Workshop 6: Python as a programming language

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')
```

In addition to the mathematical tools at your disposal, you can also use Python to program code. This has various uses; we shall think about some of the simplest, almost 'toy' uses today, which can lead on to more complicated tasks. Many mathematics graduates go on to careers which involve some awareness of programming, so this is an opportunity to get started with a useful skill for the future!

1. For loops

The basic `for` loop takes a list, then starting at some value, which is the first in the list, it carries out some procedure for that value, before moving on to the next value in the list, and carrying out the procedure for this next value. The process continues like this until it has carried out the procedure for the final item in the list, and then stops.

This may sound a little complicated, but don't worry, the example below should help clarify what a `for` loop does:

In [3]:

```
for each_item in ['cat','dog','rat']:
    print(each_item)
```

```
cat
dog
rat
```

Note: I have written `each_item`, but I could have written any name. General practice for simple loops is to use something like `i` or `n`.

The `print` command 'prints' something in the output area.

`for` loops can iterate through many different kinds of lists, but we will be focusing on lists of numbers.

Here is an example of a piece of code which tests whether the first 10 numbers are prime, using the aptly-named `isprime` function:

In [4]:

```
for i in [0,1,2,3,4,5,6,7,8,9]:
    print(isprime(i))
```

```
False
False
True
True
False
True
False
True
False
False
```

1.1. Using range in for loops

Above, we have seen how to use lists of numbers in `for` loops. It seems a bit tedious to sit and type out these lists though, especially if we are wanting to carry out a loop on a much larger list!

Fortunately, there is an easier way. Python has a command called `range(n)` which returns n integers, starting from 0 up to $n-1$. (Technically, `range(n)` is not a list, but you can use it in loops in the same way.) For example:

In [5]:

```
range(10)
```

Out[5]:

```
range(0, 10)
```

This can be substituted straight into a `for` loop:

In [6]:

```
for i in range(10):
    print(isprime(i))
```

```
False
False
True
True
False
True
False
True
False
False
```

We can also alter the starting point and the step size of `range` by passing it more than one argument:

In [7]:

```
range(10,21)
```

Out[7]:

```
range(10, 21)
```

In [8]:

```
range(5,16,2)
```

Out[8]:

```
range(5, 16, 2)
```

When given more than one argument, `range` takes the first number as the starting point, the second as the end point, and the third (if given) as the step size.

Use this new knowledge to make some code which starts at 3 and, increasing by two each time up to 21, tests and reports whether each number is prime.

Warning: Take note of the colon at the end of the `for` line. The code will give you an error if you miss it out!

In [9]:

```
for i in range(3,21,2):
    print(isprime(i))
```

```
True
True
True
False
True
True
False
True
True
```

2. The if statement

We use `if` statements to control whether certain actions take place. For example, let us define a function f, and then write a `for` loop which searches for values of n where $f(n)f(n+1)<0$. (If you think about it, this means that f must change sign somewhere in the open interval $(n,n+1)$.)

In [10]:

```
f = 4*x**2 + 4*x - 3

for n in range(-10,10):
    f1 = f.subs(x,n)
    f2 = f.subs(x,n+1)
    if f1*f2 < 0:
        print("A root lies between x =", n, "and x =", n+1)
```

```
A root lies between x = -2 and x = -1
A root lies between x = 0 and x = 1
```

Here, we have an `if` statement, followed by a condition. This condition must be something which is definitely, calculably, either true or false. For example, inequalities are permitted, as are equalities. Also, you saw that the `isprimes` function returned values of true or false, so that too can be used.

3. Nested for loops

Rather like Russian dolls, `for` loops can be fitted one inside another. This is useful if you want to run one variable up to a limit involving another. As a toy example, here is a piece of code which prints first one x, then two, up to five.

In [11]:

```
for n in range(1,6):
    x = ''
    for m in range(1,n+1):
        x = x + 'x'
    print(x)
```

```
x
xx
xxx
xxxx
xxxxx
```

Note: Python interprets '' as an empty *string*. Python can add (concatenate) strings together e.g. 'a' + 'b' = 'ab' and 'c' + '' = 'c'.

How might you get Python to print not x but consecutive numbers? That is:

1
12
123
1234
12345

Hint: You will need to make use of `str(.)`, which changes the type of data it is given into a string.

In [87]:

```
for n in range(1,6):
    x = ''
    for m in range(1,n+1):
        x = x + str(m)
    print(x)
```

```
1
12
123
1234
12345
```

In [88]:

```
x,y,z = symbols('x y z')
```

4. Exercises

1. Can you write some code which counts through the numbers 1 to 100 and prints each to the output area if, and only if, it is prime?
2. FizzBuzz is a popular game in British primary schools. The idea is that children take it in turns to say a number. Each number is exactly one higher than the previous number. If the number is divisible by 3 then then child does not say the number but "Fizz", and if the number is divisible by 5, the child says "Buzz". If the number is divisible by both 3 and 5, then the child says "FizzBuzz". FizzBuzz is also a popular computer programming exercise. Can you get Python to play it? ***Hint:*** look up the % operator to find out how to test for divisibility.
3. In the code in Section 2, in effect we are testing each open interval $(n, n+1)$ in turn. Can you alter the code so that when you specify some number N and set $d = \frac{1}{N}$, Python tests every successive interval (nd , $(n+1)d$) for a change in the sign of f ?
4. Write a short block of code which will use interval bisection and the intermediate value theorem to search for roots of a given function. Test it on $f : x \to x^x - 2$ in the interval $[1, 2]$.

In [ ]:

```
# Space for answer to Q1:
```

In [ ]:

```
# Space for answer to Q2:
```

In [ ]:

```
# Space for answer to Q3:
```

In [ ]:

```
# Space for answer to Q4:
```