Python lab 2: Modules, arrays, and plotting

Dr Ben Dudson

Department of Physics, University of York

4^{th} February 2011

`http://www-users.york.ac.uk/~bd512/teaching.shtml`

From last time...

- Last time we created lots of functions to calculate things like Fibonacci numbers and square roots

From last time...

- Last time we created lots of functions to calculate things like Fibonacci numbers and square roots
- If we had to write every part of a big program, it would take a *really* long time

From last time...

- Last time we created lots of functions to calculate things like Fibonacci numbers and square roots
- If we had to write every part of a big program, it would take a *really* long time
- Instead, whenever possible we should use functions that other people have already written. This not only saves time, but if it is widely used then it is probably efficient and well tested.
- Related functions are grouped together, say one for mathematical functions, one for network connections etc.

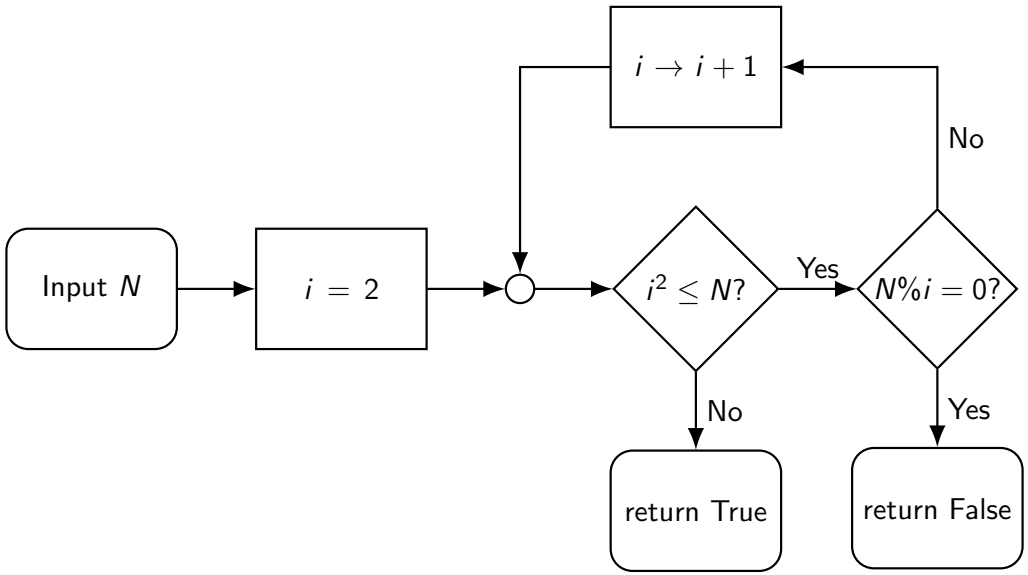
From last time...

- Last time we created lots of functions to calculate things like Fibonacci numbers and square roots
- If we had to write every part of a big program, it would take a *really* long time
- Instead, whenever possible we should use functions that other people have already written. This not only saves time, but if it is widely used then it is probably efficient and well tested.
- Related functions are grouped together, say one for mathematical functions, one for network connections etc.
- These groups of functions go by different names in various languages: **libraries**, **packages** and **modules**. Python uses a hierarchical system where functions are grouped into modules, which in turn are grouped into packages.

Python modules

Say we write a function to test if a number is prime

```
def isPrime(N):
    i = 2
    while i**2 <= N:
        if N % i == 0:
            return False # Not a prime
        i = i + 1
    return True # Is a prime
```



Python modules

- We save this in a file, say `prime.py`
- Later we might want to use this function in a second file

Python modules

- We save this in a file, say `prime.py`
- Later we might want to use this function in a second file
- Rather than copying and pasting the function, we can just **import** it

```
import prime

i = input("Enter a number:")
if prime.isPrime(i):
    print "Prime number"
else:
    print "Not a prime number"
```

Python modules

- We save this in a file, say `prime.py`
- Later we might want to use this function in a second file
- Rather than copying and pasting the function, we can just **import** it
- This allows us to use huge collections of functions which people have written. On your CD are packages for scientific plotting, making 3D animations, interactive web pages, and writing games

Python modules

Two ways to import modules in Python. The simplest way is:

```
import prime
```

If we do this, then we need to use prime.isPrime with the module name before the function name.

Python modules

Two ways to import modules in Python. The simplest way is:

```
import prime
```

If we do this, then we need to use prime.isPrime with the module name before the function name.
If typing prime. all the time gets tiresome, you can give it a different name using **import** ... as ...

```
import prime as p
```

Now we can use p.isPrime instead.

Python modules

Two ways to import modules in Python. The simplest way is:

```
import prime
```

If we do this, then we need to use prime.isPrime with the module name before the function name.
If typing prime. all the time gets tiresome, you can give it a different name using **import** ... as ...

```
import prime as p
```

Now we can use p.isPrime instead.
If we don't want this extra typing we can import the function

```
from prime import isPrime
```

This allows us to just use isPrime without the module name at the start.

NumPy

- All these ways to import functions exist to help manage big programs. If there are two different modules, each of which contains a function with the same name then we need a way to distinguish them.

NumPy

- All these ways to import functions exist to help manage big programs. If there are two different modules, each of which contains a function with the same name then we need a way to distinguish them.
- The easiest way to import modules is to use

```
from prime import *
```

  This is not encouraged for large programs since it can result in problems when names collide. For some modules, this is however the most convenient way

NumPy

- All these ways to import functions exist to help manage big programs. If there are two different modules, each of which contains a function with the same name then we need a way to distinguish them.
- The easiest way to import modules is to use

```
from prime import *
```

  This is not encouraged for large programs since it can result in problems when names collide. For some modules, this is however the most convenient way
- The main module we're going to be using for scientific calculations is Numerical Python, or NumPy. To use these functions,

```
from numpy import *
```

Arrays

- So far this course we've been dealing with variables which store one number at a time.
- Particularly in scientific fields, we often want to deal with lots of numbers, such as tables of measurements
- If we had to use a different variable for each of these numbers it would be nearly impossible to write programs to handle thousands or millions of numbers

Arrays

- So far this course we've been dealing with variables which store one number at a time.
- Particularly in scientific fields, we often want to deal with lots of numbers, such as tables of measurements
- If we had to use a different variable for each of these numbers it would be nearly impossible to write programs to handle thousands or millions of numbers
- The solution to this is to store many numbers in a single variable. These are called **arrays**, and are like vectors in mathematics: Instead of x_i Python uses `x[i]`
- This is the main purpose of NumPy: to provide an efficient way to handle large arrays of numbers

Creating arrays

There are several ways to create arrays in Python, but the one you'll probably use most is the linspace function which is part of NumPy. To see how it works, here's an example

```
from numpy import *
x = linspace(0, 4, 9)
print x
```

Creating arrays

There are several ways to create arrays in Python, but the one you'll probably use most is the linspace function which is part of NumPy. To see how it works, here's an example

```
from numpy import *
x = linspace(0, 4, 9)
print x
```

This will print out

```
[ 0.   0.5  1.   1.5  2.   2.5  3.   3.5  4. ]
```

The brackets at the beginning and end mean it's an array. It goes from 0 to 4, and contains 9 numbers.

Creating arrays

There are several ways to create arrays in Python, but the one you'll probably use most is the linspace function which is part of NumPy. To see how it works, here's an example

```
from numpy import *
x = linspace(0, 4, 9)
print x
```

This will print out

```
[ 0.   0.5  1.   1.5  2.   2.5  3.   3.5  4. ]
```

The brackets at the beginning and end mean it's an array. It goes from 0 to 4, and contains 9 numbers.
Therefore, linspace(min, max, n) creates an array which goes from min to max, and contains n numbers

Using arrays

Once we have an array, the big advantage is that we can treat it like we would a single number. For example, mathematical operations are done on each of the numbers in an array

```
from numpy import *
x = linspace(0, 4, 9)
print x
x = x * 2
print x
```

Using arrays

Once we have an array, the big advantage is that we can treat it like we would a single number. For example, mathematical operations are done on each of the numbers in an array

```
from numpy import *
x = linspace(0, 4, 9)
print x
x = x * 2
print x
```

This will produce the result

```
[ 0.   0.5  1.   1.5  2.   2.5  3.   3.5  4. ]
```

Using arrays

Once we have an array, the big advantage is that we can treat it like we would a single number. For example, mathematical operations are done on each of the numbers in an array

```
from numpy import *
x = linspace(0, 4, 9)
print x
x = x * 2
print x
```

This will produce the result

```
[ 0.   0.5  1.   1.5  2.   2.5  3.   3.5  4. ]

[ 0.   1.   2.   3.   4.   5.   6.   7.   8. ]
```

Looping over arrays

If we want to run a set of commands once for each element in an array, there are two main ways. The first is to use a **while** loop, using len to get the length of the array

```
from numpy import *
a = linspace(0, 1, 11)
i = 0
while i < len(a):
    print a[i]
    i = i + 1
```

Looping over arrays

If we want to run a set of commands once for each element in an array, there are two main ways. The first is to use a **while** loop, using len to get the length of the array

```
from numpy import *
a = linspace(0, 1, 11)
i = 0
while i < len(a):
    print a[i]
    i = i + 1
```

Alternatively, we can use a different form of loop called a **for** loop

```
from numpy import *
a = linspace(0, 1, 11)
for v in a:
    print v
```

Plotting

- One of the most common things we will want to do with arrays is plot them.

Plotting

- One of the most common things we will want to do with arrays is plot them.
- There are several different plotting modules for Python, but the one we'll be using is Matplotlib.

```
import matplotlib.pyplot as plt
```

Plotting

- One of the most common things we will want to do with arrays is plot them.
- There are several different plotting modules for Python, but the one we'll be using is Matplotlib.

```
import matplotlib.pyplot as plt
```

- This gives us the plot function (amongst others) which we can use to plot graphs. For example:

```
from numpy import *
import matplotlib.pyplot as plt

x = linspace(0, 5, 10)
plt.plot(x, x**2)
```

Summary

- To use arrays, first import the NumPy module using

  from numpy **import** $*$

- Arrays can be created using linspace(min,max,steps) and also arange, zeros, ones, and random.random

Summary

- To use arrays, first import the NumPy module using

```
from numpy import *
```

- Arrays can be created using linspace(min,max,steps) and also arange, zeros, ones, and random.random
- Once created, arrays can be used much like other variables
- To plot graphs, first import the Matplotlib module

```
import matplotlib.pyplot as plt
```

- You can then use plt.plot(x, y) to plot graphs, where x and y are arrays (of the same length)

Summary

- To use arrays, first import the NumPy module using

```
from numpy import *
```

- Arrays can be created using linspace(min,max,steps) and also arange, zeros, ones, and random.random
- Once created, arrays can be used much like other variables
- To plot graphs, first import the Matplotlib module

```
import matplotlib.pyplot as plt
```

- You can then use plt.plot(x, y) to plot graphs, where x and y are arrays (of the same length)

Getting help: Python includes lots of built in help for modules and functions. To see what a function does, there is the help() function. For example, to see how to use linspace works, run

```
help(linspace)
```

`http://www-users.york.ac.uk/~bd512/teaching.shtml`