**ICTPRG301**

**Lesson 5 Notes**

**File Programming**

**List comprehensions**

Type the following into the shell

a = [x\*\*2 for x in range(1,20) if x%2 ==1]

Now print a. This prints out a list of the first 10 odd square numbers. The code that creates the list is known as list comprehension. Here’s how it works.

This tests the original x and excludes some which fail the test.

The is the value that will be included in the list x\*\*2 is x2

[x\*\*2 for x in range(1,20) if x%2 ==1]

This is the where the x will get its value

Square brackets at the end mean that what is inside is a list

This is exactly the same as the following function:

def squares():
 list = []
 for x in range(1,20):
 if x%2 ==1:
 y = x\*\*2
 list.append(y)
 return list

it’s just easier to put it all into one line, rather than create a function.

List comprehensions can only contain for and if statements and must have at least one for statement. However they can have as many qualifying if (and for) statements and can be quite complex. They will work for any value which can go into a list, so works for Strings just as well as numbers.

Try the following, see if you can work out what the list will be before you try it:

a = [x for x in range(10)]
b = [y for y in range (1,10,2)]
c = [x for x in range(20) if x in b]
c = [x for x in range(20) if x not in a]

s = "Monty Python have made 5 movies".split()
t = [x for x in s if x.count('a')>0]
t = [x for x in s if x[0] == 'm']
t = [x.upper() for x in s if x[0].isupper()]

**Activity 1:**

Use list comprehension to create a list of the following:

The first 10 squared numbers (1, 4, 9, 16, 25, 36, 49, 64, 81, 100)

The first 8 binary numbers (2, 4, 8, 16, 32, 64, 128, 256)

The first 10 even numbers (2, 4, 6, 8, 10, 12, 14, 16, 18, 20)

Counting numbers to 100 counting by 9 (9, 18, 27, 36, 45, 54, 63, 72, 81, 90, 99)

The first 10 even squared numbers (4, 16, 36, 64, 144, 196, 256, 324, 400)

This course does not cover all the data types and structures that are available in Python. Here are some others which you may want to explore yourself:

**Tuples:** immutable group of values in round brackets eg (‘Three’, 3, ‘III”). Often used as part of larger data structures eg a list of tuples is often used as constants in a program.

**Sets:** mutable list without any duplicates in curly brackets eg {1,2,3}

**Dictionaries:** mutable data structure of groups of key-value pairs in curly brackets eg { ‘a’:4, ‘b’:25, ‘c’:2}. These are used when large data structures are needed which are searched since the searching of dictionaries by key is much faster than ordinary lists.

**Classes:** mutable custom made structures which are limited only by the programmer’s imagination.

Python Programs

So far we haven’t really worried about how to construct a program in Python, we have been concerned with commands. In the course so far your programs have consisted of one file, usually with one function and some commands to run the function. In programming language we rarely allow commands to sit by themselves, we almost always place them inside a function. In this lesson and in your next assignment you will need to use more than one function, though you can still use just one file. In the next lesson we will show you how to create and use multiple files within the one program.

In the future you may want to build more complex programs which have a number of files each with a number of functions so it is important to have an understanding of program structure. Once you start building programs this way you need a construct that Python knows is the start. For Python that construct is:

if \_\_name\_\_ == '\_\_main\_\_':
 main()

which is placed last in the file. You then create a function called main() which starts the program. (If you place this at the top of the file Python will not yet have read the main function and won’t know about it. Then when you try to use it Python will throw an error. The error will say main not defined.)

A **template** has been created for you to use to create your programs. Load this template into Wing. The program has a simple line in the main function. Run this to make sure it works. When using the template your code always goes at the top of the file. The main function is where you control how you want your functions to run.

Once you start writing computer programs as a series of functions you will need a way to remember what they do. If you have written good code you should be able to read the code and know what it does. However often it is easier to write small comments so we can quickly scan through the functions, particularly if there are a lot of them. There are two places that function comments can go:

1. Before the function is called, often as hashed comments (#comment)
2. In the first line of the function as triple quoted comments (‘’’comment’’’). This method is called doc strings and is the preferred method for library development

**Files**

Python is very useful for reading and manipulating data files especially text files. This makes it an ideal language for system engineers who want to obtain information from complex log files but don’t want to read through thousands of lines to find what they are looking for.

To open a file you use the command open(filename,type\_of\_open). filename is the name and path to the file. If you don’t specify a path then the file will be opened in the same directory as the python file is located. type\_of\_open is a String showing which of the various ways to open a file for reading or writing that we want to use. The most common types are:

‘w’ which opens the file for writing
‘r’ which opens the file for reading
‘r+’ which opens the file for both reading and writing
‘a’ which open the file for writing but any text is added to the end of the file, appended
‘b’ on Windows only opens the file as a binary file not a text file so there is also ‘rb’, ‘wb’, ‘r+b’

To use this command you give the open a variable :

file = open(‘text.txt’, ‘r’)

Now the object file has a number of functions to work with the opened file:

file.read() reads the entire file in a String so this is usually done as text = file.read()
file.readline() read the file in a line at a time, usually done in loops
file.readlines() reads the file line by line and places it in a list
file.write(text) will write to the opened file, depending on how it is opened it will overwrite what is there or append it at the end.

Note that readline and readline expect each line to be marked by a line terminator, which is usually by pressing enter in the text editor. In Python the line terminator is ‘\n’.

These commands can exist on their own, be part of loops, assigned to variables etc depending on what you want to do.

Once you have completed working with a file it needs to be closed. This is done with a command on the object eg file.close()

It is normal practise to put all the file handling into a function where the file is opened, any handling of the file is done and then closed.

Type the following in as a program (which uses the Python template)

#ICAPRG301A Program Template
#Created by Neil Williams
#Start your code here

#This open a file given as filename for reading returns the String containing the contents of the file
def readFile(filename):
 file = open (filename, 'r')
 text = file.read()
 file.close()
 return text
#This writes the text to a file of the name given as filename
def writeFile(filename, text):
 file = open(filename, 'w')
 file.write(text)
 file.close()

#This function is the start of your program
def main():
 text = readFile('small.txt')
 print(text)

#This needs to be at the bottom of the program
if \_\_name\_\_ == '\_\_main\_\_':
 main()

Save this in the same folder as the textfile called ‘small.txt’. Now run the code. It will print out the contents of the text file as one long string. Change file.read to file.readline() and rerun the code. It should be the same since the text file has no line terminators at present. Open up the text file to see this. Now change file.readline() to file.readlines() and rerun the code. The output is now a list.

**Activity 2:**

Change the text file by pressing enter where you think that a new line should go. Rerun the code with readline and readlines to see if it makes any difference.

**Activity 3:**

Create a program which reads in the text file ‘small.txt” and creates another text file called “large.txt” with the same contents. (I have given you a function to write code to a text file)

**Activity 4:**

Create a program which reads the text file ‘small.txt” and prints out another text file call ‘pig.txt’ with the quote in Pig Latin.

**Activity 5:**

Create a program which opens ‘pig.txt’ and appends to the end of it your name and the current date.

You are now ready for your second assignment. There is a textfile which shows what the output should look like.

**Assignment 2**

**10 Green Bottles**

Develop a program that creates a text file containing the words of the song 10 green bottles, exactly like the sample. The text file is to be saved in the current directory using the name TenGreenBottles.txt. Your program must:

1. Use the Python template
2. Use at least two functions
3. Most of the song should be created in a loop
4. Use lists and variables to help create the verses

Hints:

* Creating the text file would make a good function
* Most of the song is repeated with few changes, though the last verse is different. This is a good candidate for a loop
* The code to force a new line is ‘\n’
* Use meaningful names for your lists and variables
* There is no right answer, so long as your program prints out the correct text file and satisfies the four requirements you have done the task.
* There are many ways to do this, I will get suspicious if everyone’s code is exactly the same.
* My file to do this, including the comments is 40 lines. I am sure that someone can do it in less.