# Python – week1

## Contents

Python – week1	1
Intro:	
Getting Started with Eclipse	2
Lessons:	8
Application # 1	15
Design	15
Coding	15
What is 'ifname == "main"' for?	21
Resources:	21

### Intro:

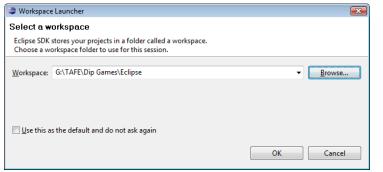
<u>Pygame</u> is a python wrapper for <u>SDL</u>, written by Pete Shinners. What this means is that, using pygame, you can write games or other multimedia applications in Python that will run unaltered on any of SDL's supported platforms (Windows, Unix, Mac, beOS and others).

If we get some time towards the end of term4 we will look at wxPython and py2exe.

### **Getting Started with Eclipse**

Eclipse unpacks to a folder ready to use. Find the exe and run it. We need to add PyDev so follow these instructions.

1. When Python first starts it will ask you for a default workspace. As TAFE computers' desktops get nuked every time there is a problem set this to your Hard Drive instead.



2. If you run into problems with workspaces, once the navigator has started just switch using this menu...

_	ava - Eclipse SDK			_		
e	Edit Navigate	Search	Project	Run	Windo	ow Help
	New		AI	t+Shift	+N ►	
	Open File					
	Close			Ctrl	+W	
	Close All		Ctr	+Shift	+W	
	Save			Ctr	l+S	
	Save As					
h	Save All		Ct	rl+Shif	t+S	
	Revert					
	Move					
	Rename				F2	
1	Refresh				F5	
	Convert Line Deli	miters To			+	
5	Print	CLARKER SEC.		Ctr	I+ P	
C	Switch Workspace	e 🦯			•	Other
8	Restart	ALC: NO			1	

 To install Pydev and Pydev Extensions using the Eclipse Update Manager, you need to use the Help > Install New Software... menu (note that in older versions, this would be the 'Find and Install' menu).

	CALIFORNIA AND A	8
ndow	Help	
-	3	Welcome
	0	Help Contents
	<b>%</b>	Search
		Dynamic Help
		Key Assist Ctrl+Shift+L
		Tips and Tricks
		Cheat Sheets
		Check for Updates
(	and the second sec	Install New Software
	CONTRACTOR OF CONTRACTOR	About Eclipse SDK
	No. of Additional Operation of Additional Operation of Additional Operation of Additional Operation of Additional Operational Operationae Operationae Operationae Operationae	and the second se

4. Type in a custom name for the plugin addition and add the link provided (http://pydev.org/updates)

Install     Available Software     Select a site or enter the location of a site.	Add Site
Work with: We or select a site	Name: Pydev and Pydev ExtensionsLocal
Find more software by working with the <u>'Available Software Sites'</u> preferences.	Location: http://pydev.org/updates Archive
type filter text	
Name Version	OK Cancel
Details	
Show only the latest versions of available software 🔲 Hide items that are already installed	
Group items by category What is <u>already installed</u> ?	
✓ Contact all update sites during install to find required software	
< Back         Next >         Einish         Cancel	

5. Now you get this screen. Only tick PyDev and it's child then click next.

😂 Install		
Available	Software	
Check the i	items that you wish to install.	
Work with:	PyDev and PyDev updates - http://pydev.org/updates	▼ <u>A</u> dd
	Find more software by working with the ' <u>Available Soft</u>	ware Sites' preferences.
type filter te	ext	
Name	Version	
	PyDev PyDev for Eclipse 1.5.7.2010050621 PyDev Mylyn Integration (optiona	
	Pydev Mylyn Integration 0.3.0	
Details		
Details		1
Show only	y the <u>l</u> atest versions of available software $\sqrt{2}$ Hide items that are already inst	alled
✓ <u>G</u> roup iter	ms by category What is <u>already installed</u> ?	
<mark> </mark>	Ill update sites during install to find required software	
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Cancel

6. You will see this window next once it's located the resources online. Click Next to continue.

😂 Install		- • •
Install Details Review the items to be installed.		
Napae	Version Id	
PyDev for Eclipse	1.5.7.20 org.python.pydev.feature.feature.group	
Size: 77 KB Details		*
()	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Cancel

7. Accept the terms and conditions and click on Next.

I accept the terms of the license agreer do not accept the terms of the license	ment energy	Sound and the second second	
(?)	< <u>B</u> ack	Next >	<u>F</u> inish

8. Next you will see the updating in progress – just let it run.

🖨 Install	
Operation in progress	
Downloading com.python.pydev	
Always run in background	
Run in Background Cancel	<u>D</u> etails >>

9. If you see this warning (probably only Vista) just click Yes

Securi	ty Warning		
<u> </u>		ware that contains unsigned content. ot be established. Do you want to con	
		OK Cancel	<u>D</u> etails >>

10. Once completed you will see this dialog. Restart Eclipse by clicking on Yes.

😂 Softv	are Updates 🔀
?	It is strongly recommended you restart Eclipse SDK for the changes to take effect. For some add-ons, it may be possible to apply the changes you have made without restarting. Would you like to restart now?
	Yes No Apply Changes

11. Configuration of Eclipse. You have to maintain in Eclipse the location of your Python installation. Open in the menu Window -> Preference and select Pydev-> Interpreter Python.

Preferences			
type filter text	Python Interpreters		← ← ⇒ < ▼
b General b Ant	Python interpreters (e.g.:	python.exe)	
⊳ Help	Name	Location	Ne <u>w</u>
⊳ Install/Update ⊳ Java			Auto Config
Plug in Development		1	Remove
Builders ⊳ Debug			Up
⊳ Editor			Dow <u>n</u>
Interactive Console Interpreter - Iron Pythou	Libraries Forced B	uiltins Predefined To Environment	String Substitution Variables
Interpreter - Jython Interpreter - Python	System PYTHONPATH	CONTRACTOR OF CONTRAC	
Logging	**************************************		New Folder
Pylint Pyunit			New Egg/Zip(s)
Scripting Pydev Task Tags			Remove
Run/Debug			
⊳ Team			
4 III >>		ſ	Restore <u>D</u> efaults <u>A</u> pply
?		0	OK Cancel

12. Press New then point the Python25 folder – it should be off the root of C:\. Now press okay

Select interpreter			<b>—</b>
Enter the name and exe	cutable of your interpreter		
	********************************		
Interpreter Name:	Python2.5		
Interpreter executable:	C:\Python25\python.exe		Browse
COLUMN BORN		-	
	Statistic Statistics and an and a statistic statistics	and the second s	
		AND DECEMBER OF	
		ОК	Cancel

13. You will see this dialog open as it processes the files.

Progress	Information			
1	Getting libs			
				Cancel

14. Finally you will see this screen.

Selection Needed			
Select the folders to be added to the SYSTEM pythonpath!			
IMPORTANT: The folders for your PROJECTS should NOT be added here, but in your project configuration.			
Check:http://pydev.org/manual_101_interpreter.html for more details.			
<ul> <li>✓ ▲ C:\Python25</li> <li>✓ ▲ C:\Python25\ULLs</li> <li>✓ ▲ C:\Python25\lib\</li> <li>✓ ▲ C:\Python25\lib\lib-tk</li> <li>✓ ▲ C:\Python25\lib\plat-win</li> <li>✓ ▲ C:\Python25\lib\site-packages</li> <li>▲ C:\Windows\system32\python25.zip</li> <li>▲ D:\eclipse-SDK-3.5.2-win32\eclipse\plugins\org.python.pydev_1.5.7.2010050621\PySrc</li> </ul>			
Select All Deselect All			
OK Cancel			

- 15. Don't change anything just click okay.
- 16. Now you will see the original screen look like this. You are ready to start programming in Python!

Preferences					- • <b>×</b>
type filter text	Python Interpreters				$ \diamondsuit \bullet \bullet \diamondsuit \bullet \bullet \bullet$
General Ant Help Install/Update Java Plug-in Development Pydev Builders Debug Editor Interactive Console Interpreter - Iron Python Interpreter - Jython Interpreter - Python Logging Pylint Pyunit Scripting Pydev Task Tags Run/Debug Team	System PYTHONPATH  System PYTHONPATH  System libs  C\Python25\ C\P	Locat C:\Py tins Predefined DLLs lib	thon25\python.exe	<ul> <li>String Substituti</li> </ul>	New Folder New Folder New Egg/Zip(s) Remove
۰				Restore <u>D</u> efaul	lts <u>A</u> pply
?				ОК	Cancel

### Lessons:

Your first Python program.

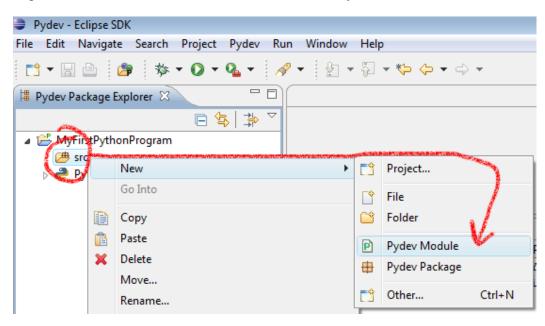
- 1. Open Eclipse
- 2. File->New Project -> PyDev->PyDev Project

😂 New Project	
Select a wizard	
<u>W</u> izards:	
type filter text	
<ul> <li>Plug-in Project</li> <li>General</li> <li>CVS</li> <li>Java</li> <li>Plug-in Development</li> <li>Pydev</li> <li>Pydev Django Project</li> <li>Pydev Google App Engine Project</li> <li>Pydev Project</li> </ul>	Le la
(?) < <u>Back</u> Next > <u>Finish</u>	Cancel

3. Click Next then change interpreter to Python25 and Grammer to 2.5, and give it a name at the top. Click Finish

•	
Pydev Project	<b></b>
🛞 Project already exists	
Project name: MyFirstPythonProgram	
Project contents:	
Use <u>d</u> efault	
Directory C:\Users\Andy\workspace\MyFirstPythonProgram	Browse
Project type <ul> <li>Python</li> <li>Jython</li> <li>Iron Python</li> </ul> Grammar Version             2.5           Interpreter           Python25           Click here to configure an interpreter not listed.           ✓           Create default 'src' folder and add it to the pythonpath?	•
(?) < <u>Back</u> <u>Next</u> > <u>Finish</u>	Cancel

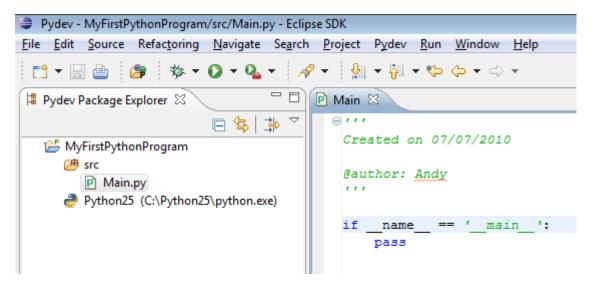
4. Right click on the "src" folder and click on New->Python Module



5. Call the file Main (no Package at this stage), select Template as Module:Main and click Finish

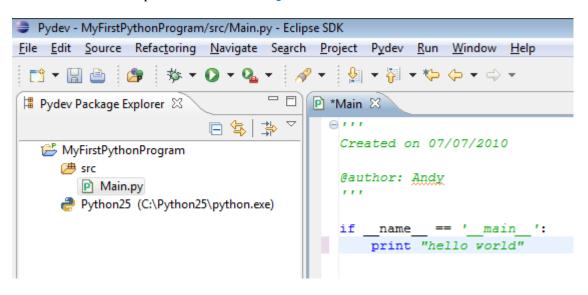
•		- • •
Create a new	Python module	ę
Source Folder	/MyFirstPythonProgram/src	Browse
Package		Browse
Name 🤇	Main	
Template	<empty> Module: Class Module: Main Module: Unittest Module: Unittest with setUp and tearDown</empty>	<u>Config</u>
?	Einish	Cancel

6. Now you'll see this. This is the template code to run as a module.

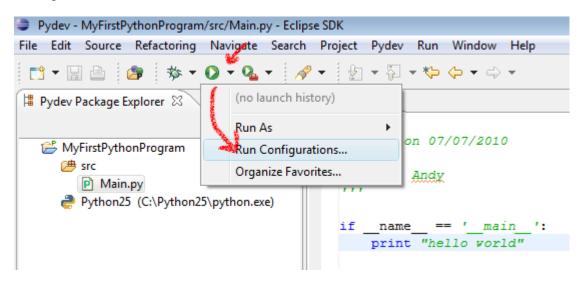


- 7. What does it mean? Let's take a look.
- 8. The ''' characters at the top is the standard DOCSTRING notation and can be considered as commenting. Comment as much as you can. You always end a comment block with a terminating ''' as seen after @author: Andy

- 9. The next line is curious if <u>\_\_\_\_\_\_name\_\_\_</u> == `\_\_\_main\_\_\_': then pass. This is a trick for Python which allows you to have this .py file exist as a module or a standalone file. See appendix for more information.
- 10. The *pass* just means that do nothing and move on. But we want it to do something.
- 11. Introducing the *print* statement! Print mainly prints characters to the screen. So type in the follow. Erase *pass* with the words *print* "*hello world*".

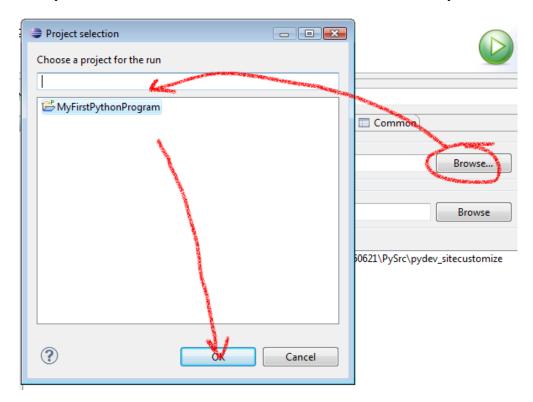


12. Now you need to run it. But before that you need to create a run configuration. Up the top you'll see a green play button – a big one – there's a down arrow next to it – click on that to bring up the menu below. Click on that then click on Run Configurations.

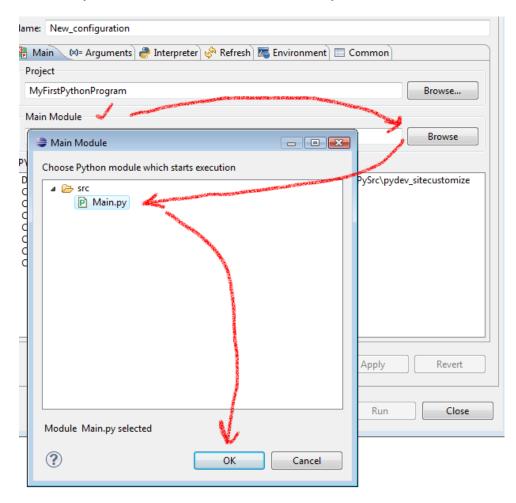


13. You'll see this screen next, double click on Python Run to create a New\_configuration.

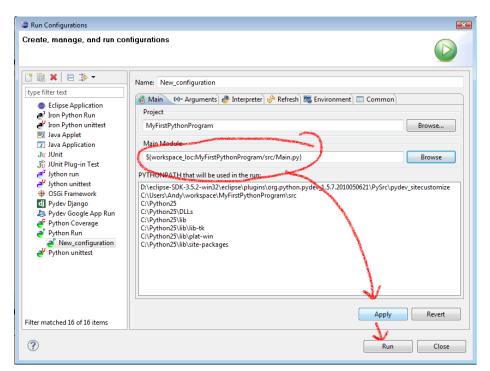
14. Next you need to browse to the main source file – which we already have.



15. Now click on the Main Module browse button then expand src and click on Main.py. As this is your main module. Then click on Okay.



16. Next click on Apply, then Run.



17. In the console you will the follow "hello world" – well done! You have created your own hello world program in Python.

i 🖂 i 🖃 i 🐜 i 🦄 i 🦝 i 🐼	
📕 Pydev Package Explorer 🛛 📃 🗌	P Main 🛛
<ul> <li>Part yack rackage explorer &amp;</li> <li>Image: Application of the second seco</li></ul>	<pre>Main &amp; Main &amp; Main &amp; Created on 07/07/2010 Cauthor: Andy ''' ifname == 'main':     print "hello vorld"      f     Problems  Console &amp; <terminated> C:\Users\Andy\workspace\MyFirstPythonProgram\src\Main.py hello world</terminated></pre>

18. So what's this hello world stuff anyway? Hello World is a standard practice for programmers. Basically it's the icing on the cake when you start a new project, using new software and want to make sure you know how to compile your program and get the results on the screen. It could be elaborated to "Finally I got through all that technical guff and got something on screen, so HELLO WORLD! – Phew".

## Application # 1.

So we are doing this unit called Automate Processes. What does this mean exactly? Well computers are great at it. You usually apply it to something that would take a human a long time to do, so make a computer do it. You are probably thinking I'm going to get you to write some ugly processing thing like a staff list and mailing list – let's look at something more practical and relevant.

You may have an iPod / iPhone and it's got this great feature called Shuffle. But how does it work exactly? Well let's write our own using Python.

### Design.

Let's identify the core functionality of the Shuffle process.

- 1) Takes a list of songs
- 2) Randomises the play order.
- 3) Keeps track of that play list so you can skip backward and forward
- 4) Never plays the same song twice in a shuffled playlist.

So how do we write this in Python? Let's do one step at time and learn a bit about programming along the way.

Python has a great system of storage called a list. A list is an array of items, and in this case we want to store the name of each song.

### Coding.

In your Main.py function go to the line that says print "hello world" and replace it with

```
print "My iPod Shuffle Simulator"...
```

and after than type the following...

```
shuffleList()
```

```
showList()
```

So your code should look like this ... (Don't worry about the red X's - the IDE is just saying these function don't exist.)

```
if __name__ == '__main__':
    print "My iPod Shuffle simulator"
    shuffleList()
    showList()
```

Okay so let's write the function headers... remember we can use *pass* so that the function doesn't do anything? So that's what we'll do.

A function is defined first before it can be called. We use the keyword *def*, the name of the function, an empty parameter list and then ending with a *colon* (:). Be sure to TAB in on the next line when adding the *pass* keyword. Should look like this...



You will also notice on the right hand side that Outline contains the new functions.

We need to create the variable myItunesLibrary at the top of the program so all of the code can access it. Let's add an empty list, then fill it with some songs using the append function available to lists – it's on the network so just copy and paste (playlist.txt)

```
🕑 Main 🖂
    ....
     Created on 07/07/2010
F.
     @author: Andy
     myItunesLibrary = []
     myItunesLibrary.append("Love The Way You Lie - Eminem - Recovery")
     myItunesLibrary.append("Dynamite - Taio Cruz - Rokstarr")
     myItunesLibrary.append("California Gurls - Katy Perry - California Gurls(single)")
     myItunesLibrary.append("I Like It - Enrique Inglesias - I Like It(single)")
     myItunesLibrary.append("Cooler Than Me - Mike Posner - Cooler Than Me(single)")
     myItunesLibrary.append("Ridin' Solo - Jason Derulo - Jason Derulo")
     myItunesLibrary.append("Airplanes - B.o.B - B.o.B(single)")
     myItunesLibrary.append("Jars of Hearts - Christini Perri - Jars of Hearts(single)")
     myItunesLibrary.append("Billionaire - Travie McCoy - Billionaire(single)")
     myItunesLibrary.append("OMG - Usher - Raymond V Raymond")
```

Above that let's add another list to be the actual playlist called myPlaylist.

```
myPlayList = []
myItunesLibrary = []
myItunesLibrary.append("Love The Way You Lie - Eminem - Recovery")
```

Okay so now for the hard part – the shuffling. Shuffling is done by randomly picking a song from a temporary library that originally is a copy of the iTunes library, which gets smaller and smaller as we remove each randomly selected song. We do this because we don't want the same song repeated in the play list.

So let's break it down into smaller steps or pseudocode.

- 1) Make a copy of the library.
- 2) Continue until the temp list is empty
- 3) Randomly pick a song from the temp list.
- 4) Add that song to the playlist.
- 5) Remove the picked song from the temp list

Now let's code it in.

Firstly we need to access a randomise function called choice. This is sitting in a module called random. So firstly we code the part where we import that functionality.

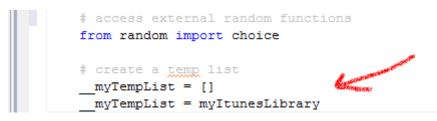
Go to the shuffleList () function and type this...

```
edef shuffleList():
    Shuffles the list from a temp buffer
    '''
    # access external random functions
    from random import choice
```

Also notice the use of "' comment "' and the # hash symbol. These are two ways of commenting in Python. The convention is to the use the DOCSTRING type on commented at the beginning of a function to demonstrate it functionality. The second method of using # is inline commenting. Please comment as much as possible because of the abbreviated nature of Python it's often difficult to understand what you wrote weeks after writing some code and trying to work out what it's doing.

Next let's begin with step 1 from our functional design.

1) Make a copy of the library -



Notice we use two underscores. This is Python's way of creating a local variable.

If you check the Outline it won't show it. This is to do with scope. Scope determines where a variable is valid and useable.

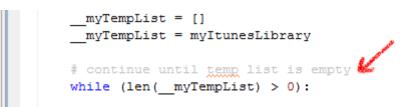
🗄 Ou	tline 🛛
0	myPlayList
•	myItunesLibrary
0	shuffleList
0	showList

In this case we only want to use it for the one function and once we pop out of the function the variable goes out of scope and the memory is returned to the system.

Next we need loop through the list.

2) Continue until the temp list is empty

We do this because of the way loops work. We need to set the condition of the loop first.

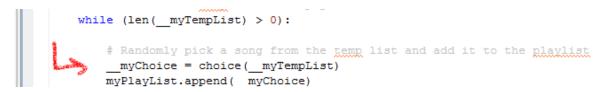


Now that we have added while  $(len (\_myTempList) > 0)$ : line, everything after that needs to be indented to make sure it only runs during this condition.

Next step...

- 3) Randomly pick a song from the temp list.
- 4) Add that song to the playlist.

Why are we doing two steps in one? Well Python is highly optimised and allows to do stuff like this – this is why Python rocks. (notice we have indented the code so it only runs while the \_\_\_myTempList still has songs to choose from.



Finally we want to avoid picking the song again so we remove it from the temp list.

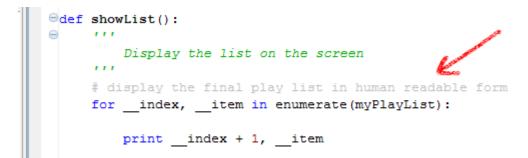
5) Remove the picked song from the temp list.

This is easy because Python lists have a function just for this – remove.

```
# remove choice from temp list
__myTempList.remove(__myChoice)
```

Okay - shuffleList() function done.

Last thing is to display this list. So let's add to the  ${\tt showList}$  () function.



We use the enumerate function because we need to have the place of the item in the list as well as the item itself. Also as is true for most lists and array in computer programming languages, they are 0 based, but humans like lists to start with 1, so we add 1 to the index variable.

Also notice we are using local variables because we aren't interested in them after the function pops out.

Okay run it – what do we see? Keep running it and every time you get a new playlist, that never repeats a song.

```
Problems Console & M= Variables
<terminated> C:\Users\Andy\workspace\MyFirstPythonProgram\src\Main.py
My iPod Shuffle simulator
1 OMG - Usher - Raymond V Raymond
2 Dynamite - Taio Cruz - Rokstarr
3 California Gurls - Katy Perry - California Gurls(single)
4 Airplanes - B.o.B - B.o.B(single)
5 Ridin' Solo - Jason Derulo - Jason Derulo
6 I Like It - Enrique Inglesias - I Like It(single)
7 Love The Way You Lie - Eminem - Recovery
8 Cooler Than Me - Mike Posner - Cooler Than Me(single)
9 Billionaire - Travie McCoy - Billionaire(single)
10 Jars of Hearts - Christini Perri - Jars of Hearts(single)
```

Because the play list is now stored we could go back forth using index or go back by remember the previous song using a temp variable if we wanted to.

Here's one way you could do it.

```
# skip forward 2 then back one
print "******** NOW PLAYING *********"
__index = 0
# get next song
__prev = __index
__index += 1
print myPlayList[__index]
# get next song
__prev = __index
__index += 1
print myPlayList[__index]
# get prev song
__index = __prev
print myPlayList[_ index]
```

Here's the new output, notice the skip forward 2 then back one at the end.



All done? Let's revisit what we did to start with by reading the appendix.

APPENDIX

### What is 'if \_\_name\_\_ == "\_\_main\_\_"' for?

The if \_\_name\_\_ == "\_\_main\_\_": ... trick exists in Python so that our Python files can act as either reusable modules, or as standalone programs. As a toy example, let's say that we have two files:

```
$ cat mymath.py
def square(x):
    return x * x

if __name__ == '__main__':
    print "test: square(42) ==", square(42)

$ cat mygame.py
import mymath
print "this is mygame."
print "this is mygame."
print mymath.square(17)
```

In this example, we've written mymath.py to be both used as a utility module, as well as a standalone program. We can run mymath standalone by doing this:

```
$ python mymath.py
test: square(42) == 1764
```

But we can also use mymath.py as a module; let's see what happens when we run mygame.py:

```
$ python mygame.py
this is mygame.
289
```

Notice that here we don't see the 'test' line that mymath.py had near the bottom of its code. That's because, in this context, mymath is not the main program. That's what the if name == " main ": ... trick is used for.

#### **Resources:**

1) Python 2.5 ( http://www.python.org/download/releases/2.5.5/ )

2) PyGame for Python 2.5 (http://www.pygame.org/download.shtml)

3) Eclipse (<u>http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-</u> 3.5.2-201002111343/eclipse-SDK-3.5.2-win32.zip)

4) PyDev (<u>http://sourceforge.net/projects/pydev/</u>) (just for my reference - don't download) Tutorials: http://www.vogella.de/articles/Python/article.html#configuration

GLU-IT ( <u>http://www.downloadsofts.com/download/Graphic-Apps/Editors/Glue-Sprites-download-details.html</u>) AUDACITY ( <u>http://audacity.sourceforge.net/download/</u>)

Tortoise SVN (<u>http://downloads.sourceforge.net/tortoisesvn/TortoiseSVN-1.6.3.16613-win32-svn-1.6.3.msi?download</u>)

Python 2.0 Quick Reference