# Workshop 3 – Python, Shell Scripting and Regular Expressions

## Objectives

In this workshop you will:

- Write some simple python programs to run on a remote server.
- Write a shell script that will execute your python programs and pipe input/output between them
- Write some regular expressions to match some specific text in a text file.
- Continue your python revision by solving more pylab problems with files, data structures and classes.

## Overview

In the lecture this week we looked at the Shell and discussed some of its uses. In the workshop we will be exploring the shell on a remote server and learning how to use some of its features. After you have done this, you can continue your python revision.

## Task 1 – Login to DWARF using an SSH client

- Start a terminal client (Putty is simple and installed on the lab machines, though you may use others (recommend MobaXterm))
- Login to DWARF - dwarf.ict.griffith.edu.au
- When prompted, enter your Griffith sNumber WITH the s(s1234567)
- When prompted, enter your Griffith password

## Task 2 – Practice moving around

- Spend a few moments traversing the file system using cd and ls commands.

  > cd ~ will take you back to your home directory

**Task 3 – Write 2 python scripts to run on the server**

- You have 2 options – you can write the script through the terminal client using a text editor like nano, or you can write the files on your pc locally and copy them over to the server using an SSH file transfer client.
  - WinSCP is a popular SSH file transfer client that should be installed on the lab PC's. PSCP is a command line client that you can quickly install on your laptop

  - Writing python in your local IDE and testing it is easier than writing in a non-code aware text editor like nano. If you don't feel confident copying the files over, another option would be to write the program, copy the text (CTRL-C) and paste it into nano on the SSH client.

- **Program 1 – Word Frequency**
  - Write a program that reads all words from standard input (until eof) and prints out a count for each word. The word should be processed case-insensitive (all capitals), punctuation should be removed (google python maketrans & translate or python string punctuation removal), and the output should be sorted by frequency. See lecture 2.1 slide 38 for a similar program.

    How much wood would a woodchuck chuck if a woodchuck could chuck wood.

```
WOODCHUCK 2
WOOD 2
CHUCK 2
A 2
WOULD 1
MUCH 1
IF 1
HOW 1
COULD 1
```

- **Program 2 – Histogram**
    - Write a program that generates a text-based histogram for a list of labels/values read from standard input. The histogram should show percentages of the total input values - if there were 3 input with values 10, 5 and 5, 10 would show 50% on the histogram, 5 would show 25%. For the above listing, the following histogram would print:

```
WOODCHUCK      [***] 15%
WOOD           [***] 15%
CHUCK          [***] 15%
A              [***] 15%
WOULD          [**] 8%
MUCH           [**] 8%
IF             [**] 8%
HOW            [**] 8%
COULD          [**] 8%
```

**Task 4 – Write a bash script wordhisto.sh that will produce a word histogram from any text input file**

- This program should use the 2 python programs you have written and pass input between them. You should check that the BaSH script is called with an input file as a parameter, and that the file exists.
- Use nano to make a text file to pass to your program and test it.

```
> ./wordhisto.sh inputFile
```

- Remember to make the script executable using chmod

## Task 5 – Regular Expressions

- Write a Python program to extract all email addresses ended with “.org” in mbox-short.txt (using Regular Expressions)
- Write a Python program to extract all sender’s email addresses ended with “.edu” with emails sent between 15:00 and 19:00 (using Regular Expressions).