Week 6 - Algebra

Now that we have a good knowledge of programming we can take a closer look at a Mathematics package. There are a variety of such packages: Maple, Mathematica and others. The mathematics package we will use is called Sage, as stated on the Sage website (sagemath.org:

> Sage is a free open-source mathematics software system licensed under the GPL. It combines the power of many existing open-source packages into a common Python-based interface.

Sage being an open-source program means that it is being developed by other mathematicians all over the world. It is also completely free (so you can download a copy on your own computer).

Sage allows us to solves equations, differentiate expressions, plot graphs and do various other mathematical operations.

Sage is based on Python so you can use all the general programming techniques you have learnt up until now in Sage. Sage has all of the commands and functions you are used to in Python although some might do a bit more. Sage also has a lot of built in objects for specific mathematical operations.

1. **TICKABLE** Open Sage

   Waiting for our sage server to be set up

2. **TICKABLE** It is very easy to get help in Sage. Simply type any command followed by a ? to get a help file for it:

   ```
   cos?
   sum?
   ```

   Video hint

3. We can use Sage as an advanced calculator. Try out the following commands:

   ```
   cos(3)
   cos(3.)
   sqrt(4)
   sqrt(8)
   sqrt(8.)
   ```

   Sage uses I to denote the imaginary constant:

   ```
   I ** 2
   sqrt(-53.)
   ```

4. **TICKABLE** Obtain an exact form for $\sum_{i=1}^{5} \frac{1}{\sqrt{i}}$. Once you have an expression, experiment with the factor, simplify and expand functions.

   Video hint

5. Sage can also be used to obtain prime factors of numbers:

   ```
   factor(234398)
   ```

   We can do this as above using a function but also using the factor method on the object: 234398:

   ```
   234398.factor()
   ```

   To obtain the number of prime factors that a number has we can use the len function or method:

   ```
   f = 234398.factor()
   len(f)
   ```

   Obtain the mean number of factors for the first 1000 integers (you can use the mean sage function).

   Video hint

6. Using the is_prime method investigate the following claim:

All numbers of the form $k^2 - 79 * k + 1601$ are prime for $k \in \mathbb{Z}$ and $k \geq 3$.

7. **TICKABLE** A very important aspect of Sage is it's ability to handle symbolic computation. In other words it is capable of simplifying:

$$x^2 - 3x^2 + 4xy - 81y^2$$

To do so we need to first declare symbolic variables:

```
y = var('y')
```

(We could also declare x but x is a default variable in Sage that is always declared). Once we have done that we can assign the above expression to a variable:

```
myexp = x ** 2 - 5 * x ** 2 + 12 * x * y - 9 * y ** 2
```

Note that here I'm using the exponentiation used in python: ** but in Sage we can also use ^:

```
myexp = x ^ 2 - 5 * x ^ 2 + 12 * x * y - 9 * y ^ 2
```

Once we have done that we can factorise the expression using the factor method:

```
myexp.factor()
```

Video hint

8. We can also define functions in Sage. The following code defines $f : x \to x^3 + \pi x^2 - \frac{23}{2}x^2 + 15x - \frac{23}{2}\pi x + 15\pi$:

```
f(x) =  x^3 + pi*x^2 - 23/2*x^2 - 23/2*pi*x  + 15*x+ 15*pi
```

To take a look at f we can use the plot function:

```
plot(f(x), x, -15, 15)
```

Experiment with the arguments in that expression.
To idenfity the 3 (visible on the plot) roots of our function we can use the roots method on f:

```
f.roots()
```

We can also try to factorise f:

```
f.factor()
```

Finally we can use the solve function:

```
solve(f(x) == 0, x)
```

Video hint

9. **TICKABLE** Using Sage obtain the solution to the following equations:

    1. $x^2 = -1$
    2. $x^2 - 53x + 2a = 0$
    3. $sin(x) = x - 1$ (Investigate the Sage function find_root)
    4. $x^5 + sin(x) - 2 * x = .5$

Video hint

10. It is also possible to solve systems of equations using Sage. In this case we pass a list of equations as arguments to the solve function. The following code gives a solution to this system of equation:

$$\begin{cases} x + y = z \\ 3x - y = 0 \\ y + z = 1 \end{cases}$$

```
y, z = var('y', 'z')
solve([x + y == z, 3*x - y == 0, y + z ==1], [x, y, z])
```