LECTURE 9: FAREWELL TO PYTHON... WELCOME TO MAPLE!

1. A review of PYTHON's features

During the past few weeks, we have explored the following aspects of PYTHON:

- **Built-in types:** integer, long integer, float, complex, boolean, list, sequence, string etc.
- **Operations on built-in types:** $+, -, *$, `abs`, `len`, `append`, etc.
- **Loops:**

```
N = 1000
i = 1
q = []
while i <= N
    q.append(i*i)
    i += 1
```

- **Conditional statements:**

```
if x < y:
   min = x
else:
   min = y
```

- **Functions:**

```
def fac(n):
    if n==0 then:
       return 1
    else:
       return n*fac(n-1)
```

- **Modules:** `math`, etc.
- **Classes:** `Rational`, `Vector`, `Polynomial`, etc.

With these features, PYTHON is a powerful tool for doing mathematics on a computer. In particular, the use of modules makes it straightforward to add greater functionality, e.g. `NumPy` (numerical algorithms).

2. What about MAPLE?

MAPLE is really designed to be an *interactive tool*. Nevertheless, it can also be used as a programming language. It has some of the basic facilities available in PYTHON: Loops; conditional statements; functions (in the form of "procedures"); good graphics and some very useful additional modules called "packages":

- **Built-in types:** long integer, float (arbitrary precision), complex, rational, boolean, array, sequence, string etc.
- **Operations on built-in types:** $+, -, *$, mathematical functions, etc.
- **Loops:**

TABLE 1. Some of the most useful MAPLE packages

Name	Description	Example
`DEtools`	Differential equations	`exactsol(ode,y)`
`LinearAlgebra`	Linear Algebra	`GaussianElimination(A)`
`plots`	Graphics package	`polygonplot(p,axes=none)`
`Statistics`	Statistics package	`X := RandomVariable(Uniform[a,b])`

```
N := 1000 :
q := array(1..N) :
for i from 1 to N do
    q[i] := i*i :
od :
```

- **Conditional statements:**

```
if x < y then
   min := x
else
   min := y
fi:
```

- **Functions:**

```
fac := proc(n)
    if n=0 then
       return 1
    else
       return n*fac(n-1)
    fi :
end proc :
```

- **Packages:** See Table 1.
- **Classes:** None!

The most obvious difference between PYTHON and MAPLE is that the latter does not support *classes*. From our point of view, this is not a problem: MAPLE compensates for the lack of classes by providing so many specialised mathematical features.

3. An example

To illustrate some of the differences between MAPLE and PYTHON, consider the problem of deciding whether a given natural number n is a prime. The obvious method of *trial division* solves the problem as follows: for every natural number $2 \leq m \leq \sqrt{n}$, test whether m is a factor. If a factor is found, then n is composite. Otherwise, it is prime.

The PYTHON version looks like this:

```
def isPrime(n):
    M = int(n**0.5)
    prime = True
    m=2
    while (prime and m <= M):
```

```
        prime = n%m <> 0
        m += 1
    return prime
```

The swiss mathematician L. Euler noticed that the natural number

$$p_n = n^2 + n + 41$$

is prime for every $n < 40$. We can use this to test our function...

The corresponding MAPLE program would look like this:

```
isPrime := proc(n)
        local m,M,prime :
        M := floor(sqrt(n)) :
        prime := true :
        for m from 2 while (prime and m <= M) do
            prime := irem(n,m) <> 0 :
        od :
        return evalb(prime)
        end proc :
```

The most obvious syntactical differences are

In MAPLE, the assignment operator is :=

In MAPLE, every statement must end either with ; or :