avery_tutorial_1

You can start typing in an input cell and use the "evaluate" key or just hit <SHIFT>-enter to evaluate it.

You can insert a new input cell in front of another cell by hovering over the top of the next input cell and clicking on the blue line that appears.

You can insert a new text cell in front of another cell and <SHIFT>-click on the blue line.

```
#It's usually easier to use comments directly in the input cell
using a # before the comment.

4 + 5   #Here's an inline comment that's ignored by Sage during
evaluation
```

9

```
#If you evaluate multiple Sage lines, only the output of the last
line is printed

4*2
sin(2.)
sqrt(2)
sqrt(2.)
```

1.41421356237310

```
#If you want multiple commands on a line, separate them with a
semicolon ";".
#All outputs are shown, each on a separate line

4*2; sin(2.); sqrt(2); sqrt(2.)
```

8
0.909297426825682
$\sqrt{2}$
1.41421356237310

```
#If you separate items by a comma, that generates a standard
Python tuple of values
#(A tuple is a list that can't be changed)

4*2, sin(2.), sqrt(2), sqrt(2.)
```

$\left(8, 0.909297426825682, \sqrt{2}, 1.41421356237310\right)$

```
#You can always show intermediate output by using the print
statement.
#However, the output is not typeset as beautifully as when the
Typeset box above is checked.
#TeX must be installed to make typesetting work.
```

```
print sqrt(2)/2
sqrt(2)/2
```

1/2*sqrt(2)

$\frac{1}{2}\sqrt{2}$

```
#Sage treats expressions as exact, unless a decimal point is used, in which case the
#accuracy is that of standard computer floating point representation, about 15-16 digits.
#Look at the difference between these two expressions. The first is exact and the second is approximate.

1/2 + 1/3 + 1/4;        1/2. + 1/3 + 1/4
```

$\frac{13}{12}$

1.08333333333333

```
#Or look at the following expressions.
#   Note that 2 - sqrt(2)^2 is 0 exactly
#   However, 2 - sqrt(2.)^2 is not exactly zero

sqrt(2); sqrt(2.); 2-sqrt(2)^2; 2-sqrt(2.)^2; sin(1); sin(1.)
```

$\sqrt{2}$

1.41421356237310

0

$-4.44089209850063 \times 10^{-16}$

$\sin(1)$

0.841470984807897

```
#You can get the numerical value of any expression using n(), which provides standard
#floating point precision. In the following, n() is used as suffix, where it is
#called an "object method"

pi.n(); e.n(); (sqrt(2)).n()
```

3.14159265358979

2.71828182845905

1.41421356237310

```
#n() can also be used as an ordinary function

n(pi); n(e); n(sqrt(2))
```

$(3.14159265358979, 2.71828182845905, 1.41421356237310)$

```
#You can also get an arbitrary number of digits from n() if they are specified

pi.n(digits=50); e.n(digits=30); n(sqrt(2), digits=60)
```

3.1415926535897932384626433832795028841971693993751

$2.71828182845904523536028747135$
$1.4142135623730950488016887242096980785696718753769480731766$8

```
#Sage has standard math constants, including I = sqrt(-1).
#Google "Euler's constant" to find how it is defined.

pi; e; I; 5 + 3*I; euler_gamma; euler_gamma.n(); (5+3*I) * (5-
3*I)
```

π
e
i
$3i + 5$
γ_E
0.577215664901533
34

```
#+infinity is represented by oo

oo; -oo; tan(pi/2)
```

$+\infty$
$-\infty$
∞

```
#Exponentiation is performed with either the ** or ^ operator

5**4; 5^4;
```

625
625

```
#Sage has all the standard math and trig functions

exp(3); sqrt(5); sin(2); cos(2); tan(2); arcsin(1/2);
arccos(1/2); arctan(oo)
```

e^3
$\sqrt{5}$
$\sin(2)$
$\cos(2)$
$\tan(2)$
$\frac{1}{6}\pi$
$\frac{1}{3}\pi$
$\frac{1}{2}\pi$

```
#Even the hyperbolic trig functions are supported

sinh(2); cosh(2); tanh(2); arcsinh(1/2); arccosh(1/2);
arctanh(1)
```

$\sinh(2)$
$\cosh(2)$
$\tanh(2)$
$\operatorname{arcsinh}\left(\frac{1}{2}\right)$

$\operatorname{arccosh}\left(\frac{1}{2}\right)$

$+\infty$

```
#Combinatoric related quantities such as factorial and binomial are supported
#binomial(N,n) = N! / [n! * (N-n)! ]

factorial(6); binomial(6, 3)
```

720

20

```
#The factor() function factors integers or algebraic expressions.
#It can be used as a global function or object method

1035.factor(); factor(1048764);
```

$3^2 \cdot 5 \cdot 23$

$2^2 \cdot 3 \cdot 17 \cdot 53 \cdot 97$

```
(x^2 - 5*x + 6).factor(); factor(x^4 + x^3 - 38*x^2 - 8*x + 240)
```

$(x-2)(x-3)$

$\left(x^2-8\right)(x+6)(x-5)$

```
#factor() can also be used to simplify algebraic fractions

f = 3 + 1/(x+1) + 1/(x-1); f; f.factor()
```

$\frac{1}{x+1} + \frac{1}{x-1} + 3$

$\frac{3x^2+2x-3}{(x+1)(x-1)}$

```
#expand() is the opposite of factor

expand( (x^2-8)*(x+6)*(x-5) );    expand( (x+1)^6 )
```

$x^4 + x^3 - 38x^2 - 8x + 240$

$x^6 + 6x^5 + 15x^4 + 20x^3 + 15x^2 + 6x + 1$

```
#x is the only predefined symbolic variable that can be used in expressions.
#New symbolic variables are defined using the var() function

var("y z N n")   #Defines the variables y, z, N and n
factorial(N); binomial(N,n)
```

$N!$

$\binom{N}{n}$

```
#Use the subs() method to evaluate an expression with particular values for the variables

factorial(N).subs(N=10); exp(-x*y).subs(x=0.60, y=2.5)
```

3628800

0.223130160148430

```
#You can also use trig_simplify() to simplify trig expressions

f = sin(x)^2 + cos(x)^2
f; f.trig_simplify(); (tan(x)^4).trig_simplify()
```

$\cos\left(x\right)^2 + \sin\left(x\right)^2$

1

$\frac{\sin\left(x\right)^4}{\cos\left(x\right)^4}$

```
#trig_reduce replaces sin(x)^n and cos(x)^n with terms involving
sin(n*x) and cos(n*x)
#It can only be used as an object method, for some reason

(sin(x)^4).trig_reduce(); (cos(x)^7).trig_reduce()
```

$\frac{1}{8}\cos(4x) - \frac{1}{2}\cos(2x) + \frac{3}{8}$

$\frac{1}{64}\cos(7x) + \frac{7}{64}\cos(5x) + \frac{21}{64}\cos(3x) + \frac{35}{64}\cos(x)$