**ICTPRG301**

**Lesson 6 Notes**

**Library Programming**

All programing languages allow you to build libraries of commands. Most languages come with many already built and ready to use, which we will look at shortly. To use these commands from a library, the first step is to import them. (Python calls these libraries modules). In Python to import a module you use the key word **import,** there are a number of ways to do this (and it might be worthwhile to look up the import command)

Once they have been imported you use the command using the following:

Library\_name.command

For example:

import sys #this imports the library called sys

print (sys.path) #this will print a list of the paths which Python used to find libraries

sys.path.append(‘your\_path\_to\_your files’) #this will add your folder to the list and allow your python files to become libraries

You can use your own existing function as a library, which means that the functions you write can be re-used in other programs. In the following exercise we will use the countdown function you wrote in Lesson 3 as a library.

Let’s now go through this step by step

1. Find your saved file with the countdown function (from Lesson 3). We will use this file as our first library. It would be good if that file had a simple name, some complex names cannot be imported (I was not able to import My-Little-lib.py, Python did not like the dash).
2. Create a new file using the template. Import statements can be inside or outside functions but it is often best that they are placed inside functions so they become part of the flow of the program.
3. In the main function the first command is:

import sys

This brings in the library called sys. We need this library for our imports to work from folders other than the current folder or the standard libraries. The problem at the moment is that Python has only certain folders that it looks in for libraries to import. Almost certainly the place where you saved the procedures was not part of that search path. We need to add the path of the folder to the Python import path. The library sys has a command path which adds a path to this list. The command is:

sys.path.append(‘*path’*)

Note the path must be in quotes as it is a String. The backslashes can be exactly as they are written normally in console (the sys library takes care of this).

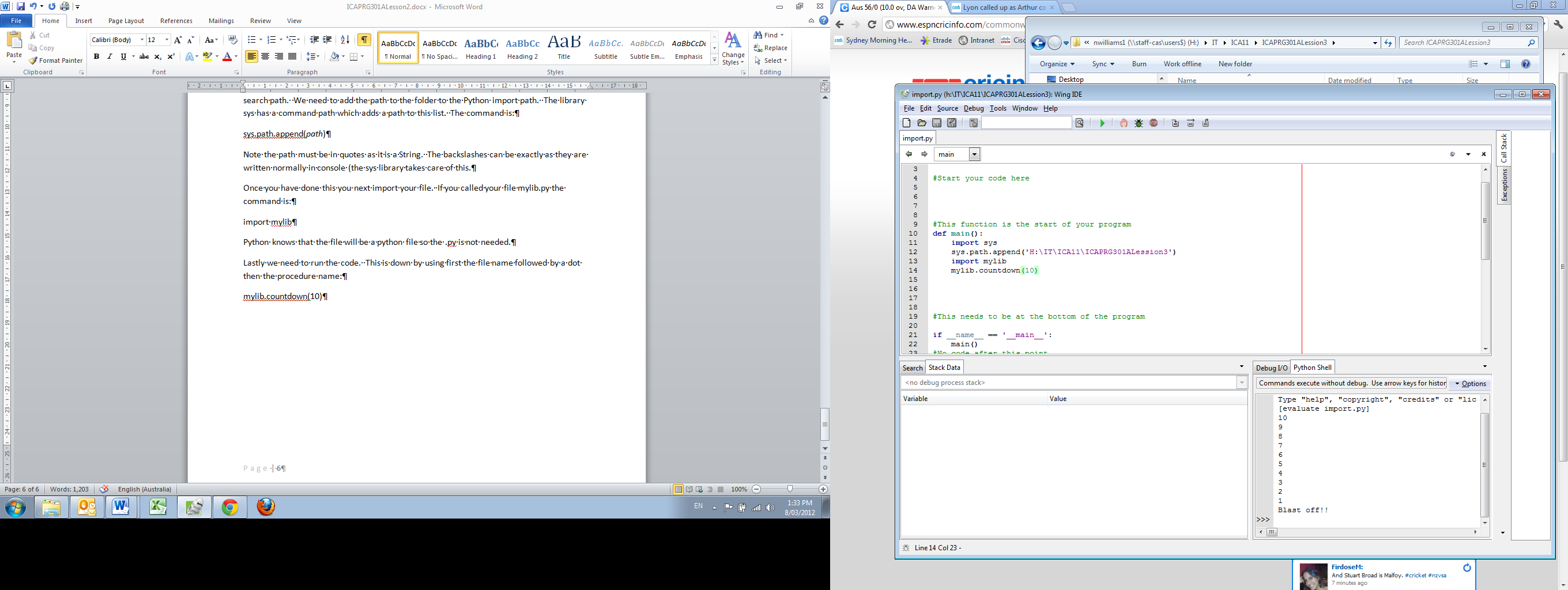
1. Once you have done this you next import your file. If you called your file mylib.py the command is:

import mylib

Python knows that the file will be a python file so the .py is not needed.

1. Lastly we need to run the code. This is down by using first the file name followed by a dot then the procedure name:

mylib.countdown(10)



**Technical note**. Import looks in two places for the file to import, one is the Python path (which we just changed in the exercise) and the other is the current folder.

The Python language comes with built in keywords and commands (see Lesson 1) and also with a standard set of Libraries. These libraries are modules which have been written to perform common tasks. We have already just used one of the libraries (sys). The full list of the standard libraries is found here:

<http://docs.python.org/release/3.1.3/library/index.html>

Let’s look at some of the more useful libraries:

**datetime** is a library we can use to show the current time and also to perform calculations on time  
**time** is a library for work with and calculating time, it includes a great method sleep  
**random** is a library which can be used to generate a random number, often used in games  
**os.path** is a library which is used to access the filing system, especially across operating systems  
**os** and **sys** are libraries to work with the operating system  
**webbrowse**r is a library to work with web browsers  
**urllib.request** is a library to work with URLs directly  
**tkinter** is a library to place a graphic interface on the screen  
  
However in addition to this there are literally thousands of other libraries which don’t come included in Python but which may be useful. One of these (called EasyGUI) uses the standard library tkinter and builds easy to use graphic interfaces, especially message boxes, input boxes and text boxes. Your version of Python does not have this installed so we will go through this process to install it.

The website for EasyGUI is here:

<http://easygui.sourceforge.net/>

Navigate to the downloads section, the version that works with our version of Python (3) is version\_0.96. Click this link and then download the zip file into a folder of your choice, however make sure that you can navigate there through the console. (Often the best place to download libraries is the Python folder). The next stage is to unzip the folder. This creates folder called easygui\_0.96 which has two files in it (easygui.py and setup.py). According to the instructions on the website to install this you need to enter the command in the console. So open the console and navigate to the folder where setup.py is saved. Then the command is:

python setup.py install

This works if your system recognises the command python, if not you need to put the full path name instead. On my system that was

C:\Python32\python.exe setup.py install

Now EasyGUI should be installed on your computer and be ready to use. To test this fire up Wing and type in the following program (you will need the file TenGreenBottles.txt in the same folder:

import easygui as eg  
answer = eg.enterbox('Enter data','another title')  
eg.msgbox ("Reply was:" + answer, "title")  
def readFile(filename):  
 file = open (filename, 'r')  
 text = file.read()  
 file.close()  
 return text  
text = readFile("TenGreenBottles.txt")  
eg.textbox("here are the words to the song","Ten Green Bottles",text)

To explain how this works

**import easygui as eg** You have seen the command import already, by adding **as eg** you are adding an alias so the commands need just eg rather than easygui all the time., thus easygui.msgbox is just eg.msgbox.

**answer = eg.enterbox('Enter data','another title')** the variable answer is set to the output of the enterbox, which is part of EasyGUI. The enterbox can take a number of inputs, we are using just the first two, ‘Enter data’ is going inside the enterbox as an instruction and ‘another title’ goes in the title of the box. answer will always be a String.

**eg.msgbox ("Reply was:" + answer, "title")** msgbox creates a message box, “Reply was: “ + answer puts this text in the main part of the message box and the “title” was put in the title.

**eg.textbox("here are the words to the song","Ten Green Bottles",text)** creates a text box to display the contents of the text file (this only works if the textfile TenGreenBottles.txt is in the same folder as our python file). You should be able to work out what the inputs do for this command.

Change some of the parameters to make sure that you know how all this works. To really understand how EasyGui works it might be worth loading the file easygui.py into Wing. This will show you the full source code of all the library functions which does include instruction on their use. You might also find other interest functions not covered here.

**Activity 1:**

In lesson one we wrote a small calculator, find that program and modify it so that all the information comes using enterbox and the answer is shown using msgbox

Type in the following program which is an example of another files that uses library commands:

import webbrowser  
import urllib.request  
import os

def writeFile(filename,text):  
 file = open (filename, 'w')  
 file.write(text)  
 file.close()

webbrowser.open("http://smh.com.au")  
text = str(urllib.request.urlopen("http://smh.com.au").read())  
writeFile('smh.txt',text)  
cmd = "notepad.exe smh.txt"  
os.system(cmd)

Run this program to see what it does.

Note:

All the imports are together first. There is a friendly argument in the Python world about where to put the import statements, my preference is together in the module or function where they are used. They must be put before they are used. Here I am not using an alias.

**webbrowser.open("http://smh.com.au")** this will open your system default browser to the URL specified. Note I am using the shortened version of the url since the browser can understand this. How this actually occurs is beyond Python and is controlled by the operating system and the browser.

**text = str(urllib.request.urlopen("http://smh.com.au").read())** this is a long command so let’s break it down. A variable text is being assigned a value. urllib.request has a function in it called urlopen which will open a website url. The urlopen command is then further asked by the read() command to read from the website. The whole command is wrapped in str() which is a built in function to ensure that the output is a String.

***Important!*** *This is not the best way to work with a URL, if there is a problem with the internet your program will crash. We will look at a better way soon.*

**os.system(cmd**) this runs the command that was created in the previous line. Anything which will run in the console will work here.

This gives you just a taste of what is in the libraries. There is too much for this or any course to cover, just dip into what you think you might need. If you think of something that you want to do in Python, chances are someone has already thought of it, and developed a library to make it easy.

**Sample program.**

I have included a sample program, SimpleWebCrawler.py. This program is a basic web crawler which reads in the html of a website, examines the text looking for the html links (which are the anchor tag in html) and will open all the pages that are linked on the page in a browser.(How your operating system reacts to this may differ from computer to computer, if this were a fully developed program this sort of testing would need to occur.) This program can also be used as a base for developing a web crawler to do anything else that you can think of with web pages. There are a number of features to note in this program.

1. All the code is contained in functions, this means the file can be a program and also a library.
2. All the functions have a comment before their functions, if the file is imported as a library these comments can be seen using the help command. I have used both methods of doing comments, the # and the doc\_string. The comments at the very top of the file also become part of the help section of the file.
3. The functions are short and named appropriately indicated what the function do. This enables them to be building blocks to something larger. I use camelCase for naming function but most Python programs use the underscore, ie getURL (camelCase) or get\_URL (underscore). This just shows my programming roots in the language Java, either way is acceptable but be consistent.
4. The code which accessed the internet is wrapped in a try: exception: block, this will be explained below
5. The code in the getAllLinks has a while loop which uses while True: This is an endless loop and to get out you must use the command break.
6. The import statements are in the function where they are used. This is essential in a library since each function must perform correctly independently of everything else.

You could have written this using the programming knowledge you have now.

*Sometimes there are problems with copying this file into Wing and running it. The issue here is the use of whitespace. If you have this problem you will need to reformat every function*

**try: except:**

When Python comes across a situation it can’t handle it gives an error message and stops working. Mostly these are errors in your code and you want this to happen. However sometime they are errors beyond your control and in this situation you want to handle the error yourself and keep going. Opening a web page is such a situation, as is getting information from the user. Everything in the try section could generate an error. If it works properly then the program keeps going. However if it generates an error it is up to you to fix that in the except section. You need to make the try section as small as possible, usually one line of code since this will only execute up to the line with the error, any further lines will be ignored.

In the sample program this is used to get a string which contains the code of the website.

def getURL(page):  
 '''This function provides the text for a web page '''   
 import urllib.request  
 try:  
 return str(urllib.request.urlopen(page).read())  
 except:  
 return ""

This uses the try: except: In this case if the try block succeeds it will return the source code of the website. If it fails the function will return an empty string.

**Assignment 3**

**Magic Number Game**

Develop a small working version of the Magic Number game. All displays to be in windows, not the console. You may use EasyGUI to create the display or something similar.

In the Magic Number game the computer thinks of a number between 1 and 100. The user then guesses a number and the computer responds with the correct response. The computer’s available responses are:

The number is higher

The number is lower

You guessed correctly

This continues until the user gets the right answer. The number of guesses is then displayed. Your program must be able to handle the user typing in characters not numbers. How you handle this is up to you but it should not crash your program.

Hints:

* There is a library called random to generate the number. It is pretty easy to use. Google it to see how it works
* Use Enter boxes and Message boxes from Easy GUI for your displays
* You should use try except when getting information from the user who might not type in what you expect. Your program should just accept this and move on. A good way to do this would be to create a function called safeEnterBox which will return the correct number in the right range or else a standard response like 0.
* Try to write this as a well-structured program.
* You might want to use the endless loop structure with a break if the user guesses the correct number.