Workshop 2: Making things simpler with SymPy

In [1]:

```
from sympy import *
%matplotlib inline
init_printing()
```

In [2]:

```
x,y,z = symbols('x y z')
a,b,c = symbols('a b c')
```

1. Shortcuts

Before discussing some more nice things that can be done with SymPy, it is worth mentioning that if you click on the *Help* drop down button in the IPython notebook window, there is an option to view the keyboard shortcuts for IPython notebooks. These will be particularly useful for writing your own IPython notebooks.

2. Manipulating expressions

2.1. Simplify

If you have an expression and you would like it in a simpler form, then you can use the `simplify` command. For example:

In [3]:

```
sin(x)**2+cos(x)**2
```

Out[3]:

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

In [4]:

```
simplify(sin(x)**2+cos(x)**2)
```

Out[4]:

$$1$$

This comes in very handy when SymPy produces odd-looking output.

Generally speaking, `simplify` is a good first option for attempting to tidy up "messy" expressions. It can be used on a variety of different expressions:

In [5]:

```
(x**3 + x**2 - x - 1)/(x**2 + 2*x + 1)
```

Out[5]:

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

In [6]:

```
simplify((x**3 + x**2 - x - 1)/(x**2 + 2*x + 1))
```

Out[6]:

$$x - 1$$

In [7]:

```
gamma(x)/gamma(x - 2)
```

Out[7]:

$$\frac{\Gamma(x)}{\Gamma(x - 2)}$$

In [8]:

```
simplify(gamma(x)/gamma(x - 2))
```

Out[8]:

$$(x - 2)\,(x - 1)$$

Make a function $h(x) = x^a - x^{-a}$, and differentiate it. SymPy should produce something which is correct but not obviously the best way to present it.

In [20]:

```
h = x**a - x**(-a)
h
```

Out[20]:

$$x^a - x^{-a}$$

In [17]:

```
q = diff(h,x)
q
```

Out[17]:

$$\frac{ax^{a}}{x} + \frac{ax^{-a}}{x}$$

Now put `simplify(.)` around the differentiation command and see the result.

In [18]:

```
simplify(q)
```

Out[18]:

$$\frac{ax^{-a}\left(x^{2a}+1\right)}{x}$$

However, `simplify` is certainly not the best command to use in every case, and it is quite likely that you will come across expressions that are **not** simplified in the way you were hoping for by using merely `simplify`, but **can** be simplified using other commands.

For example, try and use `simplify` in the space below to factorise $x^3 + 7x^2 + 16x + 12$ into $(x+2)^2(x+3)$:

In [21]:

```
simplify(x**3 + 7*x**2 + 16*x + 12)
```

Out[21]:

$$x^3 + 7x^2 + 16x + 12$$

For cases like this, we have to use commands which are more specific to the type of expression we are dealing with (for example a polynomial expression).

Next week we'll look at some commands that are used for simplifying trigonometric expressions, but for now let's go through some useful commands for dealing with polynomial expressions.

2.2. Factorising polynomials

To factorise polynomials, we use `factor(.)`. So for the example we attempted above we write:

In [22]:

```
factor(x**3 + 7*x**2 + 16*x + 12)
```

Out[22]:

$$(x + 2)^2 (x + 3)$$

Use `factor` to factorise $18x^2z + 12xyz + 2y^2z$:

In [23]:

```
factor(18*x**2*z + 12*x*y*z + 2*y**2*z)
```

Out[23]:

$$2z(3x + y)^2$$

2.3. Expanding polynomials

We use the `expand` command to force SymPy to expand things out:

In [24]:

```
(x - 3)**2*(x + 6)
```

Out[24]:

$$(x - 3)^2 (x + 6)$$

In [25]:

```
expand((x - 3)**2*(x + 6))
```

Out[25]:

$$x^3 - 27x + 54$$

This may not seem like a natural simplification tool, but sometimes we come across expressions which are a bit too condensed, and if we actually `expand` them, we can make them simpler due to cancellation.

In the space below define $p = (x + 1)(x - 3) - x(x - 2)$, and run `expand(p)`.

In [29]:

```
p = (x+1)*(x-3) - x*(x-2)
p
expand(p)
```

Out[29]:

$$-3$$

Note: Although we are just considering polynomials for the moment, the command `expand` does **not** have to be used *exclusively* with polynomial expressions.

2.4. Collecting powers of a polynomial

If we end up with an expression with lots of parameters - for example, a polynomial with more than one variable - we may find it useful to `collect` terms together appropriately.

`collect` takes two arguments (or inputs). The first argument is the function you wish to collect terms from, and the second argument is the term you wish to collect together. So, running `collect(f,x)` collects the x terms together for the function f.

Let us look at an example:

In [30]:

```
f = 3*x**3 + x**2*y*z**2 - 9*x**2*z + x + 2*x*z
f
```

Out[30]:

$$3x^3 + x^2yz^2 - 9x^2z + 2xz + x$$

In [31]:

```
collect(f,x)
```

Out[31]:

$$3x^3 + x^2\left(yz^2 - 9z\right) + x\left(2z + 1\right)$$

Define $g = xy + x + 7 + 6x^2 - zx^2 + x^3$, and run `collect(g,x)`.

In [32]:

```
g = x*y + x + 7 + 6*x**2 - z*x**2 + x**3
g
```

Out[32]:

$$x^3 - x^2z + 6x^2 + xy + x + 7$$

In [33]:

```
collect(g,x)
```

Out[33]:

$$x^3 + x^2\left(6 - z\right) + x\left(y + 1\right) + 7$$

3. Referring to previous output

You have just done a complicated integral and you are sure it can be made simpler: what to do? You could try and copy the code into a `simplify` command and hope you don't make any mistakes. But there is a quicker way. IPython has three references which are very much worth remembering: `_` refers to the last result Python calculated, while `__` refers to the last-but-one, and `___` refers to the last-but-two.

Note: It is the last result *run*, which is not necessarily the result immediately above the command using `_` !

Note: Furthermore, using `_` to refer to a command in the same code block will *not* work, since it is run at the same time, and is thus not the *previous* output.

Find $\int x^2(e^x + e^{-x})\,dx$ and then run `expand( _ )`.

In [41]:

```
f = x**2*exp(x) + x**2*exp(-x)
f
```

Out[41]:

$$x^2 e^x + x^2 e^{-x}$$

In [43]:

```
h = integrate(f)
expand(h)
```

Out[43]:

$$x^2 e^x - x^2 e^{-x} - 2xe^x - 2xe^{-x} + 2e^x - 2e^{-x}$$

4. Solve

4.1. Equals sign

An important point to make about the equals sign (=) in Python, is that it does not actually represent an equality. Rather, it represents the assignment of a variable.

The way to represent an equality symbolically in SymPy is to use the $\mathrm{Eq}(.)$ object:

In [44]:

```
Eq(x + 5, 8)
```

Out[44]:

$$x + 5 = 8$$

Note: Those of you who are more familiar with programming syntax, may think that ==, which is used for equality testing, could be used for equality in SymPy. However, this will always return a boolean type since we are asking a question.

4.2. Using solve

If you have a polynomial equation for which you could find a root, you can use the `solve` command. Specify what you want to find a root for: it might be a function or an equation. If you have more than one letter or variable in there, you should put a comma and then the variable you want to solve for.

For example you can get the well-known quadratic equation by putting:

In [45]:

```
solve(Eq(a*x**2 + b*x + c, 0), x)
```

Out[45]:

$$\left[\frac{-b + \sqrt{-4ac + b^2}}{2a}, \quad -\frac{b + \sqrt{-4ac + b^2}}{2a}\right]$$

4.3. Taking equations a step further

So we can represent equations symbolically in SymPy using $\mathrm{Eq}(.)$. However it is also worth noting that in SymPy, when it comes to using the `solve` functions, if an expression is not in an $\mathrm{Eq}(.)$, then SymPy assumes it to be equal to zero. Thus, the following are equivalent:

In [46]:

```
solve(Eq(x**2, 4), x)
```

Out[46]:

$$[-2, \ 2]$$

In [47]:

```
solve(Eq(x**2 - 4, 0), x)
```

Out[47]:

$$[-2, \ 2]$$

In [48]:

```
solve(x**2 - 4, x)
```

Out[48]:

$$[-2, \ 2]$$

Try the following in a `solve` command:

- $x^2 + 3x + 2 = 0$
- $5x^2 + 3x = 10$
- $\sin(ax) = \cos(x)$ w.r.t. x

In [49]:

```
solve(Eq(x**2 + 3*x + 2, 0),x)
```

Out[49]:

$$[-2, \ -1]$$

In [50]:

```
solve(Eq(5*x**2 + 3*x - 10, 0), x)
```

Out[50]:

$$\left[-\frac{3}{10} + \frac{\sqrt{209}}{10}, \ -\frac{\sqrt{209}}{10} - \frac{3}{10}\right]$$

In [51]:

```
solve(Eq(sin(a*x) - cos(x),0),x)
```

Out[51]:

$$\left[\pi, \ \frac{\pi}{a}, \ \frac{\pi}{2\,(a-1)}, \ \frac{\pi}{2\,(a+1)}\right]$$

5. Further information

Remember, you can find more information about all these commands by clicking on the *Help* drop down and selecting *SymPy*.

In [ ]:

```

```