functions

September 8, 2021

1 Conditional Statement

#Try ann Except if logitalStatement: Body else: Body

try: Code (run if no error) except: Code (run if there is an error in the try part)

[3]:
```
num = input("Enter positive integer: ")
num = int(num)
print(num ** 0.5)
```

```
Enter positive integer: 12.34


      ␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-3-889151cf76b1> in <module>
         1 num = input("Enter positive integer: ")
   ----> 2 num = int(num)
         3 print(num ** 0.5)


       ValueError: invalid literal for int() with base 10: '12.34'
```

[5]:
```
#above example using try and except
num = input("Enter positive integer: ")
try:
    num = int(num)
    print(num ** 0.5)
except:
    print("Enter proper number")
```

```
Enter positive integer: 23.34
Enter proper number
```

[7]:
```
#another example
num1 = input("Enter number 1 : ")
num2 = input("Enter number 2 : ")

try:
    print(num1 + num2)
except:
    print("Valid proper number")
```

```
Enter number 1 : 12
Enter number 2 : 23
1223
```

[ ]:
```
#factorial of a number
#if num is negative: Please enter positive number
#if num is float: Only integers are allowed
#by Aditi
num = input('Enter the number: ')
try:
    num = int(num)
    if num > 0:
        i = 1
        fac = 1
        while i <= num:
            fac = fac * i
            i += 1
        print('Factorial of the number is:',fac)
    elif num == 0:
        print('Factorial of the number is 1')
    else:
        print('Please enter positive number')
except:
    print('Only integers are allowed')
```

[13]:
```
#by aditi (modified)
num = input('Enter the number: ')
try:
    num = int(num)
    fac = 1
    if num >= 0:
        i = 1
        while i <= num:
            fac = fac * i
            i += 1
        print("factorial is", fac)
    else:
        print('Please enter positive number')
```

```
except:
    print('Only integers are allowed')
```

```
Enter the number: 23.34
Only integers are allowed
```

[16]:
```
#arnav
  #arnav
num=int(input('enter a positive integer:'))

factorial=1
if num<0:
    print('please enter positive number')
if float(num):
    print('only integers are allowed')
while num>=factorial:
    factorial=factorial*i
    num=num+1
    print('factorial of', num,'is:')
if num==0:
    print('factorial of number is 1')
```

```
enter a positive integer:3
only integers are allowed
factorial of 4 is:
factorial of 5 is:
factorial of 6 is:
```

[23]:
```
if 0:
    print("Arnav")
else:
    print("Muskaan")
```

```
Muskaan
```

[21]:
```
bool(0)
```

[21]: False

[25]:
```
#True - > 1
#False -> 0
print(int(True))
print(int(False))
```

```
1
0
```

[26]:
```
int(12.3)
```

[26]:
```
12
```

[27]:
```
round(12.5,0)
```

[27]:
```
12.0
```

[28]:
```
round(12.36, 1)
```

[28]:
```
12.4
```

[29]:
```
round(12.36, 0)
```

[29]:
```
12.0
```

[30]:
```
round(12.36, 2)
```

[30]:
```
12.36
```

[31]:
```
round(2**0.5, 3)
```

[31]:
```
1.414
```

[32]:
```
2 ** 0.5
```

[32]:
```
1.4142135623730951
```

2 Math Library

[33]:
```
import math
```

[34]:
```
dir(math)
```

[34]:
```
['__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'acos',
 'acosh',
 'asin',
 'asinh',
 'atan',
 'atan2',
 'atanh',
```

```
 'ceil',
 'copysign',
 'cos',
 'cosh',
 'degrees',
 'e',
 'erf',
 'erfc',
 'exp',
 'expm1',
 'fabs',
 'factorial',
 'floor',
 'fmod',
 'frexp',
 'fsum',
 'gamma',
 'gcd',
 'hypot',
 'inf',
 'isclose',
 'isfinite',
 'isinf',
 'isnan',
 'ldexp',
 'lgamma',
 'log',
 'log10',
 'log1p',
 'log2',
 'modf',
 'nan',
 'pi',
 'pow',
 'radians',
 'remainder',
 'sin',
 'sinh',
 'sqrt',
 'tan',
 'tanh',
 'tau',
 'trunc']
```

[35]:
```
help(math)
```

```
Help on module math:
```

```
NAME
    math

MODULE REFERENCE
    https://docs.python.org/3.7/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 >= x.

    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.

    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 <= 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(x, y, /)
        Return the Euclidean distance, sqrt(x*x + y*y).

    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.
```

```
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.

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

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
    /opt/anaconda3/lib/python3.7/lib-dynload/math.cpython-37m-darwin.so
```

[ ]: