UniSA STEM

COMP 1039 Problem Solving and Programming

Practical 1 (Windows Version)
Introduction to Python and IDLE

Getting Started – Introduction to Python and IDLE

Using the computers on campus

Internal students are required to attend weekly practical sessions in order to undertake their practical work. Practical sessions are held in computer pools. You will be required to login to a computer in order to use it.

Login to the computer using your username and password.

Your username and password are your electronic signature-you can use them to access the University's online resources and services. If you are a new student you'll find them in the letter the University sends to you in January (or February for late offers).

a) Type in your user name in lowercase (this is your UNINET username – e.g. bondj007).

b) You will then be asked to enter your UNINET password.

The password allocated to you at the beginning of your program is the first four letters of your family name (add x if it's less than four letters) and the day and month of your date of birth. For example, if your surname is Ng and your date of birth is 17 March, your password will be ngxx1703.

To protect the privacy of your information it is important that you change your password from the default one described above. Your new password must be 6 to 8 character long and cannot start with a number. To help you in case you forget your password setup a password reset account.

If you forget your password try using the password reset facility. If you have not setup your your password reset facility contact the IT Help Desk (Monday to Friday, 8.30am to 9.00pm) by phoning 25000 (internal phones) or +61 8 8302 5000 (externally). Country or interstate callers can phone 1300 558 654 for the cost of a local call.

You can get further information or log a service request on line at: www.unisa.edu.au/ITHelpDesk

External Students

Please ensure that Python is installed on your home computer so that you are ready for this week's practical. See the 'Resources' page on the course website for information on installing Python.

Introduction to Python and IDLE

The only way you can learn how to write programs is by doing it – **practise is the key**!

If you have already done some programming in another language, the first couple of practicals may seem slow, but you should work through them to become familiar with the low level Python constructs.

1. Log in to your computer.

2. Start the IDLE environment – you can do this by using the menu system on your computer.

   On Windows:

On Windows 8 (and above), press the Windows key on your keyboard, when the large start screen appears (start is written at the top left hand corner of the screen), begin typing the name of the program (you don't have to click anywhere, just start typing IDLE(or Python) once you are on the start screen). You will notice that as you begin to type the start screen will change and display all programs that match your search term. Select IDLE (Python GUI).
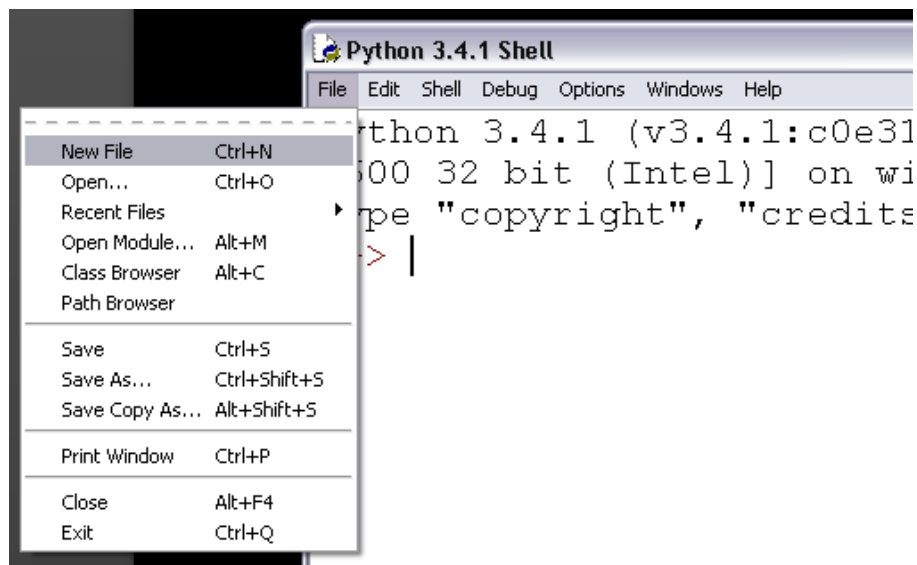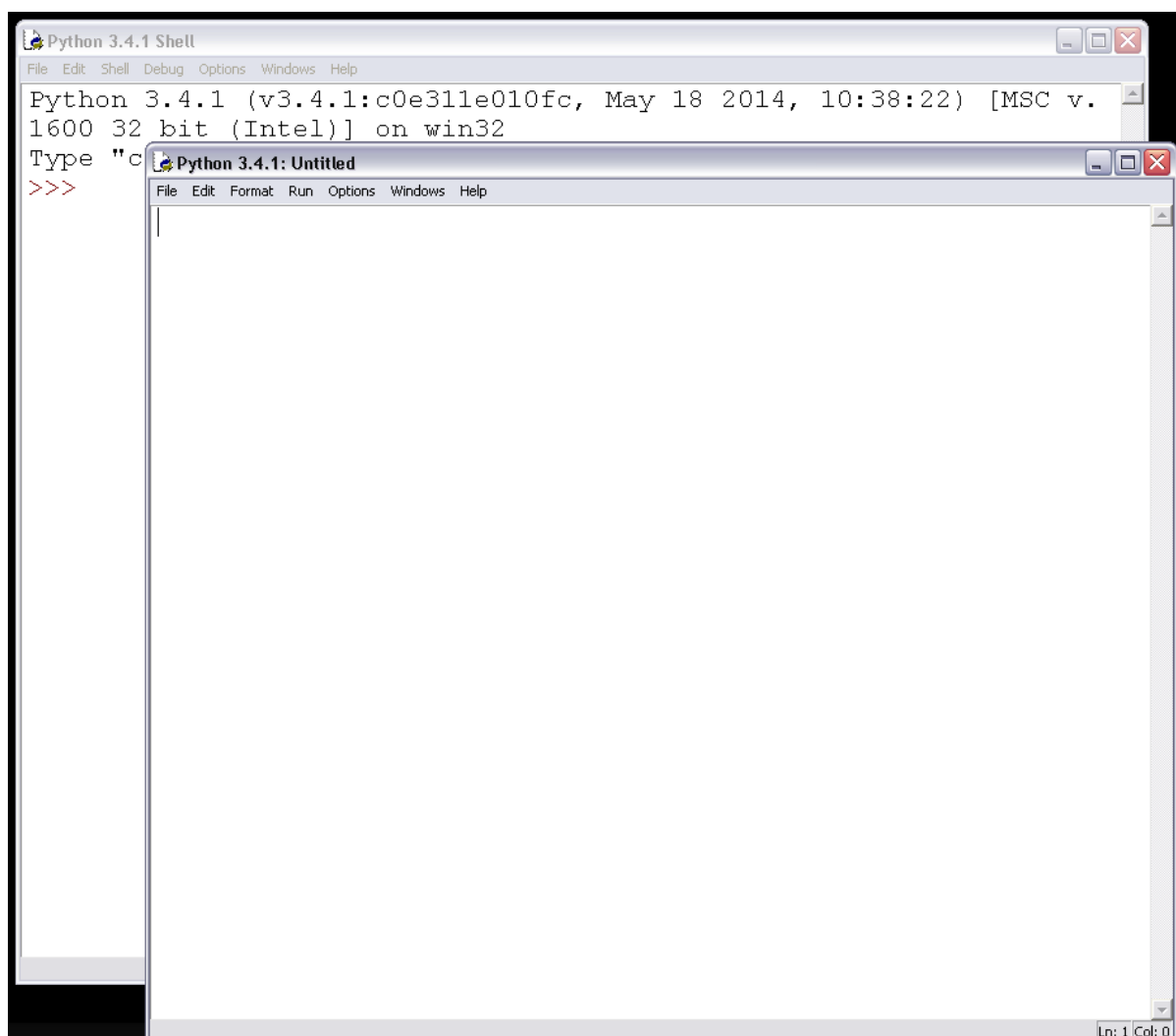
On Windows, the Python shell should look like this:
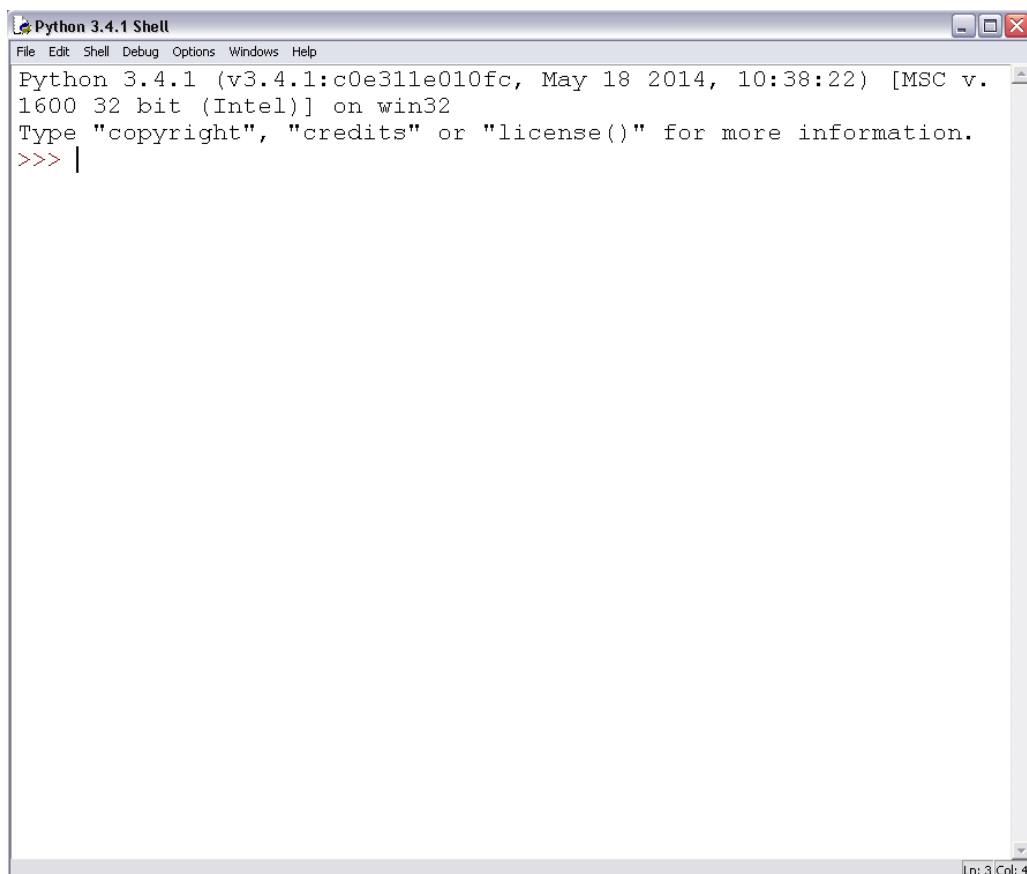
Creating, Saving and Running a Python Program

3. Let's open an editor window – to do this, select `File -> New File`
This will open the editing window where you may enter your Python program (next step).

This will open the editing window where you may enter your Python program (next step).

4. Enter the following program from the slides.

As seen below, your file should include appropriate comments consisting of the following- file name, your details, version number and date, brief program description and the University's academic misconduct statement. ***All practical and assignment work you produce in this course should have the following comments.***

```
#
# File:      welcome.py
# Author:    your name
# Email Id:  your email id
# Version:   1.0  5 August 2014
# Description:  Practical 1 - displays a welcome message to the screen.
#    This is my own work as defined by the University's
#    Academic Misconduct policy.
#


# Display welcome message to the screen
print('Welcome to Problem Solving and Programming!')
```
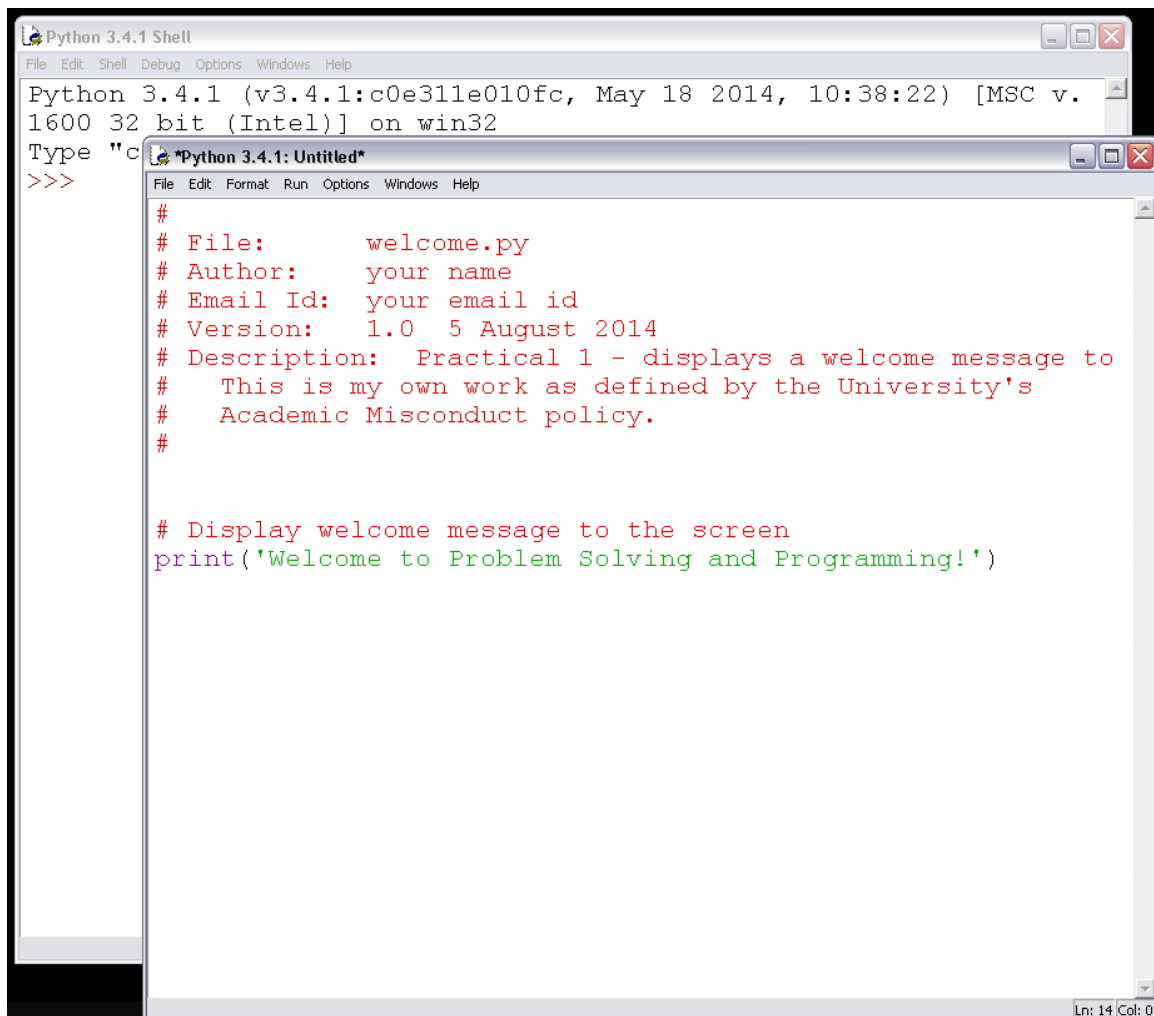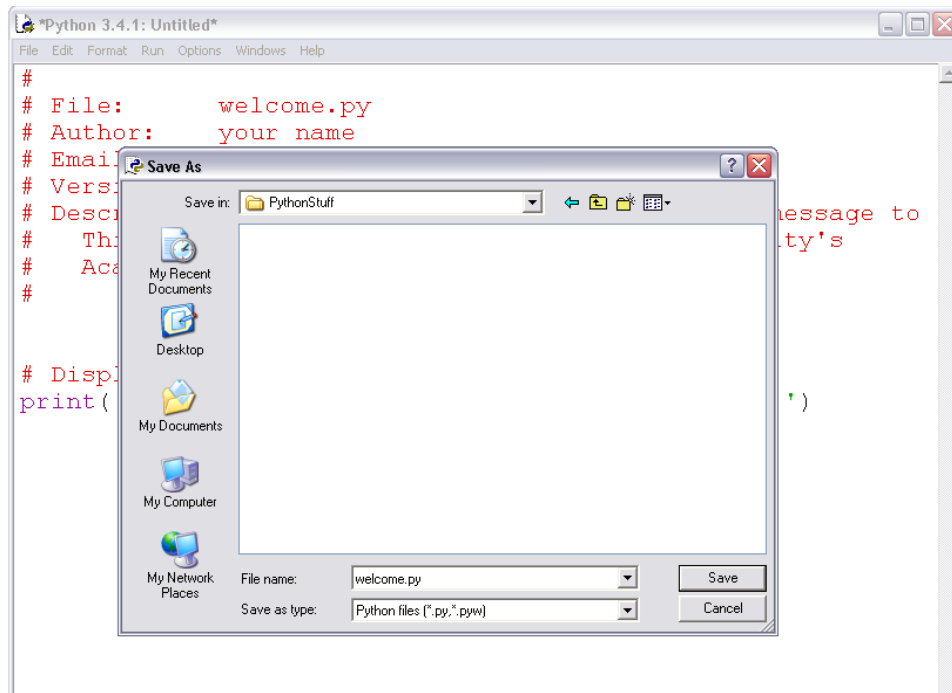
5. Once you have entered your Python program, you may run it by selecting `Run -> Run Module`
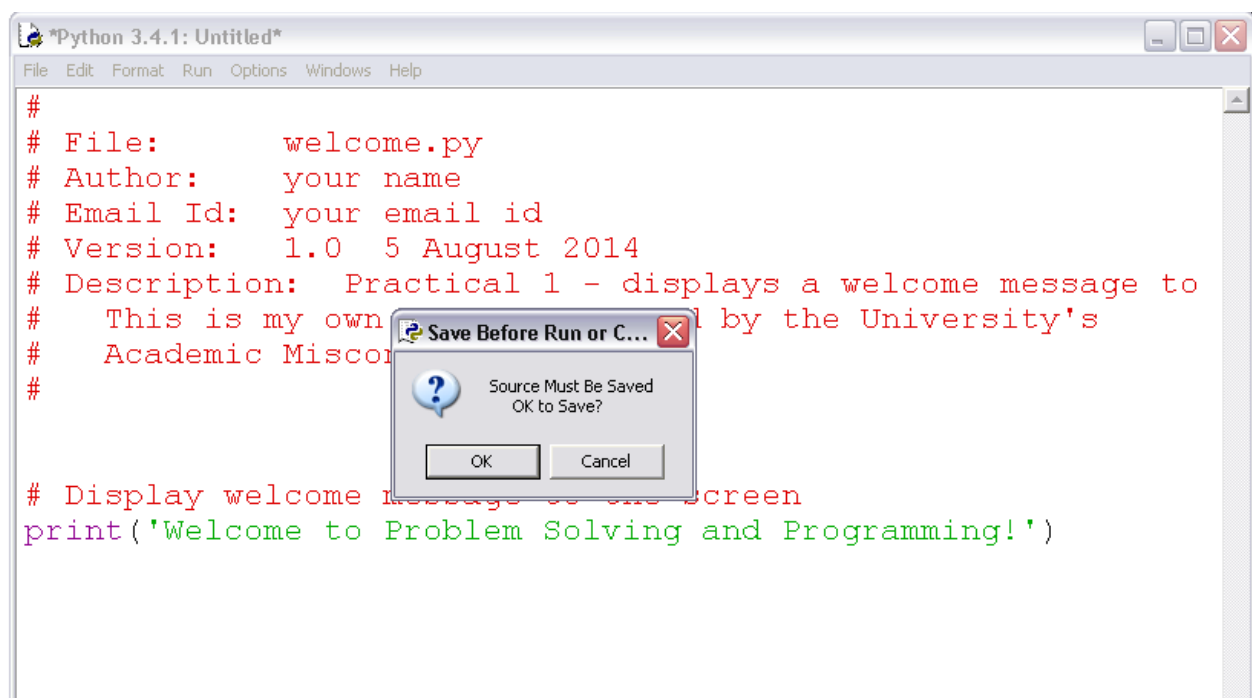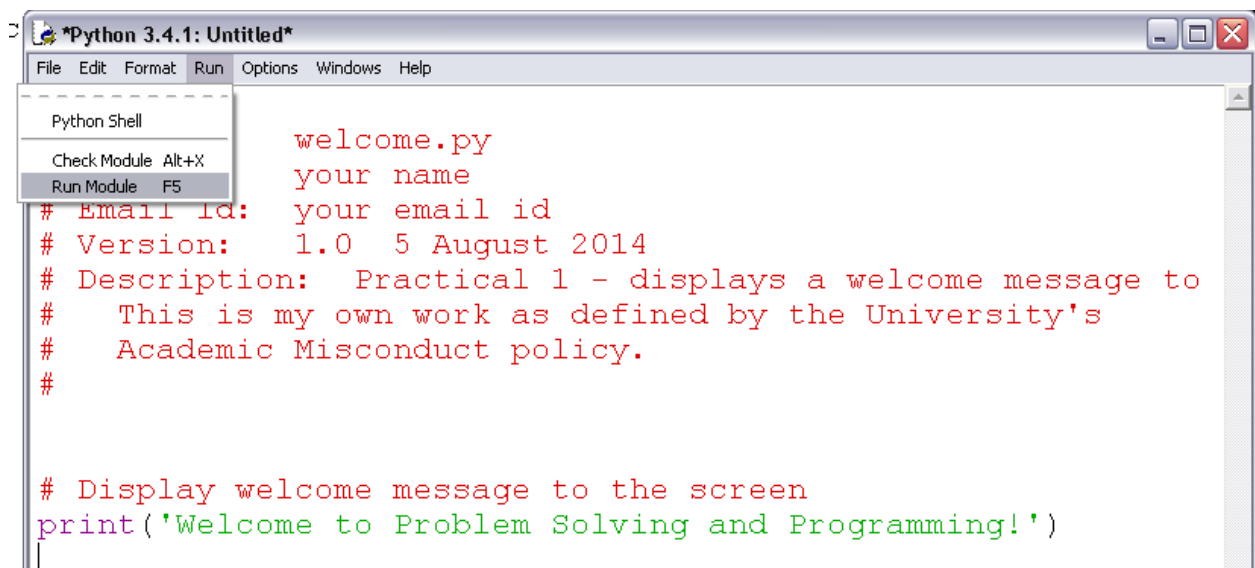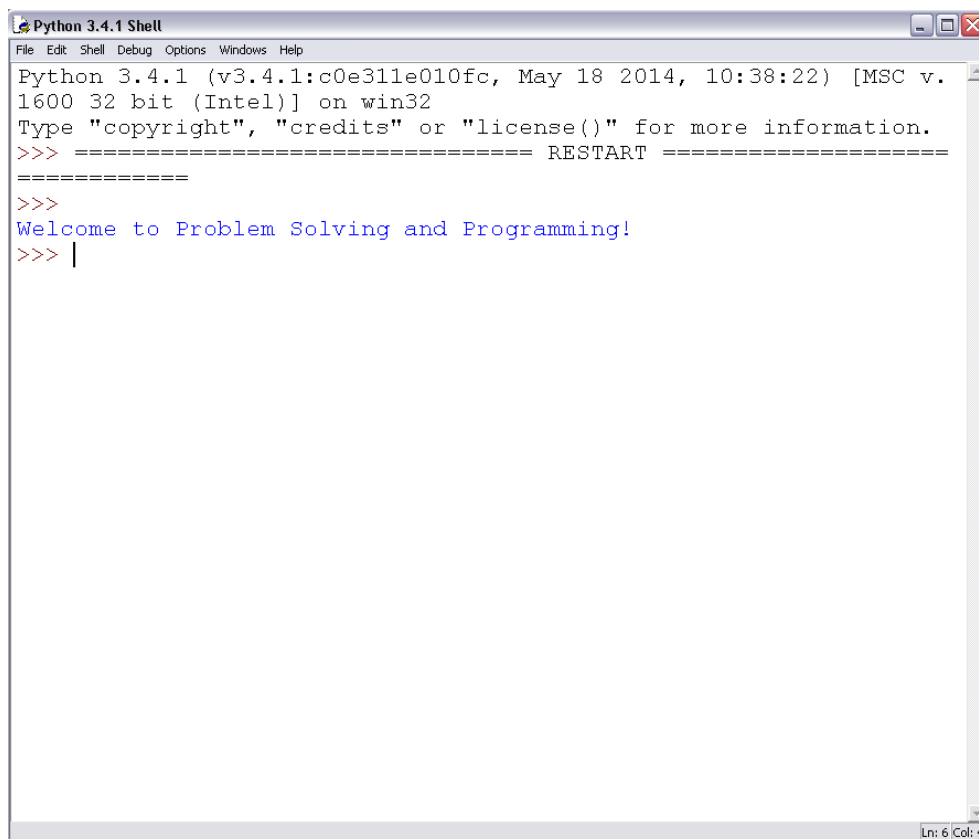
6. Enter a file name, say… `welcome.py`

   Note: Python programs must have a *.py file extension.

7. You should now see the output of the program you have just entered in the Python Shell (interpreter) window.

Checkpoint: Please make sure your supervisor sees both your program output and code.

8. **Check for errors.** Errors, if any, will appear in red text in the Python shell window or as a pop up dialog box. If you have errors, look at the editing window and compare the code very carefully against the code above. If you can't find the problem, ask your practical supervisor. When you find and fix an error (in the editing window), be sure to save and run the program to ensure it is working correctly. You may do this as many times as it takes to ensure your code is working correctly.

Let's talk about a few common errors.
A common error is to forget to close the string. i.e…

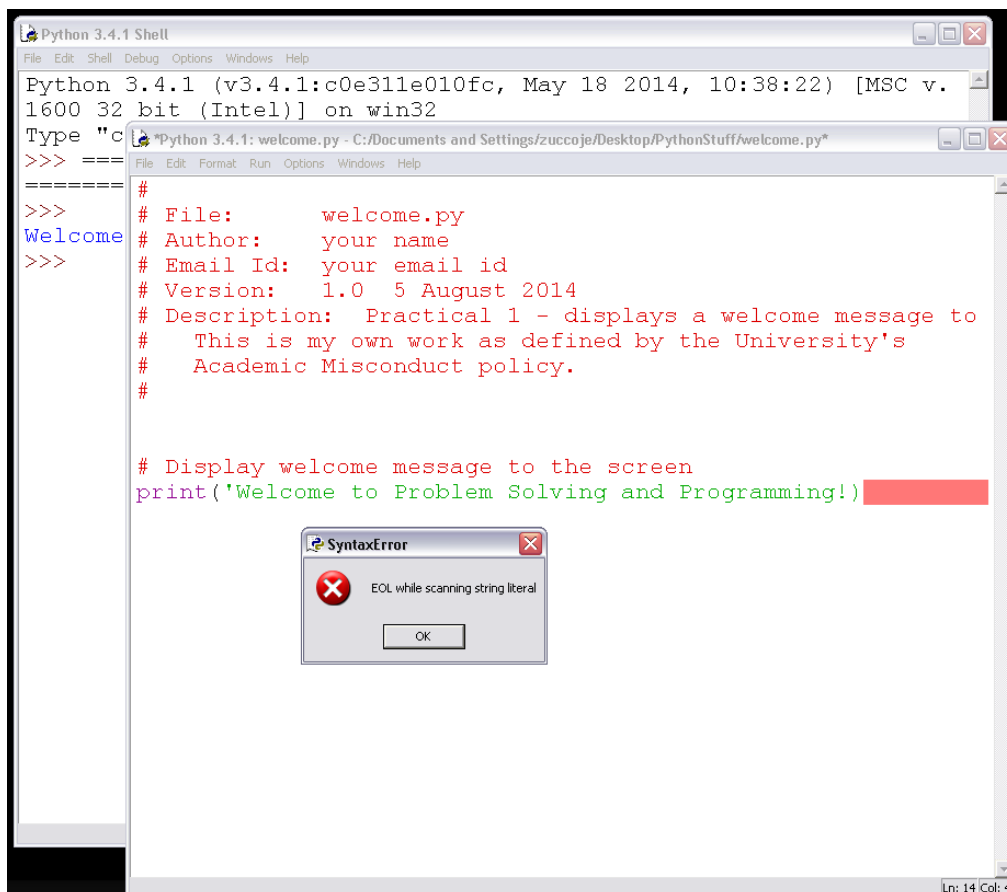   print('Welcome to Problem Solving and Programming!)

… notice ' is missing at the end of the string.

Actually, let's do that so we can get a feel for how Python reports errors.
    Remove ' from the end of the statement

        print('Welcome to Problem Solving and Programming!).

- Save and run your code.
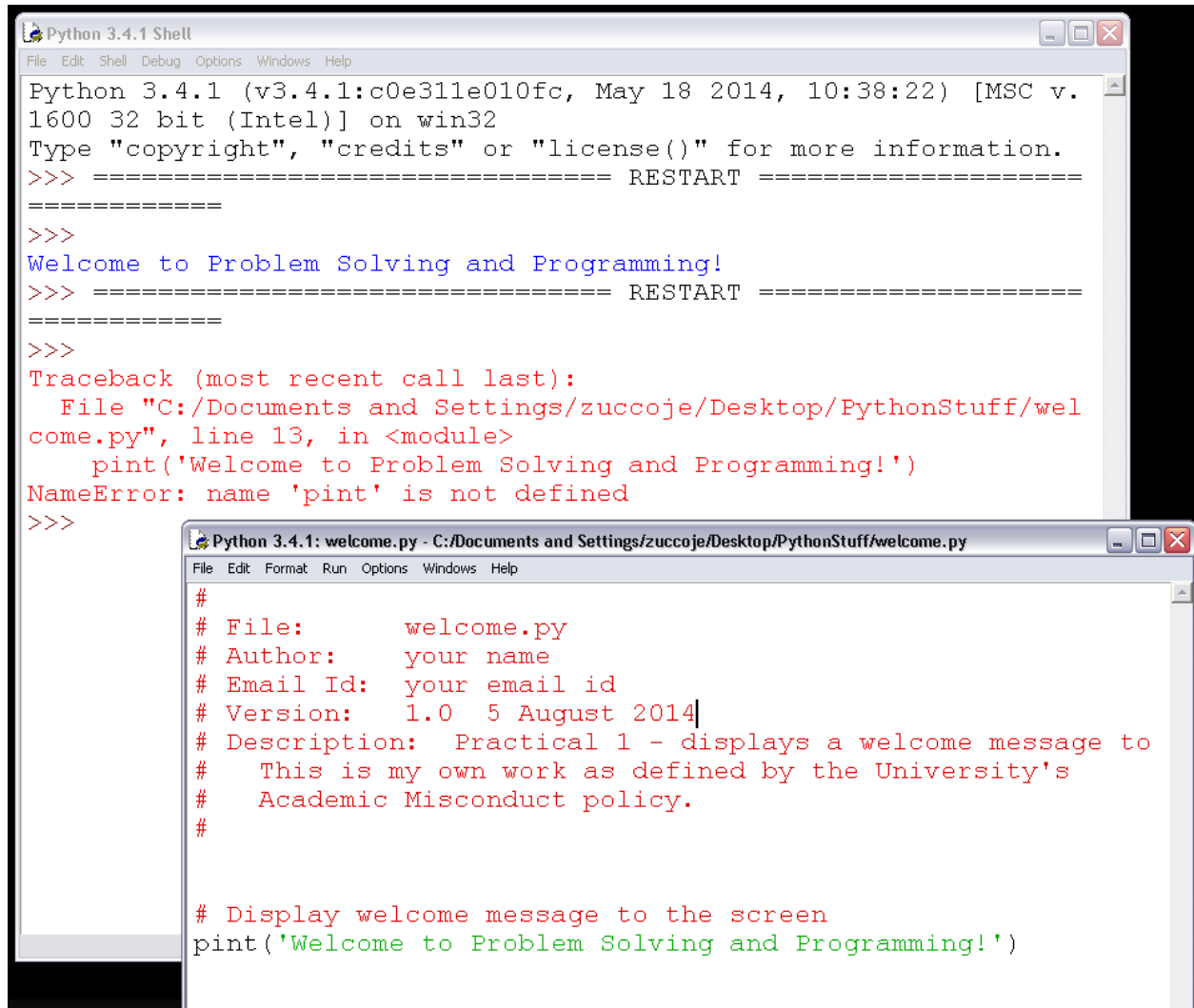- Notice the error reported.

- Now place the ' back onto the end of the string.
- Save and run your code.
- Notice the error is no longer reported and the output is displayed to the screen as expected.

Another common error is to incorrectly type commands or to misspell them.

Let's do that again:

- Remove the character r from the statement
    pint('Welcome to Problem Solving and Programming!')
- Save and run your code.
- Notice the error reported.

```
Python 3.4.1 Shell
File  Edit  Shell  Debug  Options  Windows  Help

Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.
1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ============================= RESTART =====================
============
>>>
Welcome to Problem Solving and Programming!
>>> ============================= RESTART =====================
============
>>>
Traceback (most recent call last):
  File "C:/Documents and Settings/zuccoje/Desktop/PythonStuff/wel
come.py", line 13, in <module>
    pint('Welcome to Problem Solving and Programming!')
NameError: name 'pint' is not defined
>>>
```

```
Python 3.4.1: welcome.py - C:/Documents and Settings/zuccoje/Desktop/PythonStuff/welcome.py
File  Edit  Format  Run  Options  Windows  Help

#
# File:       welcome.py
# Author:     your name
# Email Id:   your email id
# Version:    1.0  5 August 2014
# Description:  Practical 1 – displays a welcome message to
#    This is my own work as defined by the University's
#    Academic Misconduct policy.
#


# Display welcome message to the screen
pint('Welcome to Problem Solving and Programming!')
```

Try to read the error message to get a feel for how they are reported. Notice the file name, the line number, and the error message – name 'pint' is not defined.

- Now place the r back into the statement.
- Save and run your code.
- Notice the error is no longer reported and the output is displayed to the screen as expected.

9. Instead of displaying the text `'Welcome to Problem Solving and Programming!'` to the screen as you did in exercise 5, modify the code above to display the following to the screen (*hint: you will need to use more than one print statement*):

```
Author:  your name
Email Id:  your email id
This is my own work as defined by the
University's Academic Misconduct Policy.
```

Save and run the program. Look at the command window for output. If you have errors, refer to the discussion in exercise 8 (regarding errors), in order to help you identify and fix any error(s) you may have. If you can't find the problem, please ask your practical supervisor to help you.

10. Write a program that displays your name surrounded by a box made from -, + and | characters. See example output below. Hint: you will need to use print statements in order to achieve this.
    Create a new file called 'name.py' (remember: Python programs must have a *.py file extension) in order to complete this exercise (refer to exercise 3 onwards or ask your supervisor if you are having problems doing so).

```
+----------------+
|   James Bond   |
+----------------+
```

Save and run the program. Look at the command window for output. If you have errors, refer to the discussion in exercise 9 (regarding errors), in order to help you identify and fix any error(s) you may have. If you can't find the problem, please ask your practical supervisor to help you.

Checkpoint: Please make sure your supervisor sees both your program output and code.

11. What output does the following code produce? Check each of your answers by entering the code in a file and running the programs.

a)
```
number1 = 0
number2 = 7
number3 = 2
number1 = number2 + number3
number3 = 9
number1 = number3 - number2
number2 = 7 * number1 + 6
print(number1)
print(number2)
print(number3)
```

b)
```
number1 = 2
number2 = 7
number3 = 3
result = number1 * number2
print(result)
result = number3 * number2
print(result)
```

c)
```
number1 = 1
number2 = 2
number3 = 3
number1 = number1 + 2
number1 = number1 + number2
number1 = number1 + number3
print(number1)
```

d)
```
number1 = 5
number2 = 6
number3 = 7
result = 0
number3 = number2 / 3
number2 = number1 * number2
number1 = number1 + 2
print(number1, number2, number3)
```

Python Interpreter Window (Python Shell)

Okay, now that we have explored how to create, save and run a Python program, let's spend some time working in the Python Shell. Remember from last week's lecture that the Python shell is a great place to try out commands to see how they behave.

On Windows, the Python shell should look like this:
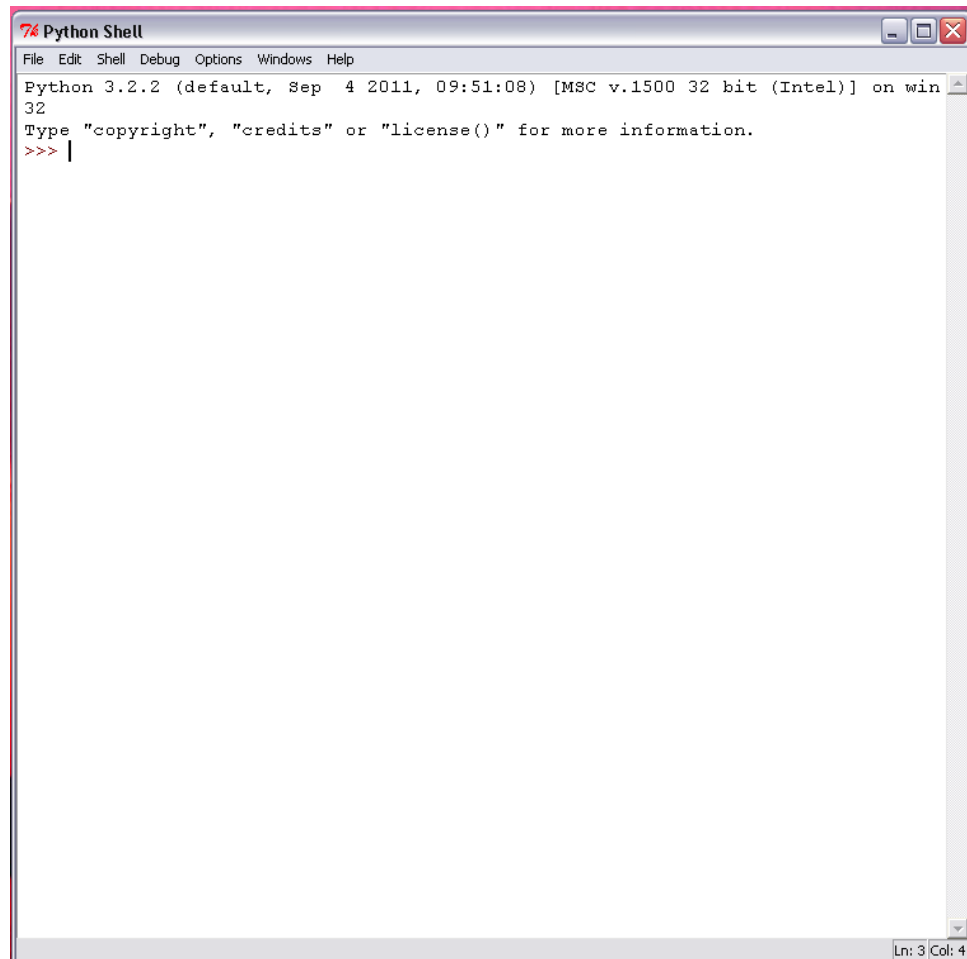
12. What do the following commands produce? Check your results by entering the expressions into the interpreter (Python Shell window).

```
5 + 2
5 * 2
10 - 2
10.75 - 5.5
1 + 3/4
(3 + 5) * 2
3 + 5 * 2
5 / 2
5 // 2
7.0 / 2
5*6*4/2
5/2*6*4
```

```
2**2

2**4

5**2*3

5**(2*3)

1 + 3 + 5/5 + 3 + 1

(1 + 3 + 5)/(5 + 3 + 1)

5 % 2

11 % 3

9 % 3
```

In interactive mode, the last printed expression is assigned to the variable _.

Type _ at the prompt and press enter to reveal the result assigned to variable _

You may also find it useful to work through section 3.1.1 of the Python tutorial (excluding the complex numbers section). Attempt all of the examples within the Python shell and try additional examples of your own, so that you are familiar using (for integers and floating point types) the operators +, -, /, //, %, **.

(Ref: 3.1. Using Python as a Calculator; 3.1.1. Numbers: http://docs.python.org/py3k/tutorial/introduction.html#numbers)
Please exclude the complex numbers section.

13. The equal sign ('=') is used to assign a value to a variable. Enter the following and notice that no result is displayed following an assignment statement.

    Enter the following:
    ```
    >>> number1 = 20
    >>> number2 = 2
    >>> number1 * number2
    40
    ```

    In Python, a variable has to be defined (assigned a value) before it can be used, or an error will occur:
    Enter the following and notice the error message:

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

14. The area of a square is the edge length squared (area = edge**2).
    Using question thirteen (above) as a guide, define variable edge length as 5, then find the area of a square.
    Display the area to the screen.

15. The area of a circle is calculated as follows: area = PI * radius**2.
    Hint: Define radius as 5, define PI as 3.14, then find the area of a circle. Display the area to the screen.

16. What is the output produced by the following three Python statements? Check your answers by entering the
    statements into the interpreter (Python Shell window) one at a time.

```
print('I\'m having fun and can\'t wait to learn more!')
print("I'm having fun and can't wait to learn more!")
print('I'm having fun and can't wait to learn more!')
```

Congratulations!   You have just created, saved and run your first Python programs! Well done! ☺

Please make sure you save and keep all of your practical and assignment work. You will need to save your work on a USB or the like. Please ask your supervisor if you are having difficulties doing so.

End of practical 1.