University of Hull
Department of Computer Science

Playing with Python

Vsn. 1.0 Rob Miles 2014

File Handling and Debugging

Practical Break 1: File Handling

In this lab we are going to persist some player data in a file and bring it back.

Using your previous program

We are going to build on the Cricket Player program that you built last week.

Before you go any further; perform the following:

1. Start the Idle environment.
2. Use "**File>New Window**" to find your program file and open it to work with.

Persisting Player data in a file

A player is made up of a name value and a score value. These will both be stored in the file as strings:

```
p = player('Fred',100)
f=open('Fred.txt',mode='w')
f.write(p.name + '\n')
f.write(str(p.score))
f.close()
```

Before you go any further; perform the following:

3. Open your player program and use the above code to write the score for fred into a file.
4. You should find the file fred.txt in the folder where you have stored the Python program. Open the text file with notepad and make sure that it has the content in it that you expect.

You can write any number of lines into a file this way. Remember that you **must** close the file when you have finished writing. Remember also that the above code will overwrite the file fred.txt when it runs, the user is not asked to confirm the action.

If you want to have all the items on separate lines you have to put the '\n' on the end of each line. This inserts the newline character If you don't do this all the print items will be concatenated and it will be much harder to

Reading Player data from a file

Just to test that the program works we can read some player data back from a file:

```
f=open('Fred.txt',mode='r')
name = f.readline()
name = name.strip()
score = int(f.readline())
f.close()
p = player(name,score)
print(p.name)
```

This code reads in the data for a player and then creates a new one from the values that were fetched from the file. Note that the line feed characters are stripped from the name so that we can print it properly. There is no need to strip the line feed from the score as the int method will just stop reading the number when it reaches a non-digit character.

Before you go any further; perform the following:

5. Use the code above to read the player back

Storing Large Numbers of Players

If you want to store lots of players you can use a loop to write out each player in turn. You could write the number of players first and then each player in turn. You need to do this so that the reading program knows when it has reached the end of the player information.

This means that you should design the format for storing the players. If you want to store structured data like this there are some Python libraries that will help. Search for "Python Pickle" and "Python json".

Practical Break 2: Starting with PyScripter

PyScripter is a great place to write Python programs it is installed on top of an existing Python installation and lets you compile, run and debug Python code.

You can start PyScripter by searching for it in the Programs on the machines. You can use it to open and debug Python program code.

Before you go any further; perform the following:

1. Open PyScripter.
2. Select **File>Open** and then navigate to your program file.
3. Run your program by pressing the green "play" triangle, by selecting **Run>Run** from the Run menu or pressing **CTRL+F9**

When you run the program you will notice that rather than asking for text input in the console window the program will instead display a dialog box pop-up to request the text. The output is printed in the text area at the base of the PyScripter program.

This behaviour only occurs when you are using PyScripter. If someone was to open your program using IDLE they would have the same user experience that you have seen since the start of the course. The text pop-up is because of the way that PyScripter works. In reality this is not a problem because you can use Python to write programs that have the more modern windowed display, in which situation the user input and output is performed using the windows that the program generates.

Finding Errors

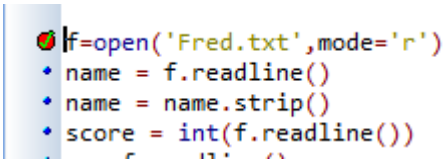
PyScripter will highlight errors in your code automatically. This makes debugging much easier.

Before you go any further; perform the following:

4. Use the editor to remove the colon from the end of the player class definition.
5. Run the program again. You should find that it stops and identifies the faulty line.

Adding a Breakpoint

You can add a breakpoint by clicking on the blue dot to the left of any statement in your program.

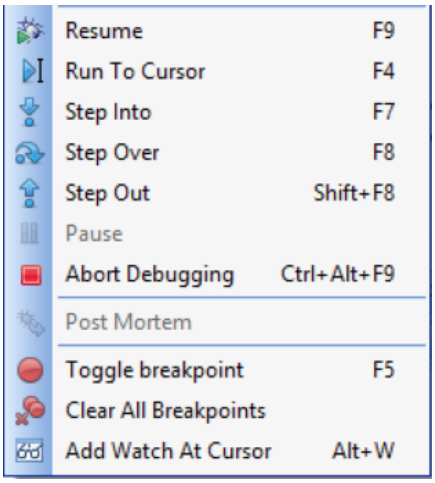


When the program is running in debug mode it will stop at this location and allow you to get control. Note that breakpoints are only hit when the program is running in "debug" mode. This is activated from the Run menu or by pressing the **F9** key.

Before you go any further; perform the following:

6. Find a statement that you are interested in and set a breakpoint there.
7. Press F9 to start the program running in Debug mode. It should stop at the breakpoint that you set. The statement should be highlighted in blue.

When the program hits the breakpoint it is paused. You can control what the program does next. All the options are on the Run menu, but there are also matching buttons on the display along with keyboard shortcuts:

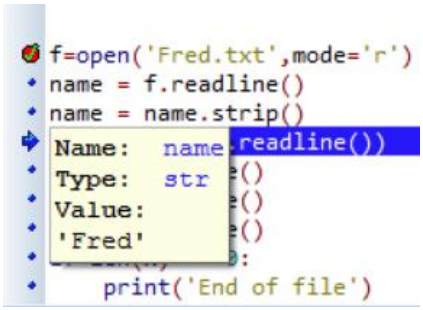


The difference between "Step Into" and "Step Over" is that Step Into will debug into the method, were as Step Over will just call the method and then move on to the next statement.

Before you go any further; perform the following:

8. Step through a few lines of your program.

As you step through the program the blue line will show you the line your program is "at", i.e. the next statement to be obeyed. You can view the content of any variable just by resting your cursor over it:



The debugging information includes the type of the data being held at that location along with the current value of that variable.

Before you go any further; perform the following:

9. Investigate some variables in your program.
10. Abort Debugging by selecting **Run>Abort Debugging**.

PyScripter contains a lot of interesting features, it can help you format your code and makes it much easier to browse for programs and work with them.

Developing the Cricket Scores Program

You can now move your development into the PyScripter environment.

- Add save and load features which allow the user to save and load score values.
- You could also add a "match report" feature which will print a text file that identifies the best and worst players
- You can create "comma separated value" (CSV) files if you print out lists of items separated by commas. These can be imported directly into Excel.

Rob Miles March. 2014