MEI FM FPT: Number theory

Section 1: Programming

Notes and Examples

These notes contain the following subsections:

These notes have been written using Python 3.7.0. Python can be downloaded for free from https://www.python.org/downloads/. There are various programs that can be used to run Python scripts; however, the basic shell (or IDLE) is sufficient for the programs you will meet in this topic.

Mathematical operations and functions in Python

Python can used as a calculator for arithmetic with the basic operations defined using `+`, `-`, `*` and `/`.

For example:

```
>>> 4+3
7
>>> 4-3
1
>>> 4*3
12
>>> 4/3
1.3333333333333333
```

There are also some other numerical functions built in that you will find useful:

- `x//y` gives the whole number part when x is divided by y.
- `x%y` gives the remainder when x is divided by y.
- `int(x)` gives the integer part of x.
- `x**y` gives x^y.

```
>>> 27//7
3
>>> 27%7
4
>>> int(5.2)
5
>>> 5**2
25
```

Note that Python uses the convention that `x//y` rounds towards negative infinity but `int(x)` rounds towards zero and therefore `-25//7` and `int(-25/7)` will give different outputs.

Python uses the convention that $0^0 = 1$.

Python also contains a factorial function in the math library.

To access this first type `import math`.

```
>>> import math
>>> math.factorial(6)
720
```

The math library also contains the function `math.sqrt(x)` for the non-negative square root of a number, which you might find more useful than `x**(1/2)` in some situations.

Activity

Explore using Python for arithmetic.

Writing short programs

In Python programs can be defined using the command `def` followed by a program name and any inputs.

For example the following program will give the value when an inputted number is multiplied by 5:

```
def multfive(n):
    m=5*n
    print(m)
```

This gives the output:

```
>>> multfive(12)
60
```

It is important that you use the correct syntax such as spaces, brackets, colons and indentations. This can be seen by examining each line of the program in turn.

`def multfive(n):`	Any name other than a command can be used for a program. The input variable or variables are listed in brackets after the name. A program with no inputs would just use an empty pair of brackets: e.g. `prog()`. The colon after the definition is required to define everything after this as part of the program.
`m=5*n`	This line has been indented once to show it is part of the program (your Python environment should do this automatically for you).
`print(m)`	This command prints the output to the screen. Note that in some cases `return(m)` will be used. This will be considered later.

Using Python you can write short programs that will systematically check a list of numbers to find when a particular property is true. The main two commands for this are `for` and `if`.

For example, the following program checks which integers are multiples of 5:

```
def fivemults(n):
    for i in range(1,n):
        if i%5==0:
            print(i)
```

This gives the output:

```
>>> fivemults(20)
5
10
15
```

Deconstructing this program:

`def fivemults(n):`	An input of n has been used as a maximum for the program to use.
`for i in range(1,n):`	The program will use a local variable, i, as a counter that will take each of the values $1,2,3,\ldots n-1$. Note that Python will stop and not carry out the code block beneath it when it reaches n.
`if i%5==0:`	This statement is indented to show that it is an instruction to be carried out for each value of i. The program checks if the statement is true, and if it is, carries out the instructions in the block beneath it. Note that Python uses the double equals sign, ==, for checking if a statement is true. The single equals sign, =, is used for setting a variable to a value (as seen in the previous program).
`print(i)`	This statement is indented to denote that the command is to carried out if the 'if' statement is true

If you are going to use the output from one program as the input in another programme then the command `return(x)` might be preferable to `print(x)`.

For example, the first of the following programs checks whether an integer is 1 more than a multiple of 5, and then second program uses this to count the number of integers, less than or equal to a given maximum, whose square is 1 more than a multiple of 5.

```
def fiveone(n):
    if n%5==1:
        return(1)
    else:
        return(0)

def squaresfiveone(n):
    output=0
    for i in range(1,n+1):
        output=output+fiveone(i*i)
    print(output)

>>>squaresfiveone(50)
20
```

It is good practice with any program to perform a check to see if it works correctly for a simple case. In the example above this can be done by checking the program for a small number.

```
>>> squaresfiveone(5)
2
```

This can be confirmed by checking the integers from 1 to 5:

$1^2 = 1$, which is $0 + 1$.

$2^2 = 4$ which is not one more than a multiple of five.

$3^2 = 9$ which is not one more than a multiple of five.

$4^2 = 16$ which is $15 + 1$.

$5^2 = 25$ which is not one more than a multiple of five.