**Python Scripting 3**

**1**

The following program prompts the user for the name of a directory. The number of entries in that directory are then counted. One of two messages will then be displayed depending on the number of entries present. Note that this script implements a user defined function called get\_input().

Using the **vi** editor, enter the following python script, call the file **ifexample.py**

**# your name and date here**

**import subprocess**

**subprocess.call([‘clear’])**

**def get\_input():**

**directory=raw\_input(“Enter a directory name: “)**

**return directory**

**directory=get\_input()**

**filecount=subprocess.check\_output([‘ls %s | wc –l’ % directory], shell=True)**

**if int(filecount)<10:**

**print “There are less than 10 entries in %s, actual count is %s” % (directory,filecount)**

**else:**

**print “There are at least 10 entries in %s, actual count is %s” % (directory,filecount)**

Save the file, change the permissions and execute it. Run test to verify each condition i.e. enter a directory that contains less than 10 entries then test for a directory that contains more than 10 entries.

**2**

The following example prompts the user for the name of a file or directory. Tests are then carried out to determine if the entry is a file or directory, and, if it is a file whether or not it has non-zero length. Suitable messages will be displayed depending on the outcome of the tests.

Using the **vi** editor, enter the following shell script, call the file **elifexample.py**,

**# your name and date here**

**import subprocess**

**import os**

**subprocess.call([‘clear’])**

**def get\_input():**

**return raw\_input(“Enter a file or directory name: “)**

**input=get\_input()**

**if os.path.isdir(input):**

**print “%s is a directory” % input**

**elif os.path.isfile(input) and os.path.getsize(input)>0:**

**print “%s is a file of non-zero length” % input**

**else:**

**print “%s is a file of zero length” % input**

Save the file, change the permissions and execute it. Run test to verify each conditioni.e. enter a directory, enter a filename of a zero length file, enter a filename of a non-zero length file.

Rather than nesting 'if..then' statements, the elif operator can be used.

**3**

The following program asks the user for a number, and then displays the corresponding day of the week.

Using the **vi** editor, enter the following shell script, call the file **dictionaryexample.py**,

**# your name and date here**

**import subprocess**

**subprocess.call([‘clear’])**

**def numbers\_to\_days(argument):**

**switcher = {**

**“1”: “Sunday”,**

**“2”: “Monday”,**

**“3”: “Tuesday”,**

**“4”: “Wednesday”,**

**“5”: “Thursday”,**

**“6”: “Friday”,**

**“7”: “Saturday”**

**}**

**return switcher.get(argument, “invalid day”)**

**day = raw\_input(“Please enter a number between 1 and 7”)**

**print numbers\_to\_days(day)**

Save the file, change the permissions and execute it. Run tests to verify that for each selection between 1 and 7 you get the correct day of the week displayed. Test to see what happens if you enter a number out with the range.

**4**

The following program reads in a single character, and then prints a message to indicate whether the characters is a numeric digit between 0 and 9, a lower case character, an upper case character or something else. It imports the regular expression module **re**, which enable us to perform pattern matching.

Using the **vi** editor, enter the following shell script, call the file **regexexample.py**,

**# your name and date here**

**import subprocess**

**import re**

**subprocess.call([‘clear’])**

**uppercase\_pattern = ‘[A-Z]’**

**lowercase\_pattern = ‘[a-z]’**

**digit\_pattern = ‘[0-9]’**

**def does\_it\_match(character):**

**if re.match(uppercase\_pattern, character):**

**return “Uppercase character”**

**elif re.match(lowercase\_pattern, character):**

**return “Lowercase character”**

**elif re.match(digit\_pattern, character):**

**return “Digit between 0 and 9”**

**else:**

**return “Unable to match”**

**character = raw\_input(“Please enter a single character: “)**

**print does\_it\_match(character)**

Save the file, change the permissions and execute it. Run tests to verify each of the different conditions i.e. enter a lowercase character, enter an uppercase character, enter a digit, enter a symbol.

**5**

The following program displays a menu, giving the user options to display a long or a short listing. If a valid choice is selected, an appropriate command is issued, and the program abandoned, otherwise the user will be informed this is an invalid choice.

Using the **vi** editor, enter the following shell script, call the file **menuexample.py**

**# your name and date here**

**import subprocess**

**subprocess.call([‘clear’])**

**def display\_menu():**

**menu = {**

**“1”: “Long Listing”,**

**“2”: “Wide Listing”**

**}**

**options = menu.keys()**

**options.sort()**

**for entry in options:**

**print entry, menu[entry]**

**def long\_listing():**

**print subprocess.check\_output([‘ls –l’], shell=True)**

**def wide\_listing():**

**print subprocess.check\_output([‘ls’], shell=True)**

**def invalid():**

**print “Invalid menu choice, should be 1 or 2”**

**def process\_selection(choice):**

**switcher = {**

**“1”: long\_listing,**

**“2”: wide\_listing**

**}**

**func = switcher.get(choice, invalid)**

**return func()**

**display\_menu()**

**selection = raw\_input(“Please select an option from the menu, 1 or 2”)**

**process\_selection(selection)**

Save the file, change the permissions and execute it. Run tests to verify that each option operates as expected i.e. option 1 produces a long listing and option 2 produces a wide listing and any other option displays a message “Invalid menu choice, should be 1 or 2”.