Function

We use a number of common mathematical functions from the Python standard library's math module. Some of the common functions is given below:

function	use
math.acos()	Returns the arc cosine of a number
---	---
math.acosh()	Returns the inverse hyperbolic cosine of a number
---	---
math.asin()	Returns the arc sine of a number
---	---
math.asinh()	Returns the inverse hyperbolic sine of a number
---	---
math.atan()	Returns the arc tangent of a number in radians
---	---
math.atan2()	Returns the arc tangent of y/x in radians
---	---
math.atanh()	Returns the inverse hyperbolic tangent of a number
---	---
math.ceil()	Rounds a number up to the nearest integer
---	---
math.comb()	Returns the number of ways to choose k items from n items without repetition and order
---	---
math.cos()	Returns the cosine of a number
---	---
math.cosh()	Returns the hyperbolic cosine of a number
---	---
math.degrees()	Converts an angle from radians to degrees
---	---
math.dist()	Returns the Euclidean distance between two points (p and q), where p and q are the coordinates of that point
---	---
math.exp()	Returns e raised to the power of x
---	---
math.expm1()	Returns e to the power x - 1
---	---

function	use
math.fabs()	Returns the absolute value of a number
---	---
math.factorial()	Returns the factorial of a number
---	---
math.floor()	Rounds a number down to the nearest integer
---	---
math.fmod()	Returns the remainder of x/y
---	---
math.fsum()	Returns the sum of all items in any iterable (tuples, arrays, lists, etc.)
---	---
math.gamma()	Returns the gamma function at x
---	---
math.gcd()	Returns the greatest common divisor of two integers
---	---
math.hypot()	Returns the Euclidean norm
---	---
math.isqrt()	Rounds a square root number downwards to the nearest integer
---	---
math.log()	Returns the natural logarithm of a number, or the logarithm of number to base
---	---
math.log10()	Returns the base-10 logarithm of x
---	---
math.log1p()	Returns the natural logarithm of 1+x
---	---
math.log2()	Returns the base-2 logarithm of x
---	---
math.perm()	Returns the number of ways to choose k items from n items with order and without repetition
---	---
math.pow()	Returns the value of x to the power of y
---	---
math.prod()	Returns the product of all the elements in an iterable
---	---
math.radians()	Converts a degree value into radians
---	---
math.remainder()	Returns the closest value that can make numerator completely divisible by the denominator
---	---

function	use
math.sin()	Returns the sine of a number
---	---
math.sinh()	Returns the hyperbolic sine of a number
---	---
math.sqrt()	Returns the square root of a number
---	---
math.tan()	Returns the tangent of a number
---	---
math.tanh()	Returns the hyperbolic tangent of a number
---	---
math.trunc()	Returns the truncated integer parts of a number

In [1]:
```
import math
math.sin(math.pi/2)
```

Out[1]:
```
1.0
```

#to install sympy pip install sympy

math.sin() works for numerical values. It gives error ' can't convert expression to float' when it uses symbol. Thus we use the function sympy.sin()

In [2]:
```
import sympy
from sympy import Symbol
x = Symbol('x')
y = math.sin(x)
print(y)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-2-81f2b792921f> in <module>
      2 from sympy import Symbol
      3 x = Symbol('x')
----> 4 y = math.sin(x)
      5 print(y)

~\anaconda3\lib\site-packages\sympy\core\expr.py in __float__(self)
    356         if result.is_number and result.as_real_imag()[1]:
    357             raise TypeError("can't convert complex to float")
--> 358         raise TypeError("can't convert expression to float")
    359
    360     def __complex__(self):

TypeError: can't convert expression to float
```

In [3]:
```
import sympy
from sympy import *
sympy.sin(math.pi/2)
x = Symbol('x')
y = sympy.sin(x)
print(y)
```

```
sin(x)
```

In [4]:
```
#Similar to the standard library's sin() function, SymPy's sin() function expects
import sympy
sympy.sin(math.pi/2)
```

Out[4]: 1.0

In [ ]:
```

```

1.Program check whether the expression x + 5 is greater than 0.

In [5]:
```
# define x with sign
x = Symbol('x', positive=True)# positive = True should be specified otherwise Sym
#if we create a Symbol object specifying positive=True, we tell SymPy to assume c
# check the condition
if (x+5) > 0:
 print('Do Something')
else:
 print('Do Something else')
```

```
Do Something
```

We can use the solve() function to express one variable in an equation in terms of the others

In [6]:
```
x = Symbol('x')
y = Symbol('y')
solve(2*x+3*y-5, y)
```

Out[6]:
```
[5/3 - 2*x/3]
```

2. Derive the quadratic formula for solving a quadratic polynomial ax^2+bx+c

In [7]:
```
x = Symbol('x')
a = Symbol('a')
b = Symbol('b')
c = Symbol('c')
exp = a*x*x + b*x + c
solve(exp, x)
```

Out[7]:
```
[(-b + sqrt(-4*a*c + b**2))/(2*a), -(b + sqrt(-4*a*c + b**2))/(2*a)]
```

3.Derive the expression for the time it takes for a body in projectile motion to reach the highest point if it's thrown with initial velocity u at an angle theta. (At highest point the vertical component of a projectile motion is 0. That is, $u\sin(\theta) - gt = 0.$)

In [8]:
```
from sympy import sin, solve, Symbol
u = Symbol('u')
t = Symbol('t')
g = Symbol('g')
theta = Symbol('theta')
solve(u*sin(theta)-g*t, t)
```

Out[8]:
```
[u*sin(theta)/g]
```

4.Derive the expression for determining hypotenuse of a right angled triangle (Pythagoras Theorem). Using that find the hypotenuse of a right angled triangle with given base and altitude.

In [9]:
```
from math import *
base = Symbol('base')
alt = Symbol('alt')
hyp = Symbol('hyp')
t = Symbol('t')
t = hyp**2 - alt**2 - base**2
t1 = solve(t, hyp)
u=t1[1]
print("hypotenuse = ", u)
a = float(input("Give altitude: "))
b = float(input("Give base: "))
res = u.subs({hyp:hyp, alt:a, base:b})
print("when altitude =", a, "and base = ", b, "hypotenuse =", res )
```

```
hypotenuse =  sqrt(alt**2 + base**2)
Give altitude: 3
Give base: 4
when altitude = 3.0 and base =  4.0 hypotenuse = 5.00000000000000
```

5.Using the formula for equations of motion $s = ut + \frac{1}{2}at^2$ derive the equation for

1.Time, t

2.Acceleration, a

3.Initial velocity, u

In [10]:
```
from sympy import Symbol, solve, pprint
s = Symbol('s')
u = Symbol('u')
t = Symbol('t')
a = Symbol('a')
expr = u*t + (1/2)*a*t**2 - s
print("1. Time is given by: t =", solve(expr,t))
print("2. Accelartion is given by: a =",solve(expr,a))
print("3. Intial velocity is given by: u =",solve(expr,u))
```

```
1. Time is given by: t = [(-u + 1.4142135623731*sqrt(a*s + 0.5*u**2))/a, -(u +
1.4142135623731*sqrt(a*s + 0.5*u**2))/a]
2. Accelartion is given by: a = [2.0*(s - t*u)/t**2]
3. Intial velocity is given by: u = [-0.5*a*t + s/t]
```

In [ ]:
```

```