Introduction to Python

Thomas Dack
Scientific Computing, STFC

Originally written by Adrian Coveney, SCD STFC

Lesson One

- Learn about Python, a scripting language
- Use Python as a calculator
- Storing information
- Getting input from the user
- Using functions
- Using Python to make decisions
- Loops and branches in programs
- Guess the number game

Key

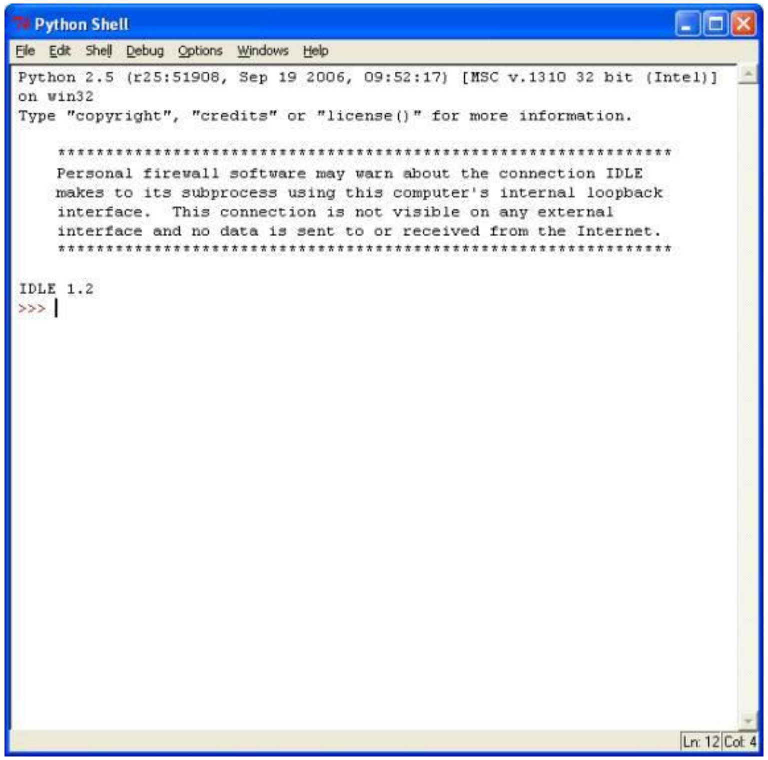
- **Bold is for things which are common computing terms**
- Blue is for things you need to do to work along with me
- Red is for outputs wich Python will print out
- Green is for Strings
- *Italics are things which need to be written into a program*

What is Python?

- Scripting language
- Like a script for a play or film
- Tells the computer what to say and do

IDLE – The easy Python interface

- To start up Python, we need to run the IDLE editor
- Start -> Python3.2 -> IDLE
- You should get a window that looks like this



```
Python 2.5 (r25:51908, Sep 19 2006, 09:52:17) [MSC v.1310 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 1.2
>>> 
```

What are we looking at?

- **Console**: text terminal where we can enter commands
- Direct access to the Python **Interpreter**
- What you type in here is computed as soon as you press return

So what now?

- Computers are great at doing maths (simple maths anyway)
- Python makes a really good calculator
- You can just use it by typing in what you want like a normal calculator
- Try 2+2 and press return

Well, almost a calculator

- Similar, but some things are different

Addition	**+**
Subtraction	**-**
Multiplication	*****
Division	**/**

Storing results

This is a bit like the memory on a calculator:

```
ans = 5*5
print (ans)
25
```

You can also use it in other expressions:

```
ans+10
35
```

Variables

The stored results are actually called **variables** and don’t need to be called "ans"

They can be anything you like, a little like algebra, here we use x, y and z

```
x = 5*5
y = 27-12
z = x+y
print (z)
40
```

Variable names

In old languages variable names needed to be short and sometimes obscure

```
a = 1 b = 22.3 aa = 52
```

Now, and particularly in Python, we can use descriptive names

```
number_of_turns_in_the_game = 10
numberOfTurnsInTheGame = 10
```

Exercise 2 - Variables

```
a = 15 + 5
b = 20 - a
c = 15 / 5
d = 10 + c
ans = a + b + c + d
```

What’s the answer?

36!

These variables are stored until you close python, or rewrite them

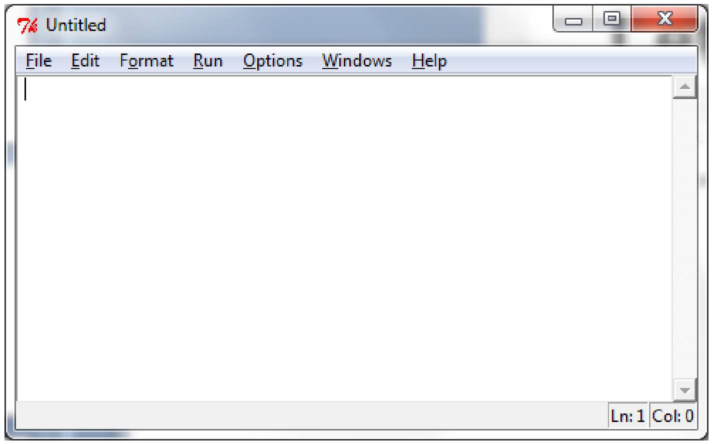
…

… so we can get all the parts back

```
print(a,b,c,d)
20 0 3.0 13.0
```

Let’s write a program

- Typing into the terminal is great for getting an idea of what's happening, but we want to keep our code.
- In IDLE use File->New Window
- This creates a new window which looks like this:



What’s a program?

- A program is a list of commands, which are executed one after the other
- Here is the example program, which we will write

```
welcome = “Hello World”
print(welcome)
```

Running the program

- To run a program, we first need to save it

  File -> Save or ctrl-s

- Once it is saved we can run the program

  Run -> Run Module or F5

- You should get the following output

  ```
  Hello World
  ```

What’s happening?

- The **program** we have written is being run line by line by the **interpreter**
- Line 1 : welcome = “Hello World”
  - defining a new variable containing the string we want to use for our welcome message
- Line 2 : print(welcome)
  - prints our welcome message to the screen

Why "Hello World“?

- It's the classic first thing to do with any language
- Look at the Wiki page for "Hello World Program"

```
Hello World in red
Hello World in green
Hello World in blue

Hello World with free char width
Hello World with block char width 10
Hello World with block char width 12

Hello World with opacity 100%
Hello World with opacity  50%
Hello World with opacity  10%

Hello World with shadow

Hello World from pspdev
```

User input

- Getting the computer to do things is cool, but it’s a bit boring
- Most programs want some input
- Try the following in the console

```
name = input()
print ("Hello " + name)
```

But text shouldn’t add!

- Correct, in Python, anything inside a set of quotation marks " " is called a **string**
- In Python, when dealing with strings you can add them: it’s called **concatenation** and it just puts them one after the other

```
print("Hello World")
print("Hello" + " " + "World")
```

Exercise 3 - Customise your program

- Ask the user for some info, like their name, and then use it in the program
- Remember to print a statement to ask what your user should enter into the space

```
print ("Please enter your favourite colour?")
```

Imports and functions

- Not everything is included in Python as standard, lots of content is included from other places
- Most other content comes in the form of **functions**, which we have already been using

```
print("Hello World")
name = input()
```

The random module

- Random numbers are one of the main things in a lot of games, just think of all the dice and cards in games

```
import random
value = random.randint(1,6)
print(value)
```

Exercise 4: Roll the dice

- Add a simulated dice roll to the hello world program
- Extras
  - Add multiple dice, or change the number of sides on the dice
  - Display results of dice added together
  - Ask the user for input about the type of dice

```
size = input()
roll = random.randint(1,int(size))
```

Conditionals

- **Conditionals** result in a logical outcome, such as True or False

- Some conditions are

Greater than	**>**
Less than	**<**
Equal to	**==**
Not equal to	**!=**

Boolean Variables

- These are **True** and **False** and you can set a variable to be these as you would any other
- For example:

```
x = True
y = False
```

Exercise 5 – True or false?

- Try some of the following statements

```
a = 5          a+b == b+a
b = 10         a-b == b-a
a > b          a*b == b*a
a < b          a/b != b/a
```

- Experiment with some other comparisons

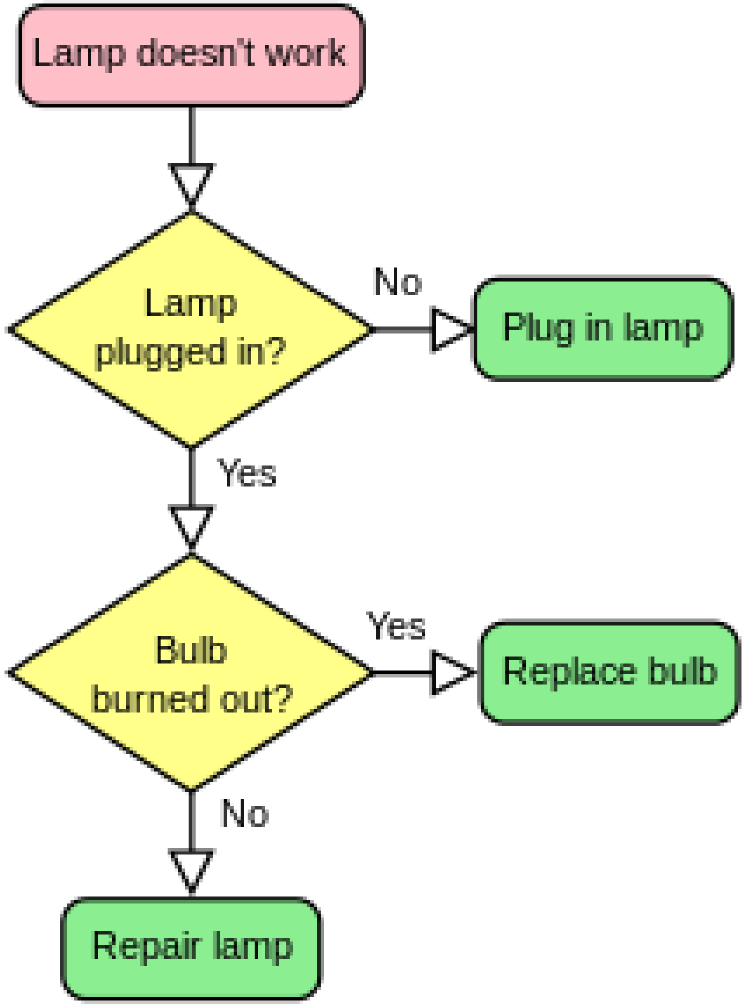
Flow control

Not everything happens in order, sometimes decisions must be made and code run (or not) depending on the result

If

While

For



While

Loops are useful for repeating bits of code without having to retype the code

A **while** loop will keep going round while the condition is met

```
x = 0
while (x < 5):
    print (x)
    x = x+1
```

While – Looping in the program

The **while** loop is the main part of the program

```
while guessesTaken < 6:
    # Do something
    guessesTaken = guessesTaken + 1
```

Note the 4 spaces before everything which happens inside the loop - the loop ends when things aren't spaced in any more

If – Choosing what to do

- Code inside an **if** statement only gets executed if the condition is true

```
if guess < number:
        print("Your guess is too low.")
```

- Once again, when the indenting ends, the **if** statement is over

If statements

Let’s look at an **if** statement

```
if (x > 5) :
    print (“x is bigger than 5”)

else :
    print (“x is less than or equal to 5!”)
```

Break

If for some reason we want to end the while loop before its natural end, we can always use the **break** statement, which gets us out of the loop

```
if guess == number:
    break
```

Exercise 6 – Guess the number

Make a ‘Guess the number’ game, and add any or all of the following features

- Customise the program
- A difficulty setting, which the player can select
- A 2-player version

Boolean operators: and

AND	True	False
True	True	False
False	False	False

Boolean operators: or

OR	True	False
True	True	True
False	True	False

Boolean operators: not

not True

not False

Boolean operators: not

- You can also check if something is “not” a Boolean

```
not True
not False
```

Boolean operators

Boolean operators (**and or not**) are used to test multiple statements

```
while cave != ‘1’ and cave !=‘2’
while cave != ‘1’ or cave !=‘2’
not True
```

Functions

In Python you can write **functions** for blocks of code that you'll want to use many times

```
def greeting(name):
    print("Welcome " + name)

greeting("Sophy")
greeting("Angus")
greeting("Callum")
```

Exercise 7

- Use functions in your “guess the number” game
  - A guess quality indicator (e.g. hot, warm, cold)

Exercise 8

- Write a program that tells you if a number is prime
  - what is a prime number?
  - special cases

Exercise 9

- Hangman
  - How many guesses?
  - How to display wrong guesses?
  - Deal with bad input – capital letters, numbers, punctuation