also comes with a one-line summary of what it does; to list the modules whose name or summary contain a given string such as "spam", enter "modules spam".

To quit this help utility and return to the interpreter, enter "q" or "quit".

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.

```
        The result is between 0 and pi.

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

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

        The result is between -pi/2 and pi/2.

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

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

        The result is between -pi/2 and pi/2.

    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.

cbrt(x, /)
    Return the cube root 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 without 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.

exp2(x, /)
    Return 2 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(n, /)
    Find n!.

    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(*integers)
    Greatest Common Divisor.

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.
```

```
lcm(*integers)
    Least Common Multiple.

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 is 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.

nextafter(x, y, /, *, steps=None)
    Return the floating-point value the given number of steps
after x towards y.

    If steps is not specified or is None, it defaults to 1.

    Raises a TypeError, if x or y is not a double, or if steps
is not an integer.
    Raises ValueError if steps is negative.

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.

sumprod(p, q, /)
    Return the sum of products of values from two iterables p
and q.

    Roughly equivalent to:

        sum(itertools.starmap(operator.mul, zip(p, q,
strict=True)))

    For float and mixed int/float inputs, the intermediate
products
    and sums are computed with extended precision.

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.

    ulp(x, /)
        Return the value of the least significant bit of the float
    x.
```

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

FILE (built-in)

help> q

You are now leaving help and returning to the Python interpreter. If you want to ask for help on a particular object directly from the interpreter, you can type "help(object)". Executing "help('string')" has the same effect as typing a particular string at the help> prompt.

In [5]:
```
import math
```

In [7]:
```
x = math.sqrt(25)
x
```

Out[7]: 5.0

In [9]:
```
x1 = math.sqrt(15)
```

In [11]:
```
x1
```

Out[11]: 3.872983346207417

In [13]:
```
print(math.floor(3.87))
```

```
3
```

In [15]:
```
print(math.ceil(3.87))
```

```
4
```

In [17]:
```
print(math.floor(5.77))
```

```
5
```

In [19]:
```
print(math.ceil(5.77))
```

```
6
```

In [21]:
```
print(math.pow(5,4))
```

```
625.0
```

In [23]:
```
print(math.pow(3,2))
```

```
9.0
```

In [25]:
```
print(math.pi)
```

```
3.141592653589793
```

In [27]:
```
print(math.e)
```

```
2.718281828459045
```

In [29]:
```
m.sqrt(25)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[29], line 1
----> 1 m.sqrt(25)

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

In [31]:
```
import math as m
```

In [33]:
```
m.sqrt(20)
```

Out[33]:
```
4.47213595499958
```

In [35]:
```
from math import sqrt, pow
```

In [37]:
```
pow(2,3)
```

Out[37]:
```
8.0
```

In [41]:
```
from math import sqrt,pow,floor,ceil
print(sqrt(25))
print(pow(4,2))
print(floor(5.77))
print(ceil(4.5))
```

```
5.0
16.0
5
5
```

In [43]:
```
from math import *
```

In [49]:
```
print(sqrt(45))
print(pow(3,3))
```

```
6.708203932499369
27.0
```

In [39]:
```
round(pow(2,3))
```

Out[39]:
```
8
```

user input function

In [70]:
```
x = input()
x
```

Out[70]: 'hello'

In [68]:
```
type(x)
```

Out[68]: str

In [72]:
```
x = input()
y = input()
z = x+y
```

In [74]:
```
x = input()
y = input()
z = x+y
print(z)
```

```
35
```

In [76]:
```
x1 = input('Enter the 1st number')
y1 = input('Enter the 2nd number')
z1 = x1 + y1
print(z1)
```

```
4518
```

In [78]:
```
x1 = input('Enter the 1st number')
y1 = input('Enter the 2nd number')
z1 = x+y
print(z1)
```

```
35
```

In [80]:
```
type(x1)
type(y1)
```

Out[80]: str

In [86]:
```
x1 = input('Enter the 1st number')
a1 = int(x1)
y1 = input('Enter the 2nd number')
b1 = int(y1)
z1 = a1 + b1
print(z1)
```

```
20
```

In [82]:
```
x2=int(input('enter the 1st number'))
y2=int(input('enter the 2nd number'))
z2=x2+y2
print(z2)
```

```
9
```

Char format

In [91]:
```
ch = input('enter a char')
print(ch)
```

```
Hello Devil
```

In [97]:
```
print(ch[0])
```

```
H
```

In [99]:
```
print(ch[6])
```

```
D
```

In [101…
```
print(ch[-1])
```

```
l
```

In [103…
```
print(ch[-4])
```

```
e
```

In [105…
```
print(ch[0:6])
```

```
Hello
```

In [111…
```
ch = input('enter a char')[1:3]
print(ch)
```

```
el
```

In [107…
```
ch = input('enter a char')
print(ch)
```

```
2+4-2-1
```

Eval Function using input

In [114…
```
result = eval(input('enter an expr'))
print(result)
```

```
6
```

In [116…
```
result = eval(input('enter an expr'))
print(result)
```

```
13
```

In [ ]:
```

```