

Practical Sheet 1

Interactive Python and Python Scripts

Aims

Section		Aim
1	Writing the first Python program	Use the IDLE Python program development environment to write your first Python program.
2	Numbers and Strings:	Understand Python expressions for numbers and string
3	Variables – names values	Understand how a variables is a name for a value and can change
4	Input: Getting Data from the User	Learn how to use 'print' and 'input'
5	Breaking it Down: Step by Step	Understand how variables are used to solve a problem in stages.

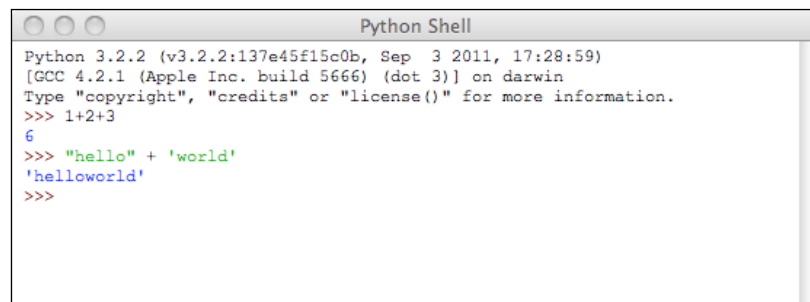
Related topics

- **Topic 1.1 First Python Program**
- **Topic 1.2 Python numbers and strings**
- **Topic 1.3 Python variables**

1 Writing the First Program

1.1 Starting IDLE

Start-up the Python development environment 'IDLE'. Your screen should look something like:



```
Python 3.2.2 (v3.2.2:137e45f15c0b, Sep  3 2011, 17:28:59)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> 1+2+3
6
>>> "hello" + "world"
'helloworld'
>>>
```

The IDLE interface has one main window 'Python Shell'; others window are opened for each file.

1.2 Saving Python Scripts in Files

A Program is a Document

A program is a document that is held in a file (just like a word processor or spread sheet file). We will assume that a Python program is a single file, with extension '.py'. As you complete each exercise, create a new file for each stage. That way you can look back at the programs you have written.

You can create a new program as follows:

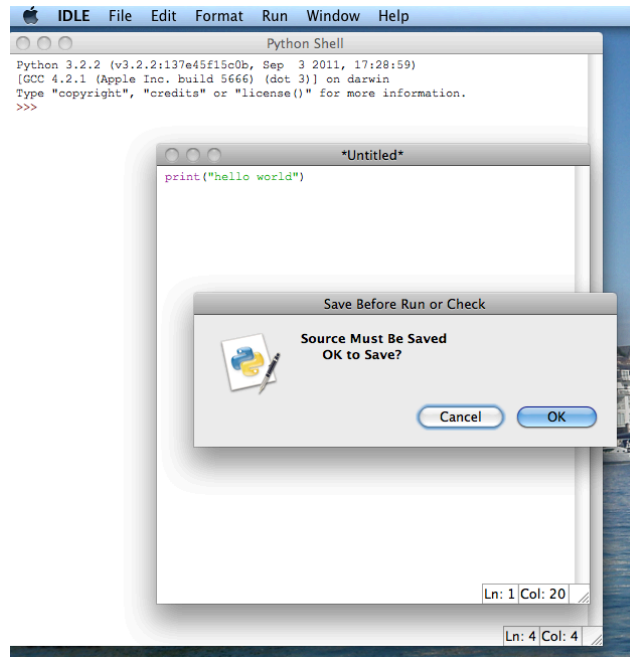
1. Menu: File/New Window

2. Edit text in the new window
3. Menu: Run/Run Module or Run/Check module. You will be prompted to save the file. Give it extension .py.

Exercise 1.1. To get used to IDLE try lots of different ways of writing the 'hello' Write the following script:

```
print("Hello world")
```

Here is what it looks like on my computer:



1.3 Output: Printing

A program should output something! For this, you use the 'print' function.

Exercise 1.2. To get used to IDLE try lots of different ways of writing the 'hello' Here are some examples to try to show how print can be used:

```
print("word 1", "word 2")
print("My age=", 21, " again!")
print("Hello ", end=" ")
```

1.4 Writing Scripts (Programs) versus Using the Shell

For beginners, there are two ways to use Python. It is flexible, but it can be *confusing*.

Writing Scripts

A program or script is a document. The text of the program is in a file, with a name.

To see what the program does, you must tell IDLE to 'run' the program.

A program must have a 'print' in it; otherwise nothing happens.

Large programs are always written in files.

Using the Shell

The 'Python Shell' works like a calculator: type an expression at the prompt >>>

You get the answer as soon as you hit return.

It is enough to type an expression; the value is echoed.

The shell is good for trying out small bits of program.

2 Numbers and Strings

Alert: Using the Shell

In this section we are using the Python Shell. Be sure you understand the difference between this and writing a script in a file.

Python can do arithmetic. Enter an arithmetic expression. Here are some examples:

```
>>> 1 + 1
2
>>> 10 - (3 * 2)
4
>>> 2 ** 8
256
```

Exercise 2.1

- Try some expressions of your own.
- What is the difference between $10 - 5 - 2$ and $10 - (5 - 2)$
- Python has two types of division: what is the difference between $10/4$ and $10//4$
- The '%' is an operator – as in $13 \% 3$. Work out what it does.

2.1 Fun with Strings

A string is a list of letters, digits or symbols. Some examples are:

```
"My name is George"      'My name is Dave'      "Vince hates George"
```

You will see that there is a quote mark at the start and end of a string: either " or ' (but it must be the same one at either end).

String can be concatenated (joined). Try:

```
>>> "My name" + ' is George' + ' my friend is Dave'
```

Other operations on a string:

- Find its length using 'len()'. Example: `len("william")`
- Select a character or range of characters using '[']. Examples
 - `"william"[0]`
 - `"william"[6]`
 - `"william"[1:4]`

Exercise 2.2. Try the following:

- Concatenate some strings.
- Find the length of different strings using 'len'.
- Investigate the behaviour of the 'slicing' operator `[m:n]` where n and m are numbers.

There are lots more possible expressions for strings. There are also other types of expressions (not just numbers and strings) that we will encounter in Python.

2.2 Errors: Talking Nonsense

Python does not produce as many errors as some other languages (e.g. Java) but nevertheless some programs make no sense. Here are some examples:

Let's try to concatenate a string and a number:

```
>>> "my age is" + 21
```

```
Traceback (most recent call last):
  File "<pyshell#35>", line 1, in <module>
    "my age is" + 21
TypeError: Can't convert 'int' object to str implicitly
```

Whoops! Only strings can be concatenated.

```
>>> 21 / 0
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    21 / 0
ZeroDivisionError: division by zero
```

Whoops! Division by zero has no answer.

```
>>> "william"[7]
Traceback (most recent call last):
  File "<pyshell#36>", line 1, in <module>
    "william"[7]
IndexError: string index out of range
```

Whoops! There are only 7 letters in my name (with numbers [0] up to [6]).

Exercise 2.3 Try the following:

- Check each of errors above and check you understand its cause.
- Write something you expect to give an error.
- Write an expression that is very large e.g. try `n ** (n ** n)` for `n = 2, 3, 4 ...`
What happens when the numbers become very large?
- Explain the difference between the follow two expressions (try them out):

```
>>> "21" + "21"
>>> 21 + 21
```

3 Variables – Names for Values

An important idea is giving a name to a value: a variable. In Python variables are created when a value is assigned to them:

```
pmName = "David Cameron"
```

Here the variable `pmName` is assigned a string. A variable can change value; depending on the result of the election we may need to change the value assigned to the variable:

```
pmName = "Ed Miliband"
```

We can use the variable as if it were a string. For example:

```
>>> pmName = "David Cameron"
>>> len(pmName)
13
>>> pmName.index('v')
2
>>> pmName[0:4]
David
>>> pmName.index(' ')
5
```

Here, we have used the operation `string.index(character)` to find the number of the first space character. We could use this to find the first name and so to find the mayor's first name.

Exercise 3.1. Try the following:

- Create variables for your first and last name. Concatenate them. (Did you remember a space?)
- Create variables for the ages of your children / partner / friends/ football team and add them up.
- Write an expression to find the last name of a person from a string with their first and last names, separated by a space.

3.1 More Errors and Utter Nonsense

Variables create a new way to talk nonsense:

```
>>> yourname
Traceback (most recent call last):
  File "<pyshell#28>", line 1, in <module>
    yourname
NameError: name 'yourname' is not defined
```

Whoops! I have used a variable `yourname` without having first given it a value.

In the errors above, the expression seems to make sense but cannot be answered (i.e. evaluation fails). A **syntax error** occurs when Python cannot even understand the words (so evaluation never starts).

```
>>> 1001nights = 'tales'
SyntaxError: invalid syntax
>>> 'not a string'
SyntaxError: EOL while scanning string literal
```

Exercise 3.2.

- Explain the error in each case above.
- Try the following. Although all look wrong, some are not errors – try to explain why not.

```
>>> 1 + / 2
>>> 1 / - 2
>>> 1_2& + 3
>>> * 3
>>> 2 ++ 1
>>> 2 + + 1
```

4 Input: Getting Data from the User

The script would be more useful if the user could provide data. Here is an example of the `input()` function:

```
name = input("What is your name?")
greeting = "Hello, " + name + "! Welcome"
print(greeting)
```

4.1 Numbers or Strings?

Try this simple program that shows how good Python is at sums.

```
print("873 + 1276", "=", 873 + 1276)
```

This program includes the character

873 + 1276

twice. However, the effect is rather different:

- The first occurrence is a string – Python prints the characters entered
- The second occurrence is an expression – Python calculates and then prints the answer

Here is another way to write the same program. Which version do you prefer?

```
mysum = "873 + 1276"  
myanswer = 873 + 1276  
print(mysum, "=", myanswer)
```

Exercise 4.1

- Use the same idea to show the results of other calculations (e.g. 365 divided by 12).
- Write another version of the program that uses the sign + several times, sometimes for addition and sometimes for joining (concatenating) strings

4.2 Input of Numbers: A Problem

17 is your favourite number (*if it isn't, change the number*). Here is a script that is intended to allow you to enter a number and have the result multiplied by 17. For example, if you enter '3' you expect the answer '51'.

```
x = input("Enter x? ")  
print("17*x", " = ", 17*x)
```

Test this program. Does it do what you expected? Can you explain why not?

Think about this before turning the page

4.3 Converting between Strings and Integers

The problem is that 'strings' are being confused with 'numbers'. For example, when you choose $x=3$, the program becomes:

```
print("17*x", " = ", 17*"3")
```

You can enter the expression at the interpreter prompt:

```
>>> 17*"3"
```

Compare this with:

```
>>> 17*3
```

In Python, variables can be of different types. We have encountered the following types:

- An expression or variable can be a number
- An expression or variable can be a string

4.4 The int() Function: Strings to Integers

The function `int()` can be used to convert between strings and numbers. Compare the following:

```
>>> int("5") + int("3")
>>> "5" + "3"
```

5 Breaking it Down: Step by Step

Variables can be used to avoid writing complex expressions. For example, suppose that we wish to calculate the volume of a column (cylinder)

```
>>> height = 10
>>> diameter = 3
```

then we can use the (rather complex) formula:

```
>>> volume = height * 3.142 * ((diameter/2) ** 2)
>>> volume
70.695
```

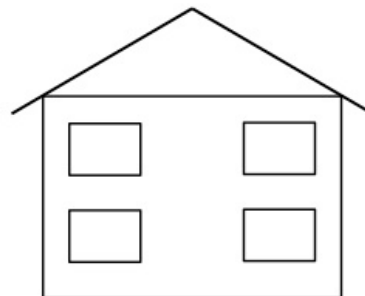
However, it is clearer to take it in stages by using some extra variables:

```
>>> radius = diameter / 2
>>> area = 3.142 * (radius ** 2)
>>> volume = height * area
>>> volume
70.695
```

Exercise 5.1 The following exercise follows the pattern of a maths question:

A painter is calculating the area of the back wall of a house. The house has 4 windows, which he does not need to paint.

Use height and width of both the house and the windows to calculate the area to be painted. Ignore the roof.



Solve the problem twice in Python. Introduce some variables to breakdown the calculation into stages. Here are the dimensions:

```
>>> hh = 5
>>> hw = 12
>>> ww = 1.2
>>> wh = 0.8
```

6 Summary

- Python allows you to type expressions (it's an interpreter).
- Python handles numbers and string. Python has lots of built-in operators and functions. We have learnt about
 - Arithmetic operators `+` `-` `*` `/` `//` `**` `%`
 - Operators for strings (and lists): `+` `[]` `[:]` `.index()`
- Python gives an error message when the script is meaningless.
- A Python script is a sequence of instructions in a file.
 - Use `print()` for output and `input()` to a string from the user.
 - The `int()` function converts strings of digits to numbers.
- A variable gives a name to a value; this value can change.
- Variables can be used to break things down into simpler steps.