

UNIVERSITY OF WEST LONDON

Logbook For Intro To Software development

Assignment 2

Hamza Bhatti

21223241

Contents

Introduction	3
Week 4	3
Exercise 1	3
Exercise 2	4
Exercise 3	5
Exercise 5	7
Exercise 6	8
Exercise 7	9
Week 5	10
Exercise 1	10
Exercise 2	12
Exercise 3	13
Exercise 4	14
Exercise 5	16
Week 6 & 7	17
Exercise 1	17
Exercise 2	18
Exercise 3	19
Exercise 4	19
Exercise 5	21
Week 8	23
Exercise 1	23
Exercise 2	24
Exercise 3	25
Exercise 4	26
Exercise 5	27
Week 9	28
Exercise 1	28
Exercise 2	29
Exercise 3	30
Exercise 4	32

Exercise 5.....	34
Week 10	36
Exercise 1.....	36
Exercise 2.....	37
Pushing Tasks to BitBucket	38
Week4	38
Week5	38
Week6&7.....	39
Week8	39
Week9	40
Week10	40
Full Evaluation of project	41

Introduction

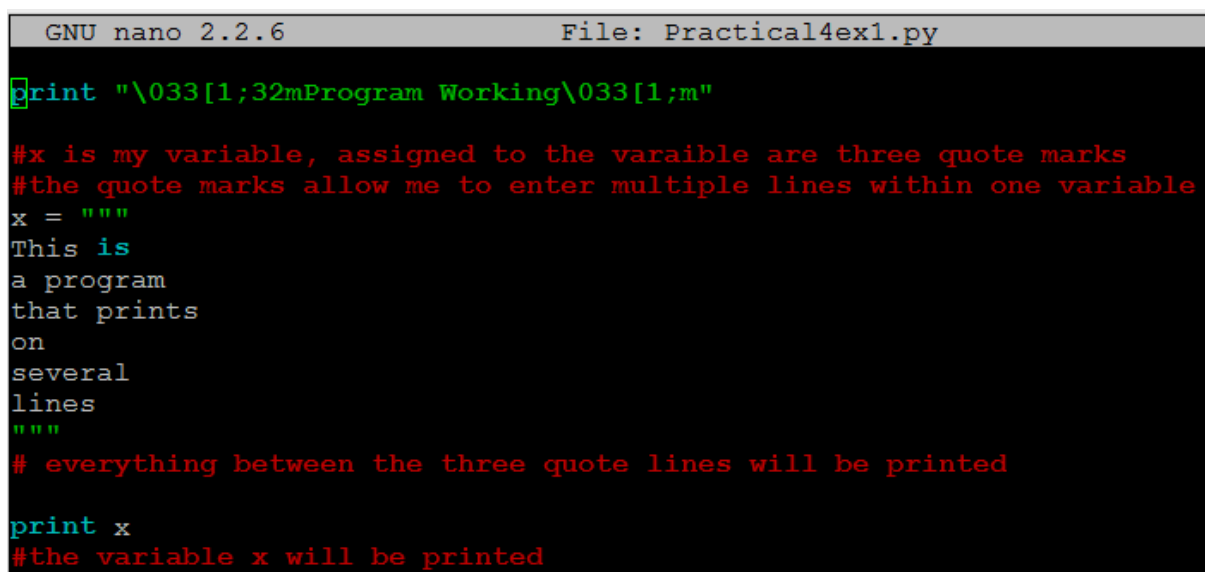
In this piece of documentation, I am presenting a series of exercises that needed to be complete using Python programming. Nano, an editor in Linux, was used to write the code for each task. In this document, I will show the task that had to be completed, along with the code that was written, executed programs and an evaluation about each task.

Week 4

Exercise 1

Write a program that says: "This is a computer program that prints on several lines."

Code:

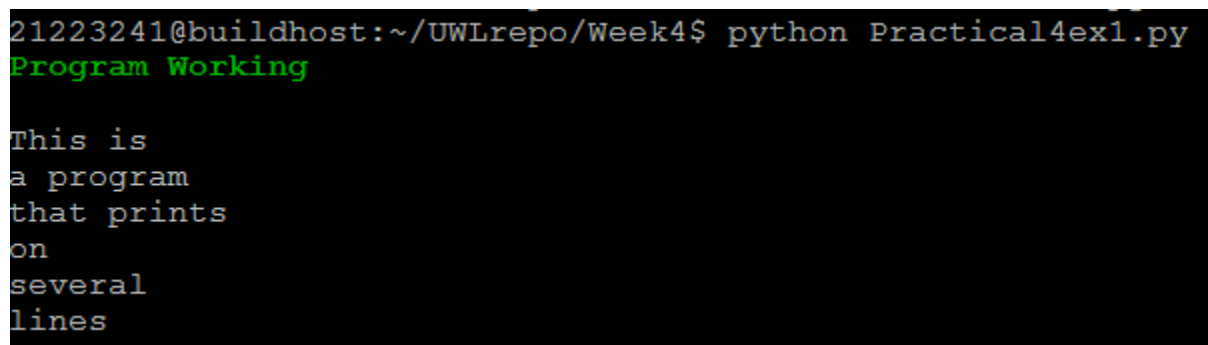


```
GNU nano 2.2.6 File: Practical4ex1.py
print "\033[1;32mProgram Working\033[1;m"

#x is my variable, assigned to the variable are three quote marks
#the quote marks allow me to enter multiple lines within one variable
x = """
This is
a program
that prints
on
several
lines
"""
# everything between the three quote lines will be printed

print x
#the variable x will be printed
```

Program:



```
21223241@buildhost:~/UWLrepo/Week4$ python Practical4ex1.py
Program Working

This is
a program
that prints
on
several
lines
```

Evaluation:

This was an easy task to complete as it did not require much skill to complete. The task was competed in less than a minute. This was mostly due to the fact that I had completed a similar task that was set on a website called "Learn python the hard way."

Exercise 2

Write a program that will display a joke.

Don't display the punch line until the reader hits the enter key.

Challenge task:

Display the punch line in a different colour.

Code:

```
GNU nano 2.2.6 File: Practical4ex2.py

print "\033[1;32mProgram Working\033[1;m"

print "Why did the golfer have an extra sock?"#The question has been printed

userinput = raw_input ("Press Enter to see the punch line ")
#I have instructed the user to press enter to see the punchline.
#to do this I had created a raw_input instruction before the string.

print "\033[1;32mIncase he got a whole in one\033[1;m"
#When the user presses enter, they will then see the punchline
# \033[1;32m is associate with green and is on begining and end of the line
# to output green.[]
```

Program:

```
21223241@buildhost:~/UWLrepo/Week4$ python Practical4ex2.py
Program Working
Why did the golfer have an extra sock?
Press Enter to see the punch line
Incase he got a whole in one
```

Evaluation:

This was another easy task to complete. This was mainly due to the fact the task required me to print two lines and create one variable. The variable that was made was a `raw_input`, which acts an instruction for the user.

An issue that I did face was the challenge task. This was to display the punch line in a different colour. This would allow it to stand out. I searched a few websites, all containing answers which required an installer command in Ubuntu terminal. I had attempted using one of these installer codes. I was then given the code from one of my colleagues to change the output colour of the punch line.

Exercise 3

Write a program that will ask you your name. It will then display “Hello *Name*” where “*Name*” is the name you have entered.

Challenge task:

Complete the following program so it uses the variables to print out ‘the cat sat on the mat’. It should print out on one line, with spaces.

E.g. word1='the' word2='cat' word3='sat' word4='on' word5='the' word6='mat'

Code:

```
GNU nano 2.2.6 File: Practical4ex3.py Modified
print "\033[1;32mProgram Working\033[1;m"

name = raw_input ("What is your name? ")
# I instructed the user to enter their name. the raw input will be associate with the
# variable name.

print "Hello %r" % name
# the program will print the name. %r (raw input) acts as a store within a string.
# outside the string the store needs to be defined.

word1 = 'The'
word2 = 'cat'
word3 = 'sat'
word4 = 'on'
word5 = 'the'
word6 = 'mat.'
# i have assigned strings to many variables

print "%s %s %s %s %s %s" % (
    word1, word2, word3, word4, word5, word6
)
# to the print the variables i used the store %s (string. i then identified what variables
# are being used.
```

Program:

```
21223241@buildhost:~/UWLrepo/Week4$ python Practical4ex3.py
Program Working
What is your name? Hamza
Hello 'Hamza'
The cat sat on the mat.
```

Evaluation:

This task required more time to complete as many variables had to be created. The name variable was a raw input. The name entered was then used to greet the user. Six other variables were then made for six separate strings. The strings were outputted onto one line. I had previously completed a task with the same method from “Learn python the hard way.”

Exercise 4

Write a program that asks the user their name and then asks what their favourite food is, using their name in the question, and responds to their answer.

Code:

```
GNU nano 2.2.6 File: Practical4ex4.py

print "\033[1;32mProgram Working\033[1;m"

name = raw_input("What is your name?")
print "Hello %s " % name
# i instruct the user to enter their name with a raw input. it is assigned to the
# variable name. the user is greeted with their name in the string

food = raw_input("What is your favourite food " +name+ "?")
# the user is then asked another question. the variable name is added to the string.
# it is a string within another string

print "I like %s too" % food
# the user is replied to with the food they mentioned
```

Program:

```
21223241@buildhost:~/UWLrepo/Week4$ python Practical4ex4.py
Program Working
What is your name?Hamza
Hello Hamza
What is your favourite food Hamza?Pizza
I like Pizza too
```

Evaluation:

There was one issue that I found in this exercise, and that was including the variable name within the variable food. When I first attempted writing the variable food, I assumed including the users name would require me to enter %r. This %r is associated with raw input. When I did this, the name of the user did appear, but with quote marks. What I needed to do was add a string within a string. This was done by writing a sentence, then +variable+. When this was entered the quote marks did not appear.

Exercise 5

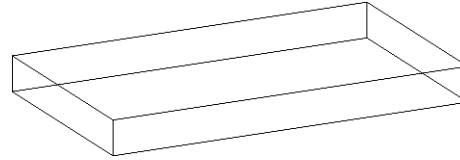
Write a program to work out the areas of a rectangle.

Collect the width and height of the rectangle from the keyboard

Calculate the area and display the result.

Extension:

Display the volume of a cuboid.



Code:

```
GNU nano 2.2.6 File: Practical4ex5.py
print "\033[1;32mProgram Working\033[1;m"

print "An area of a rectangle is measred by multiplying its height by the width"
print "Area = Height x Width" # quick description of how to calculate area

height = float(raw_input("Please enter a Height in cm ")) # user inputs as float values
width = float(raw_input("Please enter a Width in cm "))

area = height*width
print "Your area is", area, "cm squared"
#The 2 user inputs are then multiplied together to give the area

print "To calculate volume, we need another value, which is length"
print "We will use your previous numbers, all you need to is add a value for length"
print "Volume = Width x Height x Length" # another description but for volume

length = float(raw_input("Enter your Length "))
volume = area*length #the user input for length is multiplied by identified area
print "Your volume is", volume, "cm cubed"
```

Program:

```
21223241@buildhost:~/UWLrepo/Week4$ python Practical4ex5.py
Program Working
An area of a rectangle is measred by multiplying its height by the width
Area = Height x Width
Please enter a Height in cm 12
Please enter a Width in cm 33
Your area is 396.0 cm squared
To calculate volume, we need another value, which is length
We will use your previous numbers, all you need to is add a value for length
Volume = Width x Height x Length
Enter your Length 44
Your volume is 17424.0 cm cubed
```

Evaluation:

The task didn't require any difficult code to be entered. Like the previous tasks, this was mainly printing variables and data manipulation. The formula was simple to create a program for. I did not find any hurdles when completing this task.

Exercise 6

Write a program that has 3 variables (a, b and c).

a = 12 and b = 6.

c = a + b

The program should print the value of c.

Code:

```
GNU nano 2.2.6 File: Practical4ex6.py
print "\033[1;32mProgram Working\033[1;m"
print "This is simple addition"
a = float(raw_input("Enter any number: "))
b = float(raw_input("Now enter another number: "))
# a and b are variables which require the user to enter any number
c = a + b
# c is another variable which adds the two inputs entered and provides an answer
print "Your value is", c
# c is printed
```

Program:

```
21223241@buildhost:~/UWLrepo/Week4$ python Practical4ex6.py
Program Working
This is simple addition
Enter any number: 2
Now enter another number: 3
Your value is 5.0
```

Evaluation:

This was another easy task as it only required me to assign raw inputs to variables and display another. I found no trouble completing this task.

Exercise 7

Write a program that asks the user their first name and then asks them their surname and then prints their whole name 3 times.

E.g.

What is your first name?

What is your last name?

John Doe

John Doe John Doe John Doe

Code:

```
GNU nano 2.2.6 File: Practical4ex7.py

print "\033[1;32mProgram Working\033[1;m"

firstname = raw_input("What is your first name? ")
secondname = raw_input("What is your second name? ")
# the user is instructed to input their first and second name

print (firstname + " " + secondname + ", ")*3
# the first and last name are printed. the string will be outputted 3 times
# as there is a *3, which means repeat 3 times
```

Program:

```
21223241@buildhost:~/UWLrepo/Week4$ python Practical4ex7.py
Program Working
What is your first name? Hamza
What is your second name? Bhatti
Hamza Bhatti, Hamza Bhatti, Hamza Bhatti,
```

Evaluation:

Overall this task required the same steps as the previous ones. However, I did find an issue on the last line of the code. The issue was getting the correct spacing for when the strings were being printed next to each other. I had to do a little manipulation to get the correct spacing between the strings. Ultimately, the task was not difficult.

Week 5

Exercise 1

Write a program that will work out the distance travelled if the user enters in the speed and the time.

Hint:

Speed = Distance / Time

Therefore, Distance = Speed * Time

Code:

```
GNU nano 2.2.6      File: Practical5ex1a.py

print "\033[1;32mProgram Working\033[1;m"

print "We can work out Distance travelled by multiplying Speed by Time"
speed = float(raw_input("Please enter your speed in mph "))
time = float(raw_input("Please enter the amount of time it took in mins "))
#user is asked to input the speed and time

distance = speed*(time/60) # the speed is then multiplied by time(which is converted to hrs

print "Your distance travelled in miles was ", distance
#The answer is then displayed
```

Program:

```
21223241@buildhost:~/UWLrepo/Week5$ python Practical5ex1a.py
Program Working
We can work out Distance travelled by multiplying Speed by Time
Please enter your speed in mph 23
Please enter the amount of time it took in mins 120
Your distance travelled in miles was 46.0
```

Evaluation:

This was a simple mathematical task, similar to one in week 4. I didn't find any difficulty completing the task, the only errors I did find were syntax errors that were a result of misspelling.

Challenge task:

Get the program to tell you the speed you would have to travel at in order to go a distance within a certain time entered by the user.

Code:

```
GNU nano 2.2.6      File: Practical5ex1b.py

print "\033[1;32mProgram Working\033[1;m"

print "I can tell you how fast you need to go within a certain time and distance"

distance = float(raw_input("Enter how far you need to go in miles "))
time = float(raw_input("Enter the amount of time you have in mins "))
# user is asked to input time and distance

speed = distance*(time/60)
# distance is multiplied by time (which has been converted to hrs)

print "You need to travel at ", speed, " mph."
print "What are you still doing here, GO!"
# the speed is printed
```

Program:

```
21223241@buildhost:~/UWLrepo/Week5$ python Practical5ex1b.py
Program Working
I can tell you how fast you need to go within a certain time and distance
Enter how far you need to go in miles 90
Enter the amount of time you have in mins 120
You need to travel at  180.0  mph.
What are you still doing here, GO!
```

Evaluation:

The challenge task was completed as a separate file. This didn't require me to add any other code, rather, rearranging the variables.

Exercise 2

Write a program that uses the following variables to calculate the number of minutes in a week.

Code:

```
GNU nano 2.2.6 File: Practical5ex2.py
print "\033[1;32mProgram Working\033[1;m"

# variables that are needed
daysperweek = 7 #days
hoursperday = 24 #hours
minutesperhour = 60 #mins
#Formulas below

minsaday = minutesperhour*hoursperday
minsaweek = minsaday*daysperweek
# all the calculations have been stored

print "We have 7 days a week. Each day lasts 24 hours in each hour we have 60 mins"

print """
To calculate how many minuts there are in a week we must do the following:
First we need to find out how many minutes there are in a day.
We will multiply the minutes in one hour by the hours in a day.
"""
print "Which 60 x 24, which gives us:" , minsaday # displaying one of the calcs

print "All we need to do now is multiply our minuts in a day which is by the days of the week, which is 7"
print "Our minuts in a week are:"
print minsaweek
# displaying the answer
```

Program:

```
21223241@buildhost:~/UWLrepo/Week5$ python Practical5ex2.py
Program Working
We have 7 days a week. Each day lasts 24 hours in each hour we have 60 mins

To calculate how many minuts there are in a week we must do the following:
First we need to find out how many minutes there are in a day.
We will multiply the minutes in one hour by the hours in a day.

Which 60 x 24, which gives us: 1440
All we need to do now is multiply our minuts in a day which is by the days of the week, which is 7
Our minuts in a week are:
10080
```

Evaluation:

Unlike the previous tasks, I did face many minor troubles when completing it. One of the troubles I faced was making sure the variables that I had created at the beginning where entered exactly the same later on. I found many of these syntax errors when I ran the program. I did not face any troubles when it came to the maths of the program.

Exercise 3

Write a program that:

- (i) asks the user their name
- (ii) says 'Hello' to them, using their name
- (iii) and then asks how old they are.

The program should then calculate their age at their *next* birthday and respond "So, at your next birthday you are (age)?"

Code:

```
GNU nano 2.2.6 File: Practical5ex3.py

print "\033[1;32mProgram Working\033[1;m"

name = raw_input("What is your name? ") # User is asked to input their name

age = int(raw_input("How old are you " +name+ "? "))
# the variable is added as a string in the new question, which asks the user to
# input age

newage = age + 1
# The age they will be next year is calculated by adding 1 to curentt age

print "Oh, so your on your next birthday your going to be" ,newage
# the new age is then printed
```

Program:

```
21223241@buildhost:~/UWLrepo/Week5$ python Practical5ex3.py
Program Working
What is your name? Hamza
How old are you Hamza? 20
Oh, so your on your next birthday your going to be 21
21223241@buildhost:~/UWLrepo/Week5$
```

Evaluation:

This task was no different from previous tasks. The variable "name" was added to another string, similar to a previous task. The task didn't require any new skills to complete it.

Exercise 4

Write a program to work out how many days you have lived for.

Algorithm:

Enter the date of birth. Get today's date Calculate the difference in days between the two dates. Display the result

Challenge task:

Work out how many seconds you have lived for.

Code:

```
GNU nano 2.2.6 File: Practical5ex4.py

print "\033[1;32mProgram Working\033[1;m"

#importing the date function/ module
import datetime

print "I can tell you how many days you have lived for..."

#These are the raw inputs for the user to enter the date,month and year
date = int(raw_input("Enter the date that you were born on "))
month = int(raw_input("Now enter the month you were born on "))
year = int(raw_input("Finally, enter the year that your were born on "))

#format for date of birth and todays date for thr function
dob = datetime.date(year, month, date) # year month and date are the arguments
today = datetime.datetime.today().date() # no arguments here, so puts standard time in

print "Todays date is the %d" % today.day
print "The month we are in is %d" % today.month
print "And the year we are in is %d" % today.year
# showing the current date and time with the time function

days = today - dob

print "You have lived a total of ", days.days, "days"
# the days are then presented with the format days.days function

sec = 60
secondsinhr = 60*sec
secondsinday = 24*secondsinhr
# seconds in a day is then calculated with an functions

print "You have also lived for", days.days*secondsinday, "seconds"
# the days.days calculated is then multiplied by the seconds in a day to give
# how many seconds the user has lived for
```

(RUN PROGRAM ON NEXT PAGE)

Program:

```
21223241@buildhost:~/UWLrepo/Week5$ python Practical5ex4.py
Program Working
I can tell you how many days you have lived for...
Enter the date that you were born on 12
Now enter the month you were born on 12
Finally, enter the year that your were born on 200
Todays date is the 7
The month we are in is 1
And the year we are in is 2014
You have lived a total of 662211 days
You have also lived for 57215030400 seconds
```

Evaluation:

When I had initially attempted the task, the variables and calculation I had written only worked for my birthday. This would have been inappropriate as the program would not work for any other user. A colleague of mine put it to my attention that I needed to import a function called time. This allowed integers to be manipulated as time, allowing the task to complete effectively. The challenge task only required three more variables to be written, using the variables that were already present.

Exercise 5

Make a game for seeing how good people are at guessing when 10 seconds have elapsed.

Algorithm:

Tell them to hit the enter key when ready

Get the first time in seconds

Tell them to hit the enter key when they think 10 seconds have elapsed.

Get the second time in seconds.

Subtract first time from the second time

Tell them how close to 10 the answer was.

Code:

```
GNU nano 2.2.6      File: Practical5ex5.py
print "\033[1;32mProgram Working\033[1;m"
import time # time function is imported
begin = raw_input("Lets see how good you are at guessing 10 seconds. Press enter the start $
# raw input instructing the user to press enter

start =int( time. time())
# start of the counter

end = raw_input("Press enter when you think 10 seconds has elapsed")
# instruction of the user, the count has started

print "You guessed", int(time.time()) - start, "seconds."
# the time that the user hit enter is displayed
```

Program:

```
21223241@buildhost:~/UWLrepo/Week5$ python Practical5ex5.py
Program Working
Lets see how good you are at guessing 10 seconds. Press enter the start the timer
Press enter when you think 10 seconds has elapsed
You guessed 7 seconds.
```

Evaluation:

The practical also required the same time function as the previous task. The task was quite difficult to understand. I had to look to the internet to find how to use the timer with the time function. The variable "start" used the time function. The last line of the program, where the time elapsed was displayed, was also hard to understand.

Week 6 & 7

Exercise 1

Write a program that will accept someone's date of birth and work out whether they can vote (i.e. are they 18?)

Code:

```
GNU nano 2.2.6 File: Practical7ex1_2.py

print "\033[1;32mProgram Working\033[1;m"

print "You have to be a certain age for to be able to vote "
age = int (raw_input ("Please type in your age: "))
# the user is asked to enter their age

if age >= 18 : # if the age is 18 and over, they can vote
    print "You can vote."

else: # otherwise they are not allowed
    print "You can't vote. Sorry."
```

Program:

```
21223241@buildhost:~/UWLrepo/Week7$ python Practical7ex1_2.py
Program Working
You have to be a certain age for to be able to vote
Please type in your age: 13
You can't vote. Sorry.
21223241@buildhost:~/UWLrepo/Week7$ python Practical7ex1_2.py
Program Working
You have to be a certain age for to be able to vote
Please type in your age: 23
You can vote.
```



Hamza Bhatti committed 9 minutes ago ([raw commit](#))

Added week7 ex1 with comments

Evaluation:

This was the first task that required me to enter an IF statement. The writing of the IF statement was not very hard. I did however make two versions of this exercise. This version shows the use of an IF statement and an else. Rather than having two if statements, which was how I completed the other version.

Exercise 2

Write a program that will generate a random playing card e.g. '9 Hearts', 'Queen Spades' when the return key is pressed. Rather than generate a random number from 1 to 52, create two random numbers – one for the suit and one for the card.

Challenge task:

Make a loop structure so playing cards can keep being generated.

Code:

```
GNU nano 2.2.6 File: Practical7ex2_test.py Modified
print "\033[1;32mProgram Working\033[1;m"

import random # importing the random number generator function
suit = random.randint (1,4) # suit has been given a random number, ranging 1-4
card_num = random.randint(1,13) # the card number has also been given a random number 1-13

#if statements which assign a suit name and type to the the random numbers
if card_num == 1:
    suit_type = "Ace"
if card_num == 11:
    suit_type = "Jack"
if card_num == 12:
    suit_type = "Queen"
if card_num == 13:
    suit_type = "King"
if card_num == 2:
    suit_type = "2"
if card_num == 3:
    suit_type = "3"
if card_num == 4:
    suit_type = "4"
if card_num == 5:
    suit_type = "5"
if card_num == 6:
    suit_type = "6"
if card_num == 7:
    suit_type = "7"
if card_num == 8:
    suit_type = "8"
if card_num == 9:
    suit_type = "9"
if suit == 1:
    suit_name = "Heart"
if suit == 2:
    suit_name = "Diamond"
if suit == 3:
    suit_name = "Clubs"
if suit == 4:
    suit_name = "Spades"

print "Your random card is " + suit_type + " " + suit_name
# the card suit and number are printed for the user
```

Program:

```
21223241@buildhost:~/UWLrepo/Week7$ python Practical7ex2_test.py
Program Working
Your random card is 2 Diamond
21223241@buildhost:~/UWLrepo/Week7$ python Practical7ex2_test.py
Program Working
Your random card is 2 Diamond
```

(Evaluation on the next page)

Evaluation:

This task required another function to be imported for it to work appropriately. With the range of the random numbers being identified, I had to create a large amount of if statements so that the number of the card and the card type would output accordingly. This wasn't difficult, but was time consuming and had large potential for many typing errors. These errors did occur.

Exercise 3

Write a program that asks the user how long, on average, they spend on a computer per day.

If the user spends less than 2 hours per day, the program needs to say, 'That seems reasonable'

Else if it is less than 4 hours per day says 'Do you have time for anything else?'

Else, the program needs to say, 'You need to get some fresh air once in a while and a life'

Code:

```
GNU nano 2.2.6      File: Practical7ex3.py
print "\033[1;32mProgram Working\033[1;m"

hrsspnt = int(raw_input("So tell me, how long do you spend on the computer? "))
# user is asked to enter how long they spend on the computer

if hrsspnt <= 3:
    print "Sounds reasonable."

elif hrsspnt == 4:
    print "Do you have time for anything else?"

elif hrsspnt >= 5:
    print "Dude you need a life"
# if statements give a differnt response to the user depending on what they enter
print "Thats all folks"
```

Program:

```
21223241@buildhost:~/UWLrepo/Week7$ python Practical7ex3.py
Program Working
So tell me, how long do you spend on the computer? 1
Sounds reasonable.
Thats all folks
21223241@buildhost:~/UWLrepo/Week7$ python Practical7ex3.py
Program Working
So tell me, how long do you spend on the computer? 7
Dude you need a life
Thats all folks
```

Evaluation:

This task was simpler than the previous tasks within this week. The IF statements, along with the elif statements where simple.

Exercise 4

Write a program that behaves like a calculator.

The program will ask for two numbers and then give you a menu where you can choose to add, subtract, multiply or divide the number of find the power of.

Code:

```
GNU nano 2.2.6 File: Practical7ex4.py
print "\033[1;32mProgram Working\033[1;m"
print "Want to do some maths?"

n1 = float(raw_input("Enter any number you want "))
#user is instructed to enter a number

# user is shown the list of functions they can pick from
function = raw_input("""Now pick a function. You can select from:
+ - * or / . Type it in and press enter
""")# the user must enter one of the funtion

n2 = float(raw_input("Now enter another number ")) # user put 2nd number

# variables are created using the functions in the list
add = n1 + n2
sub = n1 - n2
mult = n1 * n2
divi = n1 / n2

# if statements are created which will display an answer depending on
# which function was entered. the answers are taken from the variables
# above (add sub etc)
if function == "+":
    print "Your answer is" , add
elif function == "-":
    print "Your answer is" , sub
elif function == "*":
    print "Your answer is" , mult
elif function == "/":
    print "Your answer is" , divi
```

Program:

```
21223241@buildhost:~/UWLrepo/Week7$ python Practical7ex4.py
Program Working
Want to do some maths?
Enter any number you want 67
Now pick a function. You can select from:
+ - * or / . Type it in and press enter
-
Now enter another number 34
Your answer is 33.0
```

Evaluation:

The task required a few variables to be entered, along with three user inputs. I came across many errors which related to the way I had entered the code. Because of the many variables that needed to be entered, I did miss a few quote marks and brackets. I did not include the ability to square the numbers as it would require the whole program to be arranged differently.

Exercise 5

Extend the program in Week 5 Exercise 4 to make a game for seeing how quick people are at typing the alphabet.

Algorithm:

Tell them to hit enter key when ready. Get the first time in seconds and minutes. Get them to type in the alphabet and hit enter. Get the second time in seconds (and minutes). If they entered it correctly then. Subtract first time from the second time. Tell them how many seconds they took

Hint:

You'll need to store their attempt at the alphabet in a variable and compare with "abcdef.."

Challenge task:

Keep a record of the best time achieved.

Deal with upper or lower case letters entered.

Code:

```
GNU nano 2.2.6 File: Practical7ex5.py
print "\033[1;32mProgram Working\033[1;m"

import time # time function is imported

# instructions
begin = raw_input("""Start typing the alphabet after you hit enter.
It can be lower case or upper. Dont put a space between each letter
""")
# the timer starts when the user presses enter

start = int(time.time())
# timer started

type = raw_input("Start typing: ")
# user will type alphabet when the timer starts

if type == "abcdefghijklmnopqrstuvwxyz":
    print "You got it right. It took you", int(time.time()) - start, "seconds"
elif type == "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
    print "You got it right. It took you", int(time.time()) - start, "seconds"
# if statements made for upper case and lower case entries
else:
    print "The alphabet you entered is missing some letters or in the wrong order."
# if and else statements are produced, printing a response depending on what the user entered
# if the alphabet was typed in correctly, the program will let the user know they were correct
# it will also display the time it took by subtracting the int(time.time()) by the starts (0)
```

(Program and evaluation on the next page)

Program:

```
21223241@buildhost:~/UWLrepo/Week7$ python Practical7ex5.py
Program Working
Start typing the alphabet after you hit enter.
It can be lower case or upper. Dont put a space between each letter

Start typing: abcdefghijklmnopqrstuvwxyz
You got it right. It took you 13 seconds
21223241@buildhost:~/UWLrepo/Week7$ python Practical7ex5.py
Program Working
Start typing the alphabet after you hit enter.
It can be lower case or upper. Dont put a space between each letter

Start typing: ABCDEFGHIJKLMNOPQRSTUVWXYZ
You got it right. It took you 10 seconds
21223241@buildhost:~/UWLrepo/Week7$ python Practical7ex5.py
Program Working
Start typing the alphabet after you hit enter.
It can be lower case or upper. Dont put a space between each letter
SADSDASSADSADSASA
Start typing: DFSDFDSFD
The alphabet you entered is missing some letters or in the wrong order.
```

Evaluation:

This task was surprisingly easy to complete. This was only because it was similar to a task that had to be completed in one of the previous weeks. The only change that needed to be entered was adding an IF statement for the results. The IF statement would relate to what the user had entered, in this case the alphabet. I made three statements, one for lower case entry, upper case and wrong answer. As I only added to a previous task, I found no errors when completing this task.

Week 8

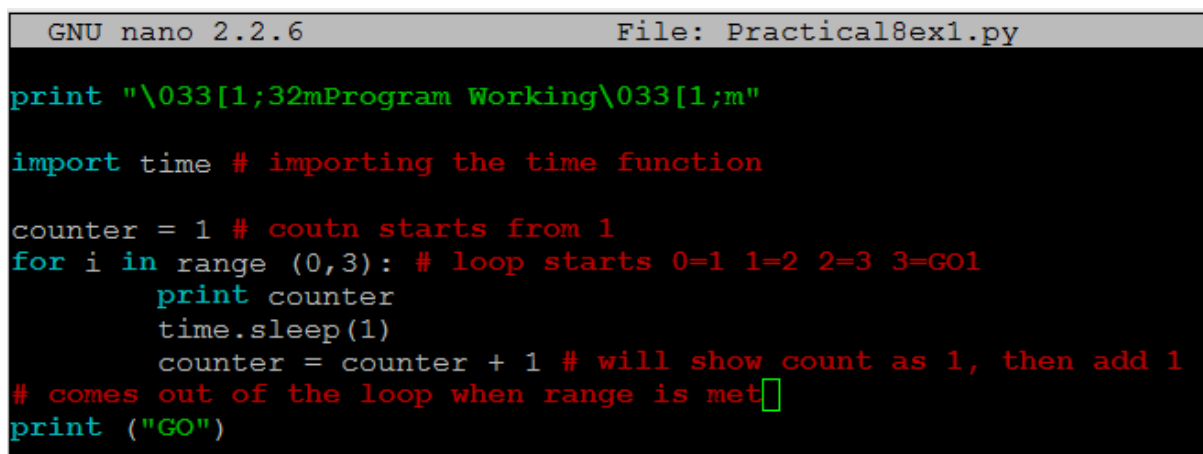
Exercise 1

Copy the code below into Python and see what it does:

```
for i in range(0,5):  
  
    print{'looping'}
```

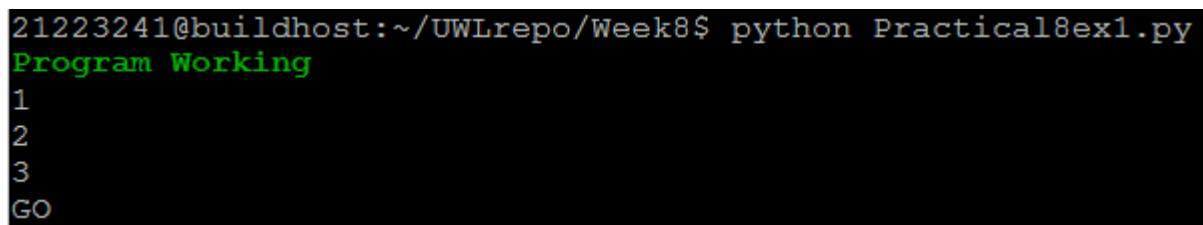
Edit the code above to write a program that counts to 3 with a 1 second pause in between the numbers and then say 'GO!'

Code:



```
GNU nano 2.2.6 File: Practical8ex1.py  
  
print "\033[1;32mProgram Working\033[1;m"  
  
import time # importing the time function  
  
counter = 1 # counn starts from 1  
for i in range (0,3): # loop starts 0=1 1=2 2=3 3=GO1  
    print counter  
    time.sleep(1)  
    counter = counter + 1 # will show count as 1, then add 1  
# comes out of the loop when range is met  
print ("GO")
```

Program:



```
21223241@buildhost:~/UWLrepo/Week8$ python Practical8ex1.py  
Program Working  
1  
2  
3  
GO
```

Evaluation:

This was the first task that needed a loop. This was quite a tricky task as I was not used to writing loops correctly. When I first attempted the task, many errors were made by me, like the other tasks, because of incorrect layout. I had figured out how to make my count go from 1 to GO. The problem I faced next was being able to put a pause between each count. One of my colleagues and I had found the answer online, and found it was quite simple. By importing the time function, we used the command `time.sleep(1)`. This meant sleep for 1 second before showing the next value, or number.

Exercise 2

Write a program that sets a password as “changeme” and asks the user to enter the password and keeps asking until the correct password is entered and then says “Accepted”. The program should count how many attempts the user has taken and tell them after they have been accepted.

Challenge task:

If the user takes more than 5 attempts the program should say, “Access denied, please press enter to exit and contact security to reset your password”

Code:

```
GNU nano 2.2.6 File: Practical8ex2.py

print "\033[1;32mProgram Working\033[1;m"

count = 1 # count starts at 1
defpass = "changeme" # the desired password is chnageme
attempt = raw_input("Please enter the default password ")
# usr asked to enter password

while attempt != defpass:
    attempt = raw_input("Please enter the default password ")
    count = count + 1
    if count > 4:
        break # count increases when pw is wrong, stops at 4
if attempt == defpass:
    print "Password Accepted, it took you " + str(count) + " attempts"
else:
    print "Access denied"
```

Program:

```
21223241@buildhost:~/UWLrepo/Week8$ python Practical8ex2.py
Program Working
Please enter the default password changeme
Password Accepted, it took you 1 attempts
21223241@buildhost:~/UWLrepo/Week8$ python Practical8ex2.py
Program Working
Please enter the default password hello
Please enter the default password no?
Please enter the default password changeme
Password Accepted, it took you 3 attempts
```

Evaluation:

This task required me to use loop and IF statements. Understanding this program was hard and lengthy. The idea of the loop was easy to understand. If the count exceeded 4, this would assume that the user has entered the wrong password and would receive the print from the else statement. If the user did type in the correct password, they would receive the print from the IF statement. This print string contained how many attempts it took to get to the correct password.

Exercise 3

A factorial means the number and all the preceding numbers multiplied together.

So 4 factorial (written 4!) = $1 \times 2 \times 3 \times 4$ And so on...

Write a program that will prompt for an integer and then calculate the factorial of that number. To test it works $8! > 40,320$

Code:

```
GNU nano 2.2.6 File: Practical8ex3.py

print "\033[1;32mProgram Working\033[1;m"

# user is asked to enter a number
number = int(raw_input("Enter any number: "))

fact = 1

while number: # loop
    fact = fact*number # the number entered will be timesed by 1
    number = number - 1 # 1 taken away from the entered number and looped

print "Your factorial for the number you enetered is", fact
# the factorial for the entered number is printed
```

Program:

```
21223241@buildhost:~/UWLrepo/Week8$ python Practical8ex3.py
Program Working
Enter any number: 5
Your factorial for the number you enetered is 120
21223241@buildhost:~/UWLrepo/Week8$ python Practical8ex3.py
Program Working
Enter any number: 8
Your factorial for the number you enetered is 40320
```

Evaluation:

Looking back at the code, I feel that it very interesting. The number entered would be multiplied by the fact variable. The number would then be subtracted by 1 and loop back to fact*number. When the calculation is over, the loop will end and display the answer. I did need to find help from online sources to complete it.

Exercise 4

Write a program to count the number of words in a sentence.

The user enters a sentence.

The program responds with the number of words in the sentence.

Hint:

Look for spaces and full stops in the string.

Challenge task:

Develop a program that will display a sentence backwards after entered.

Code:

```
GNU nano 2.2.6 File: Practical8ex4_2.py
print "\033[1;32mProgram Working\033[1;m"

sentence = [str(raw_input("Type in your sentence and I will count how many words there are: "))]
# the sentence variable has a raw_input in list format []

for item in sentence: # the loop starts here
    print len(item.split()) # the len function count the words in the string entered
    break

newsentence = raw_input("Enter another sentene and I will show you it in reverse: ") # instruction
print newsentence, " In reverse it is: ",newsentence[::-1] # ::-1 means count the string in reverse
# the reverse of the string entered is printed
```

Program:

```
21223241@buildhost:~/UWLrepo/Week8$ python Practical8ex4_2.py
Program Working
Type in your sentence and I will count how many words there are: Hello I made this program
5
Enter another sentene and I will show you it in reverse: hi there, this program rocks
hi there, this program rocks In reverse it is: skcor margorp siht ,ereht ih
```

Evaluation:

On this task, I needed the internet to complete it. I was not sure how to manually split the string that would be entered by the user. I found online the split command `len(item.split())` which takes the item entered, in this case the user input, and splits it according to the spaces in between the words.

For the challenge task, I added lines to the same program. I created another user input command; the sentence will be then be printed in reverse. I found the command to reverse the sentence online. The command needed for this was the variable followed by `[::-1]`. This causes the entered string to be read in reverse.

Exercise 5

Write a program for a Guess the Number game.

The computer selects a random number between 1 and 100. The user keeps guessing which number the computer has chosen until they get it right. The computer responds 'got it' or 'too high' or 'too low' after each guess. After the user has guessed the number the computer tells them how many attempts they have made.

Code:

```
GNU nano 2.2.6      File: Practical8ex5_2.py

print "\033[1;32mProgram Working\033[1;m"

import random # importing random function

ran = random.randint(1,10) # using the random function in a range (1-10)

guess = int(raw_input("Try guessing the number between 1-10: "))
# the user will guess the number

attempt = 1 # the attempt will start at 0
while guess:
    if guess > ran:
        print "That was higher than I wanted"
    if guess < ran:
        print "That was lower than what I wanted"
    # if statements letting the user know if its high or low
    guess = int(raw_input("Try guessing again "))
    # after getting it wrong, the user will need to guess again
    attempt = attempt + 1 # 1 will be added to attempt (1)
    if guess == ran:
        print "You got it right"
        break # loop breaks when the user gets it right
print "Well done it took you" , attempt, "attempt(s)"
```

Program:

```
21223241@buildhost:~/UWLrepo/Week8$ python Practical8ex5_2.py
Program Working
Try guessing the number between 1-10: 5
That was higher than I wanted
Try guessing again 4
That was higher than I wanted
Try guessing again 3
That was higher than I wanted
Try guessing again 2
You got it right
Well done it took you 4 attempt(s)
```

Evaluation:

I had to use the random function which needed to be imported. Similar to a previous task, the number had to be generated by random. This was done by entering random.randint. The user then had to enter a number. IF statements were then written inside a while loop. I made a mistake the first time I attempted the task when it came to writing the loop. Instead of writing "while guess:" I had written "While guess! =". With this fact, if the user was to guess correctly the first time, the loop will still be activated.

Week 9

Exercise 1

Write a function *length* that will calculate the length of a given list or string.

Comment:

Python has the `len()` function built in already, but writing it yourself is nevertheless a good exercise.

Code:

```
GNU nano 2.2.6 File: Practical9ex1.py

print "\033[1;32mProgram Working\033[1;m"

def countstring(mystring): # function created
    for item in mystring: # for loop
        print "Words in your sentence: ", len(item.split())
        # len function used to count the words in the string
        break

mystring = [raw_input("Enter a sentence an I will count how many words there are: ")]
# user entering a number

countstring(mystring) # calling on the function above
```

Program:

```
21223241@buildhost:~/UWLrepo/Week9$ python Practical9ex1.py
Program Working
Enter a sentence an I will count how many words there are: Hello out there
Words in your sentence: 3
```

Evaluation:

This task was similar to a previous task where the program would display how many words there are in a string. I took this code and put it in a function. The function was named “countstring”. There were a couple of things that needed to be added to that code. I needed to define the function, create a raw input outside the function, and call for the function. To call the function I needed to enter “countstring(mystring)”.

I found some confusion when writing this code. I was not sure what had to be in the brackets when calling the function. I entered the name of the variable “mystring” to be on the safe side.

Exercise 2

Write a function *histogram* that takes a list of integers and prints a histogram to the screen. For example, `histogram([4, 9, 7])` should print the following:

Code:

```
GNU nano 2.2.6 File: Practical9ex2.py

print "\033[1;32mProgram Working\033[1;m"

def histogram (a):# creating a function called histogram
    print ("*")*number,
    print "Amazing right?!"
# when the user enters a number, the funtion will return that many *

number = int(raw_input("Enter a number and see what happens! grrr: "))
# the user will enter a number they want
# the function is then called on
histogram (number)
```

Program:

```
21223241@buildhost:~/UWLrepo/Week9$ python Practical9ex2.py
Program Working
Enter a number and see what happens! grrr: 5
***** Amazing right?!
*****
```

Evaluation:

When attempting this task I remembered completing a task like this in a previous week. It required the name to be printed 3 times. This was done by having `print ("variable")*`. In this case the `*` would mean print the variable a certain number of times.

In this task the variable would be the entered number. The function will then called upon to repeat the `"*"` the amount of time that the user had entered.

I did not find much trouble when attempting this task as past weeks had helped, saying this I could not figure out how to output a list of number into a histogram.

Exercise 3

Write a function *leap_year* that takes a year as input and calculates whether it is a leap year or not.

Hint:

Leap years occur according to the following formula:

A leap year is divisible by four, but not by one hundred, unless it is divisible by four hundred.

For example; 1992, 1996 and 2000 are leap years, but 1993 and 1900 are not.

Result:

Please enter a year: 1996

1996 is a leap year.

Please enter a year: 1993

1993 is not a leap year.

Code:

```
GNU nano 2.2.6 File: Practical9ex3_3.py
print "\033[1;32mProgram Working\033[1;m"

def IsLeapYear(year): # function
    if((year % 4) == 0): # these are all the IF rules for the entered number
        if((year % 100) == 0): # these will check if the year entered can be divided by 4,100,400
            if( (year % 400) == 0):
                return 1
            else:
                return 0
        else:
            return 1
    return 0

Year = 0
print "Program to check Leap Year"
print "Enter Year: ",
Year = input()
if( IsLeapYear(Year) == 1): # if the returned value if 1, it is a leap year.
    print Year, "is a leap year"
else:
    print Year, "is NOT a leap year"
```

(Program and evaluation on next page)

Program:

```
21223241@buildhost:~/UWLrepo/Week9$ python Practical9ex3_3.py
Program Working
Program to check Leap Year
Enter Year: 1993
1993 is NOT a leap year
21223241@buildhost:~/UWLrepo/Week9$ python Practical9ex3_3.py
Program Working
Program to check Leap Year
Enter Year: 1996
1996 is a leap year
```

Evaluation:

This was one of the hardest tasks that I had to do. I had to search so many different websites to search a suitable method of calculate whether the date entered was in fact, a leap year. After many attempts, the above program is what I had come to. Out of all the tasks, this was the most difficult. Writing up the program initially, I didn't think this layout was possible or appropriate.

Exercise 4

Write a function *max* that takes two numbers as arguments and returns the largest of them as the result.

Hint:

Use control flow statements to test if a number is bigger or not.

Challenge task 1:

Write a function *max_of_three* that takes three numbers as arguments and returns the largest of them.

Challenge task 2:

The function *max* and the function *max_of_three* will only work for two and three numbers respectively. But if we have a much larger number of number or we do not know in advance how many they are, write a function *max_in_list* that takes a list of numbers and returns the largest one.

Code:

```
GNU nano 2.2.6 File: Practical9ex4_2.py
print "\033[1;32mProgram Working\033[1;m"

def maximum(a,b): # defining the function
    max2 = [a,b] # the two variables are put in a list
    print "The largest number out your entry was", max(max2)
    # max is the function which compares the values and gives the largest

print "Enter two number and I will give you the largest out of them."
n1 = float(raw_input("Enter your first number: ")) # user asked to enter 2 values
n2 = float(raw_input("Enter your second number: "))

maximum(n1,n2) # n1 n2 are put into the function. a=n1, b=n2

def maximum_3(q,r,s): #new function
    max3 = [q,r,s] # variables put in a list
    print "The largest out of the three numbers is", max(max3)
    # max function used again

print "Now we will see the largest number out of 3 entries"
number1 = float(raw_input("Enter a number: "))
number2 = float(raw_input("Enter your second number: "))
number3 = float(raw_input("Now your last number: "))
# user asked for 3 numbers this time

maximum_3(number1,number2,number3)
# the function is called for, number1=q, number2=r, number3=s
```

Program:

```
21223241@buildhost:~/UWLrepo/Week9$ python Practical9ex4_2.py
Program Working
Enter two number and I will give you the largest out of them.
Enter your first number: 3
Enter your second number: 4
The largest number out your entry was 4.0
Now we will see the largest number out of 3 entries
Enter a number: 2
Enter your second number: 3
Now your last number: 100
The largest out of the three numbers is 100.0
```

Evaluation:

This was easier than the previous task. I had to find out online what function was actually needed to display the largest number out of a list. When I had initially written the program, I did come across an error. To display the largest number with max, the numbers had to be in square brackets, rather than the round standard brackets.

Exercise 5

Write a function that will convert a score into a grade. The function will return 'A' -> 'U'.

The function will require a parameter to do its job: the mark

The formula for AS level is:

>=80% -> 'A'

>=70% -> 'B'

>=60% -> 'C' Etc.

Assume the maximum module mark is 100. Having written the function, we want to use it three times. Write a program with the function that allows the user to enter two modules AS scores and displays the grade. It then adds the two results together and displays the student's overall grade.

For example:

Enter Module 1 result: 78 Enter Module 2 result: 67

Result:

Module 1: B Module 2: C AS Level: B

Code:

```
GNU nano 2.2.6 File: Practical9ex5.py
print "\033[1;32mProgram Working\033[1;m"

def marktograde(a): # creating function for the first grade
    if a >= 80: # if statements for the score entered
        print "M1 grade: A"
    elif a >= 70:
        print "M1 grade: B"
    elif a >= 60:
        print "M1 grade: C"
    elif a >= 50:
        print "M1 grade: D"
    elif a >= 40:
        print "M1 grade: E"
    elif a <= 39:
        print "M1 grade: U"
mark = float(raw_input("Enter the percentage mark for module1: "))
marktograde (mark) # user input grade, the function is called upon, and will return grade

# this function is the same as the previous but for the second module grade
def marktograde2(b):
    if b >= 80:
        print "M2 grade: A"
    elif b >= 70:
        print "M2 grade: B"
    elif b >= 60:
        print "M2 grade: C"
    elif b >= 50:
        print "M2 grade: D"
    elif b >= 40:
        print "M2 grade: E"
    elif b <= 39:
        print "M2 grade: U"

mark2 = float(raw_input("Enter the second percentage mark for module2: "))
marktograde2(mark2)
```

```
def overall(c): # this function compares the two grades and gives average
    if c >= 80:
        print "Overall grade: A"
    elif c >= 70:
        print "Overall grade: B"
    elif c >= 60:
        print "Overall grade: C"
    elif c >= 50:
        print "Overall grade: D"
    elif c >= 40:
        print "Overall grade: E"
    elif c <= 39:
        print "Overall grade: U"

overallmark = (mark+mark2)/2 # working out average for the 2 scores
overall(overallmark) # the last function is called upon
```

Program:

```
21223241@buildhost:~/UWLrepo/Week9$ python Practical9ex5.py
Program Working
Enter the percentage mark for module1: 78
M1 grade: B
Enter the second percentage mark for module2: 54
M2 grade: D
Overall grade: C
```

Evaluation:

I found no issues when writing this program. The main difficulty of this task was writing up all of the if statements. I had written one IF statement then copied and pasted them to the next functions. I had changed the module names to cater to the module.

Week 10

Exercise 1

Write a class definition called *BankAccount*.

It should also have three attributes: name (a string), accountNumber (a string or an integer) and balance (a float).

It should have three methods: *makeWithdrawal*(self, amount) to withdraw money, *makeDeposit*(self, amount) to make a deposit and *getBalance*(self) to get the balance of the bank account.

Code:

```
GNU nano 2.2.6 File: Practical10ex1.py

print "\033[1;32mProgram Working\033[1;m"

class BankAccount(object): # making class
    def __init__(self, name, accountNumber, balance): # objects withing the class
        self.name_str = name # varaibles assocaites with itself
        self.accountNumber_int = accountNumber
        self.balance_float = balance

# the functions use the objects from the __init__
    def makeWithdrawal(self, amount):
        self.balance_float -= amount # functions using the objects

    def makeDeposit(self, amount):
        self.balance_float += amount

    def getBalance(self):
        return self.balance_float
```

Evaluation:

I had so much confusion going into this task. When reading the question, I had mistakenly thought it was too easy to do and there must a catch to it. I had overcomplicated the task and was concentrating on aspects that I did not to. I was worried about making the data entry, such as the name and making withdrawals, a part of the task. Completing ex2 from this week allowed me to understand how to write classes with objects.

Exercise 2

Write a program that takes three integers as input from a user and displays the sum, average, smallest and largest of the numbers.

Hint:

You need to create a class definition called *Calculations* with 4 methods: `sum(numbers[])`, `average(numbers[])`, `minimum(numbers[])` and `maximum(numbers[])`.

Code:

```
GNU nano 2.2.6 File: Practical10ex2.py

print "\033[1;32mProgram Working\033[1;m"

class Calculations(object): # Creating class
    def __init__(self, number1, number2, number3): # initialising class
        self.number1 = number1 # self.variables,
        self.number2 = number2 # ensuring variables are associated with the class
        self.number3 = number3
    def sum(self, number1, number2, number3): # creating sum function
        print "Your sum is: ", number1+number2+number3 # adding all numbers
    def average(self, number1, number2, number3): # creating average function
        print "Your average is: ", float((number1+number2+number3)/3)
    def maximum(self, number1, number2, number3): # function withing a function: max
        print "The largest number you entered was: ", max(number1, number2, number3)
    def minimum(self, number1, number2, number3): # function withing a function: min
        print "The smallest number you entered was: ", min(number1, number2, number3)

print "We are going to do some maths"

Calculations = Calculations() # calling on the class
number1 = int(raw_input("Please enter your first number: ")) # user inputs the 3 numbers
number2 = int(raw_input("Now enter your second number: "))
number3 = int(raw_input("Finally enter your first number: "))

Calculations.sum(number1, number2, number3) # class and the objects
Calculations.average(number1, number2, number3)
Calculations.maximum(number1, number2, number3)
Calculations.minimum(number1, number2, number3)
```

Program:

```
21223241@buildhost:~/UWLrepo/Week10$ python Practical10ex2.py
Program Working
We are going to do some maths
Please enter your first number: 45
Now enter your second number: 77
Finally enter your first number: 56
Your sum is: 178
Your average is: 59.0
The largest number you entered was: 77
The smallest number you entered was: 45
```

Evaluation:

I found this task easier than the previous. This could be because I did search online for a variety of examples of classes. I surprisingly had less trouble completing this task than the previous. I realised, while completing it, that all the functions were similar, where the differences came from the name of the function and the function is carried out. This took the edge off its difficulty.

Pushing Tasks to BitBucket

Week4



Hamza Bhatti committed 3 minutes ago ([raw commit](#))

Week4 practical exercises: All programs from the week are working correctly with all errors in the program being correct. Code that is ouputed will have a green line saying Program Working.

Comments (0)



Files changed (7)

+3	-1	Week4/Practical4ex1.py
+5	-4	Week4/Practical4ex2.py
+4	-2	Week4/Practical4ex3.py
+3	-1	Week4/Practical4ex4.py
+2	-0	Week4/Practical4ex5.py
+4	-2	Week4/Practical4ex6.py
+2	-0	Week4/Practical4ex7.py

Week5



Hamza Bhatti committed 4 minutes ago ([raw commit](#))

Week5 Practical exercises: All the programs from this week are working. All errors are removed. Each program has a green line saying Program Workings when executed

Comments (0)



Files changed (6)

+2	-0	Week5/Practical5ex1a.py
+2	-0	Week5/Practical5ex1b.py
+3	-0	Week5/Practical5ex2.py
+2	-0	Week5/Practical5ex3.py
+6	-4	Week5/Practical5ex4.py
+3	-1	Week5/Practical5ex5.py

Week6&7



Hamza Bhatti committed 3 minutes ago ([raw commit](#))

Weeks 6&7: Adding all of the practical tasks from the week 6/7 section. The programs withing this push are all working. All of the tasks have a Program Working printed in green when the program is executed

Comments (0)



Files changed (5)

+14	-0	Week7/Practical7ex1_2.py
+43	-0	Week7/Practical7ex2_test.py
+2	-0	Week7/Practical7ex3.py
+7	-10	Week7/Practical7ex4.py
+29	-0	Week7/Practical7ex5.py

Week8



Hamza Bhatti committed 4 minutes ago ([raw commit](#))

Week8: These are all of the week8 exercises. All of the tasks included work accordingly with all errors related to the functioning of the programs have been removed

Comments (0)



Files changed (5)

+3	-0	Week8/Practical8ex1_2.py
+2	-0	Week8/Practical8ex2.py
+3	-1	Week8/Practical8ex3.py
+16	-0	Week8/Practical8ex4_2.py
+24	-0	Week8/Practical8ex5_2.py

Week9



Hamza Bhatti committed 12 minutes ago ([raw commit](#))

Week9: Adding all of the week9 practical exercises. All of the tasks are working. When executed, a green line will appear saying the program is working.

Comments (0)



What do you want to say?

Files changed (5)

+15	-0	Week9/Practical9ex1.py
+11	-0	Week9/Practical9ex2.py
+21	-0	Week9/Practical9ex3_3.py
+27	-0	Week9/Practical9ex4_2.py
+53	-0	Week9/Practical9ex5.py

Week10



Hamza Bhatti committed 4 minutes ago ([raw commit](#))

Week10: Adding the two practical exercises from this week. The practical 1 exercise only contains a class with object. Ex2 is a full program. Both tasks contain no errors and work accordingly

Comments (0)



What do you want to say?

Files changed (2)

+16	-0	Week10/Practical10ex1.py
+26	-0	Week10/Practical10ex2.py

Full Evaluation of project

I feel that I have learnt a lot going through every one of the practical exercises throughout the weeks. The main thing that I have learnt is that, when starting a task, I thought that I can attempt them without going through the lecture material. I found out the hard way especially in week 10 that this is something I should not continue doing when it comes to programming. When going through the week 10 practicals, I went through the material again, making sure I understood the concepts. Doing this, I then went back to all the other practicals, changing the comments within them. This showed that I actually understood what was actually going on, and not making guess work.

Overall I feel that I did a good job throughout the week exercises. I found many hurdle on the way, but this showed that I could pull through and get, at least, the main task done.

Saying this, I do feel that I need a huge amount of practice doing the tasks again. I feel that I have not fully taken to grips of some concepts such as loops. I feel this may be because of the pace in which this was going at, along with other work.

Again, I have learnt that I need to have more practice and have more knowledge before attempting the tasks in the future.