Introduction to Programming
/ Week 3: Programming in Python, variables, conditionals, strings
/ Mathematical methods: complex numbers

Complex numbers

Python

Remember to start with

```
from sympy import *
```

If you want to solve an equation that is not equal to zero, the simplest thing to do might be to rearrange it for zero. For example, if you want to solve $x^2 = -4$, ask `solve` to solve $x^2 + 4 = 0$.

```
solve(x**2+4)
```

Notice how sympy uses `I` for $\mathrm{i} = \sqrt{-1}$? To use this you need to multiply `I` by a number and don't forget the multiplication `*`, for example to enter $z = 3i$ you would enter

```
z = 7 + 3*I
```

If we define a second complex number

```
w = 3-5*I
```

then we can manipulate these in the expected way.

```
z+w
z-w
z*w
```

We can ask for the real part of a complex number using

```
re(z*w)
```

and imaginary part using

```
im(z*w)
```

We can find its modulus using the `abs` (absolute value) function.

```
abs(z)
```

and its argument using

```
arg(z)
```

We can find the conjugage of z using

```
conjugate(z+w)
```

LaTeX

First, let's set up a basic document to do some maths in.

```

```

```
\documentclass[12pt]{article}
\usepackage{amsmath}

\begin{document}

\end{document}
```

We can add commands to typeset maths after `\begin{document}` and before `\end{document}`.

For the imaginary number, we can just type `i`. We can write out how this is defined using `\sqrt` for the square root.

```
\[ \sqrt{-1} = i \]
```

In solving equations like $x^2 = -4$ above, we would write the solution as $\pm 2i$. This is done in LaTeX using

```
\[ \pm 2i \]
```

There are some formulae in complex numbers that involve sine and cosine. These are represented in LaTeX using `\sin()` and `\cos()`, so for example

```
\[ e^{i \theta} = \cos(\theta) + i \sin(\theta) \]
```

The modulus of a complex number $z = a + ib$ is represented using

```
\[ |z| = \sqrt{a^+b^2} \]
```

The argument $\arg(z) = \tan^{-1}\left(\frac{b}{a}\right)$ is much more challenging to typeset because it involves several new commands.

First, the fraction $\frac{b}{a}$. This is produced by the command `\frac` which takes two parts - the numerator and the denominator. In this case

```
\[\frac{b}{a}\]
```

The inverse tan can be written $\arctan$ or $\tan^{-1}$. For the first, we use `\arctan`. For the second, we use an index.

In LaTeX an index is produced using `^`, like `x^2` for x^2. The `^` acts on the next thing that comes after it, so if that has multiple characters it must be contained with `{...}`. For example, if we write `x^23` we will get x^23 - the `^` has only acted on the `2`. If we want x^{23}, we must make the `^` act on the whole `23` using braces: `\(x^{23}\)`.

For $\tan^{-1}$, then, we use

```
\[\tan^{-1}\]
```

Now the fraction $\frac{b}{a}$ is in brackets. However, these are not ordinary brackets, they are taller. Notice how if we use `(\frac{b}{a})` the result doesn't look right. Instead of writing `(...)`, we use the commands `\left( ... \right)`. LaTeX matches up the `\left(` with the `\right)` and makes them the right height for what comes in between.

```
\[\left( \frac{b}{a} \right)\]
```

Putting that all together, we have

```
\[ \arg(z) = \tan^{-1}\left( \frac{b}{a} \right)\]
```

Another piece of maths you saw in Methods this week is the various labels for sets of numbers - $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$.

To use these, we must use the extra package `\usepackage{amssymb}` in the header of the file.

These are produced by the command `\mathbb{}`, for example $\mathbb{C}$ comes from

```
\[ \mathbb{C} \]
```

Unnecessary extra detail

Typically in maths, we typeset variables as italic letters, like x or n. Because i and e aren't variables, people often prefer to typeset these as straight (Roman) characters. This is done using `\mathrm{i}`, so we might prefer to write

```
\[ \mathrm{e}^{\mathrm{i} \pi} + 1 = 0 \]
```