Assessed Work
Term 1, 2017/18
Student Mark System

Summary
You are asked to write a Python 3 program that will read a CSV data file and output selected data in the forms of a bar chart and a scatterplot.

Data source
The CSV file holds data relating to student marks:

Column	Column heading	Description	Type
1	ID	Identifier unique to each student	integer
2	name	Given name for each student	string
3	module1	Module mark	integer
4	module2	Module mark	integer
5	module3	Module mark	integer
6	module4	Module mark	integer
7	module5	Module mark	integer
8	module6	Module mark	integer
9	module7	Module mark	integer
10	module8	Module mark	integer

Students are required to study six out of eight of the modules. Therefore, for any student, there will be six columns containing marks and two blank columns. For the avoidance of doubt, the names have been selected from an international list of commonly occurring names, the ID numbers have been randomly generated and the marks have been algorithmically generated using several random numbers.

Reading data
Your program should read data from a CSV file using the csv module, opening the CSV file appropriately [lecture 9].

Reading data from the file into the program will require type casting from strings to ints [lecture 3].

Using list processing techniques (including list slicing) [lecture 8] will allow you to store the data in a reasonably convenient way. It is suggested that you read the CSV file into a list. Each element in the list would represent an individual student. Each student record would contain the ID, student's given name and six tuples consisting of pairs of module name and mark, for instance:

```
[[21045, 'Brian ', ('module1', 56), ('module3', 69), ('module4', 86),
  ('module5', 51), ('module6', 57), ('module8', 81)],
 [21047, 'Ana ', ('module1', 27), ('module2', 49), ('module3', 40),
  ('module5', 70), ('module7', 89), ('module8', 43)],
 …]
```

Choosing data records for your scatter plot and bar chart

The scatter plot requires the choice of a module by name and the bar chart requires the choice of an ID number, using input from the keyboard [lectures 2 and 3].

Outputting the data

The output uses the mathplotlib module [lecture 9] to represent the data. The required parameters are strings, integers and lists [lectures 3, 7, 8].

Offering the user the choice of a plot or a chart or to stop

The user is allowed to set up another scatter plot or bar chart or to finish their session. This requires input from the user [lectures 2 & 3] and the use of iteration [lecture 6]. You will probably also need to use assignment with a conditional [lectures 4 & 5]

What does this program look like when it is running?

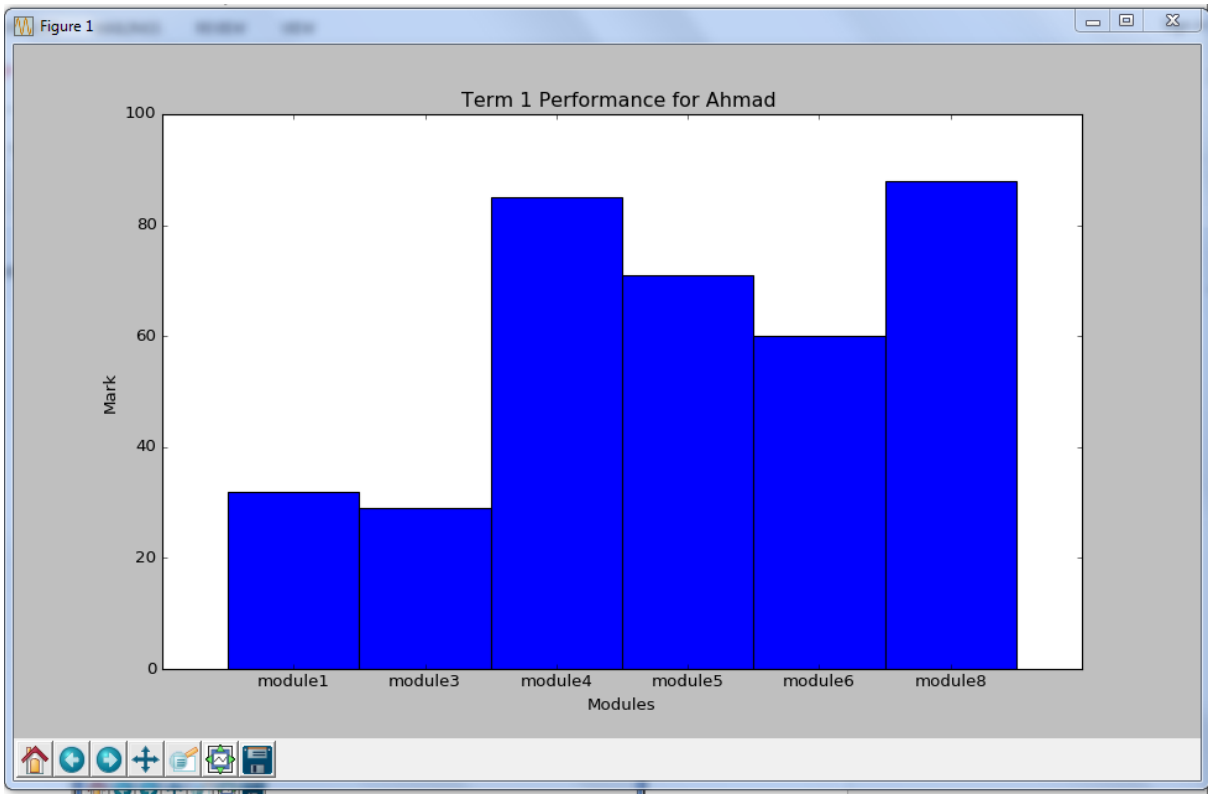
Here is an example of a program made earlier (with user input in bold):

```
Foundation Year Student Information System

Please choose one of:
  1 - display student's marks
  2 - display scatter plot of module's mark
  3 - exit the system
Your choice? 1

Please enter student ID number? 21075
```

At this point a bar chart was displayed:



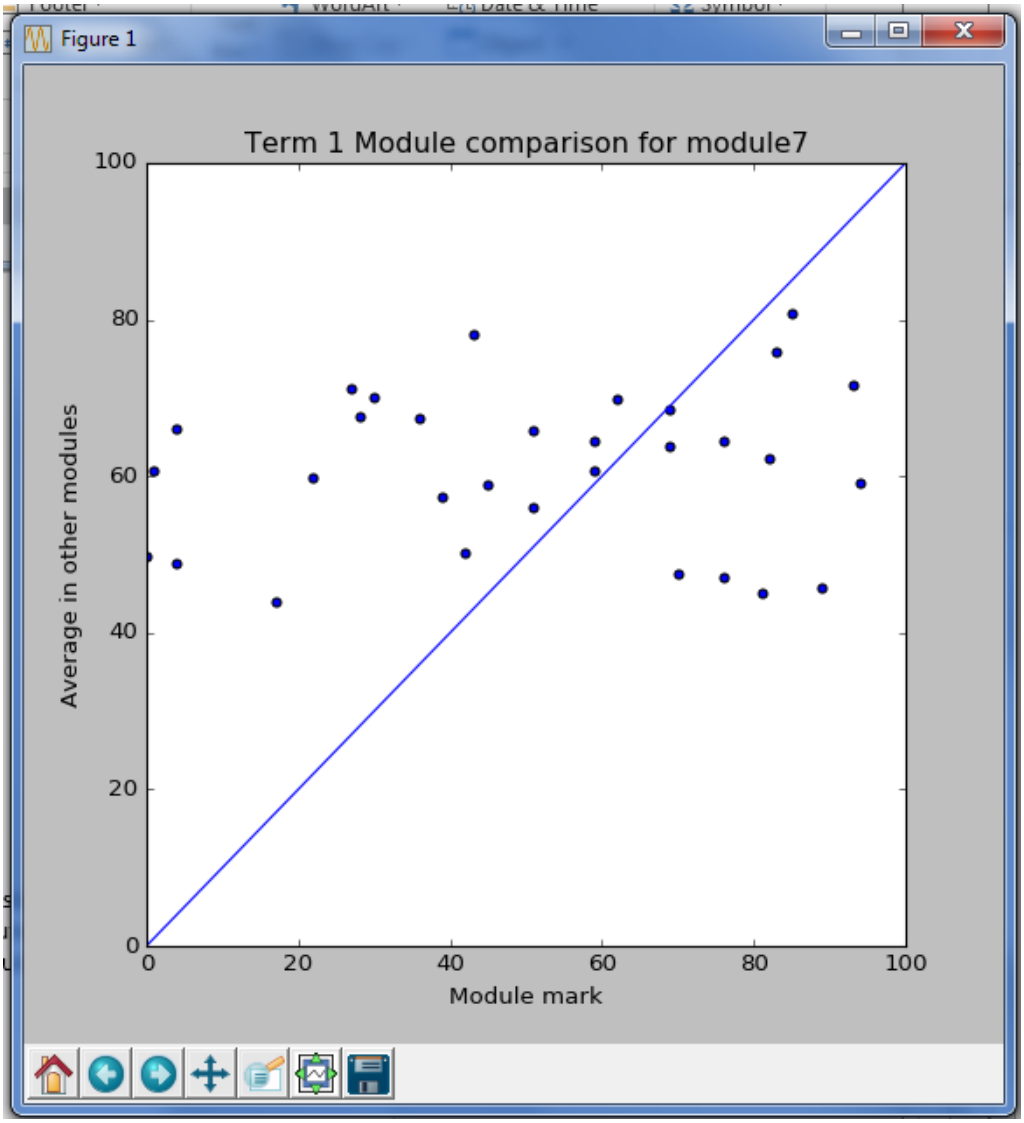
then processing resumed:

```
Foundation Year Student Information System

Please choose one of:
  1 - display student's marks
  2 - display scatter plot of module's mark
  3 - exit the system
Your choice? 2

Please enter module? module7
```

At this point a scatter plot was displayed:



then processing resumed:

```
Foundation Year Student Information System

Please choose one of:
  1 - display student's marks
  2 - display scatter plot of module's mark
  3 - exit the system
Your choice? 3
End
>>>
```

System requirements

The submitted code must be written in Python 3 in one of the 3.6.n, 3.5.n or 3.4.n versions. For the purposes of demonstration, the code must be runnable on the School of Computer Science's linux machines. If you develop your program on a machine outside the labs or on a web-hosted implementation of Python, you must transfer the program to the School machines and check that it runs before your mini-viva.

What the assessors are looking for

This is "summative" work. There is a mark which contributes 15% of the final mark for *Introductory Computer Science*. However, it is intended primarily to help you develop your skills.

The assessors are looking for a solution to the problem (ie a program that runs). There are marks for parts of the task so that, if you cannot complete all of the task, you should still demonstrate and submit the code you have developed.

A good solution should use appropriate constructs. (That means, for instance, that you use the more appropriate kind of loop if you have to use iteration.)

Output should be well presented – the use of English should be good and the layout consistent.

Duplicate code should be avoided.

10% of the mark for this exercise will be awarded for making the response to the user's input robust, for instance, to recognise invalid input for an ID number or a module name.

As university study is about becoming a self-sufficient learner capable of extending knowledge beyond lecture material, 10% of the mark for this exercise will be awarded for substantial use of a technique or technology not introduced in lectures 1 to 9. Two possible examples are the use of Python dictionaries (hash tables) to store student data instead of nested loops; use of a Tkinter Graphical User Interface instead of a text interface. (Be warned: Tkinter GUIs are difficult.)

Credit will be given for only one substantial extra technique – so a student using both dictionaries and a GUI would not get 20% (ie a possible total of 110%). The judgement of substantiality is reserved for the assessors and their judgement is final.

Lab times and additional support

You have four hours of lab times in which to have help with this work:

- Tuesday, 21st November 2017; 16:00-18:00
- Tuesday, 28th November 2017; 16:00-18:00

When the work is due

There will be mini-vivas on Tuesday 5th December 2017; 16:00-18:00. A timetable will be published nearer the time.

You should submit your program file by uploading to Canvas by 12 noon on Wednesday, 6th December 2017.

What this assignment contributes to the module mark

The mark contributes 15% of the final mark for *Introductory Computer Science*.

Feedback

Feedback will be given in the mini-viva and through a comment sheet uploaded to Canvas.

Plagiarism

All files will be scanned by a plagiarism checker that specialises in detecting plagiarism in programs. (It is particularly good at spotting where a program has been altered by changing names of variables and functions and moving a few lines around.) If plagiarism is suspected it will be investigated and, if appropriate, reported.

As a guide: you will want to discuss the work with friends. If you discuss the general way in which the problem can be solved, your work will not be considered plagiarised. If you swap specific lines of code with friends and include them in your program, the plagiarism checker is very likely to detect them.

If you are not intending to plagiarise, follow these guidelines and don't worry: deliberate plagiarism tends to look very different from high-level discussion.

Hints on producing a solution

Don't start by trying to write the whole program in one go. Divide the assignment into parts and encapsulate these in user-defined functions. Some of the obvious parts include:

1. *Reading data from the CSV file*
   Test that you can do this by opening a CSV file and printing each row

2. *Producing a bar chart*
   You might develop your skills by writing a function with the appropriate parameters that displays a bar chart. You can then use this in your final program.

3. *Producing a scatter plot*
   You might develop your skills by writing a function with the appropriate parameters that displays a scatter plot. You can then use this in your final program.

4. *Obtaining the user's input to choose which action is to be performed.*

Use the full two hour scheduled lab sessions and talk through your solution with the demonstrators.

Some extra programming items that might be useful:

Appending a tuple to a list

If you have two data items that are paired, then it is easier to keep them together in a tuple. For instance, if you were storing family and given names in a list, you might append the next name to the list like this:

```
names.append(("Nelson", "Mandela"))
```

Setting the size of a Matplotlib window
Plot sizes should be set before anything else, even before setting title, axis labels and creating the plot. An example of the command is:

```
plt.subplots(figsize=(12,7))
```

Formatting a bar chart
For a successful bar chart, you need to supply the number of bars required (using the range function); set the width of the columns and centre their labels; set the scale on the axes and insist that marks are shown from 0 to 100. Here is code that will do that:

```
x_count = range(len(ylist))
plt.bar(x_count, ylist, width=1, align="center")

# set ticks and scale on axes
plt.xticks(x_count, xlist)
plt.ylim(0, 100)
```

Adding a straight line to a scatterplot is easy
You simply plot the scatterplot and then the straight line graph:

```
# create scatter plot
plt.scatter(x_marks, y_marks)

# add x=y line
plt.plot([0,100],[0,100])
```

FINIS