

## Programming with Python Workbook

### Example 1

A simple if statement

```
a=10

if(a!=11):
    print ("The value of a is not 11")
```

A simple if else statement

```
a=10
if(a>10):
    print("Value of a is greater than 10")
else :
    print("Value of a is 10")
```

If I want to print the value of variable a in the message I can. I will need to use the %s notation to automatically coerce a non-string value into a string. In the example I cannot join the integer variable a to the string message.

```
a=10

if(a!=11):
    print ("The value of a is not %s" %(a))
```

I can join two string together using the + notation.

```
a=10

if(a!=11):
    print ("The value of a" + " is not %s" %(a))
```

## Example 2

The for loop is often used for cycling through a list. In the example below the variable 'bottle' will cycle through a range which starts at 10 and ends at 0. To do this the increment has to be -1 as this is a count down from 10 to 0. Note also that the %s is being used to format the message. Also note that three lines of print have been indented.

```
for bottles in range(10, 0, -1):  
    print("%s green bottles hanging from the wall " %(bottles) )  
    print("And if one green bottle should accidentally fall")  
    print(" There will be %s green bottles hanging from the wall" %(bottles-1))
```

I can use a while loop to do something similar. But in this case I will need to define my variables 'bottles' before the loop starts, and I will need to perform the count down within the loop with the command 'bottles = bottles-1'.

```
bottles=10  
while(bottles != 0):  
    print("%s red bottles hanging from the wall " % (bottles))  
    print("And if one red bottle should accidentally fall")  
    print(" There will be %s red bottles hanging from the wall" % (bottles - 1))  
    bottles = bottles - 1
```

## Example 3

The code creates a function called 'testClass'

```
class testClass:  
    def __init__(self, message):  
        self.message = 'Hello ' + message  
  
    def getMessage(self):  
        return self.message  
  
msg = testClass("you")  
  
print(msg.getMessage())
```

## Example 4

The code below opens a file called 'output.txt' for writing (w) in the directory 'C:\\Users\\rskeggs\\Desktop\\tmp\\'. The message 'Hello World' is written to the file. Because the 'with' statement is used the call to close the file is not required.

```
with open('C:\\Users\\rskeggs\\Desktop\\tmp\\output.txt', 'w') as tmp:  
    tmp.write('Hello World')
```

## Example 5

The code below imports the datetime module and prints out a timestamp.

```
import datetime

print (datetime.datetime.now())

# format the date time string
today=datetime.datetime.now()
print (today.strftime('%d-%b-%y'))

# format the time
print(today.strftime('%H:%M:%S'))
```

## Exercise 1

This exercise will create a little alarm clock which will display a message.

Step 1. Import the date timemodule.

Step 2: Get the current date into a variable.

Step 3: Create a loop that does not end. One way to do this is use 'while True:'

Step 4: Create the condition for the loop to break. Suggest using an 'if' statement. Test the time to see if it matches the alarm time you want to set. Would suggest using 'today.hour ==' and 'today.minute ==' as part of the if statement. Use the and keyword between both statements so both conditions have to be true.

Step 5: indent the message you want to display so it is part of the if statement and use the keyword 'break' to exit from the loop.

Step 6. Would also ensure that the test setup up I step 2 is part of the loop statement.

## Example 6

The following code opens up a file using the 'with' clause. The for loop then process each line at a time. The code uses two print statements one is commented out. Test both statements to see what happens.

```
with open("C:\\Users\\rskeggs\\Desktop\\GINF.csv", "r") as inf:
    for line in inf:
        #print (line.rstrip('\n'))
        print (line)
```

## Example 7

The example below uses a loop to write the message to file. Note that the use of 'a' which means append. If the w mode is used then any attempt to write to the same file will over write existing content. Also note that everything is written to one line in the file. To add content to a new line the new line character will need to be added to the write string in the case of windows this is '\n'.

```
# Using the loop to write text to file.

with open("C:\\Users\\rskeggs\\Desktop\\GINF1.csv", "a") as outf:
    for bottles in range(10, 0, -1):
        outf.write("%s green bottles hanging from the wall " % (bottles))
        outf.write("And if one green bottle should accidentally fall")
        outf.write(" There will be %s green bottles hanging from the wall" %
(bottles - 1))
```

## Example 8

In Python like most programming languages there are a number of approaches that can be used to open a file. Either the entire file can be loaded into a variable

```
with open("C:\\Users\\rskeggs\\Desktop\\GINF.csv") as inf:
    data = inf.readlines()
    for line in data:
        print(line)
```

Or the file can be processed line by line.

```
with open("C:\\Users\\rskeggs\\Desktop\\GINF.csv") as inf:
    for line in inf:
        print(line)
```

When working with a large file the better approach is to use the second approach.

## Example 9

Some simple processing

```
# Split the line of data on the space character
words=line.split()

# count the number of words in the file.
count = len(words)

#print first two words from the list
print(words[:2])
```

## Exercise 2

In this exercise we will open up a CSV file and begin processing data.

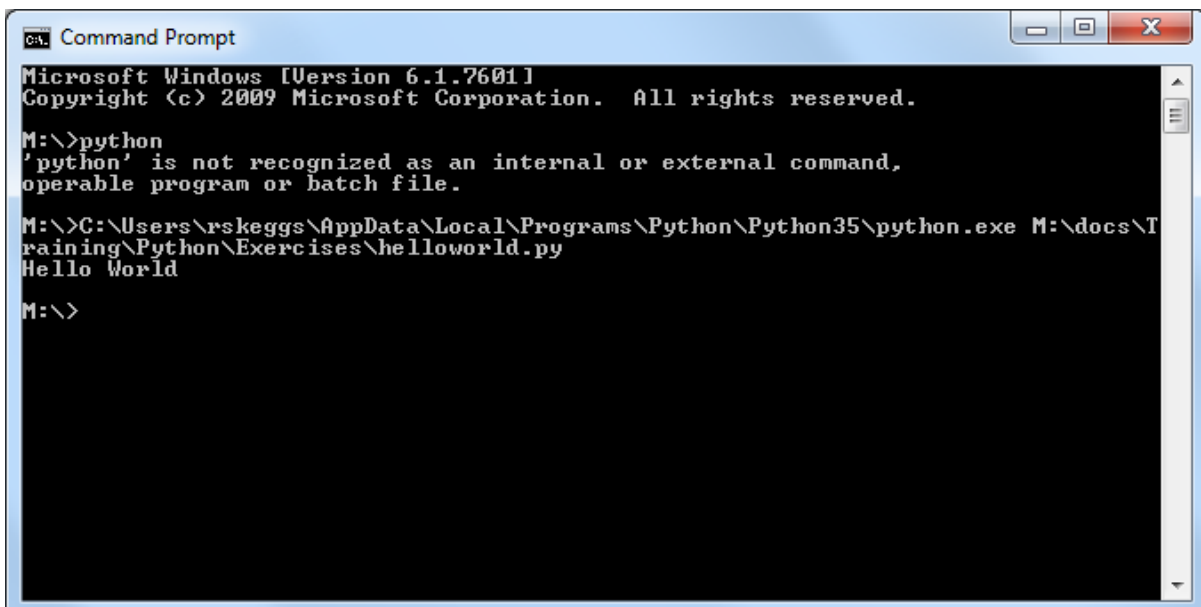
The python script below will open up a CSV file and line by line will process the file. In this case the process is to simply find all occurrences of ,, and replace with ,NA, before writing the data to a new file.

```
#
inFile = open("M:\\docs\\Training\\Python\\Exercises\\Exercise2\\loanbook.csv",
"w")
with open("M:\\docs\\Training\\Python\\Exercises\\datafile\\loanbook.csv", "r+")
as f:
    data = f.readlines()
    lines = ''.join(data)
    if lines.find(",") != -1 :
        newString = lines.replace(",", " ,NA,")
inFile.writelines(newString)
```

## Exercise 3

We are going to create a simple hello world program and execute on the command line. The screen shot shows the output from the file.

1. Create a file called helloworld.py
2. To the python script file add the string print("Hello World")
3. Save the file
4. Now execute the script
  - On the command line enter the full path to the python.exe
  - Then to the same line add the full path to the hello world script.



```
C:\> Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

M:\>python
'python' is not recognized as an internal or external command,
operable program or batch file.

M:\>C:\Users\rskeggs\AppData\Local\Programs\Python\Python35\python.exe M:\docs\T
raining\Python\Exercises\helloworld.py
Hello World

M:\>
```

- Try creating a batch script that contains the full path to the python executable and contains the full path to the python script.

- Now create a batch script which takes a parameter and passes it through to the python script. (It might be an idea to print out the parameter by adding the line `echo %1` to the batch file).
- Change to python script to take the parameter passed by the batch file and print the value to screen.

The Python script:

```
import sys  
  
print(sys.argv[1])
```

The batch script

```
echo off  
  
C:\Users\rskeggs\AppData\Local\Programs\Python\Python35\python.exe  
M:\docs\Training\Python\Exercises\helloworld.py %1  
  
pause;
```

The command line entry is

```
M:\docs\Training\Python\Exercises\run.bat hello
```