Problem Set 2019

Programming and Scripting

Due: last commit on or before March 31st

This document contains the instructions for Problem Set 2019 for Programming and Scripting. The ten problems below should be completed by you during the semester. The lecturer will indicate which problems you should solve on different weeks. Your solutions will be assessed towards the end of the semester and will be worth 50% of your marks for this module.

The marking scheme is given below, and please note that one of the categories is for a pragmatic attitude to work, part of which involves working on the exercises as indicated by the lecturer during the weeks of the semester. It is not expected that you get every program right first time. So long as an attempt is made when indicated by the lecturer, this will count as a good approach. It is important that you keep working on any incomplete problems until the deadline.

Please note that all students are bound by the Quality Assurance Framework [?] at GMIT which includes the Code of Student Conduct and the Policy on Plagiarism. The onus is on the student to ensure they do not, even inadvertently, break the rules. A clean and comprehensive git [?] history (see below) is the best way to demonstrate that your submission is your own work. It is, however, expected that you draw on works that are not your own and you should systematically reference those works to enhance your submission.

How to submit

During the semester you will learn how to use the version control software git [?] to track your work. You will sync your work with GitHub [?] (you may use another provider if you wish) and provide the URL to your GitHub repository using the submission form on the Moodle page. You should consult the videos on the Moodle page for information on how to do this.

I suggest you consider making your repository publicly available (the default) so that prospective employers may view it. Should you wish to, you may make it private so long as you make it available to the lecturer. You can submit your URL at any time as git will track what work was done before the deadline and what work was completed afterwards. In general, only work completed before the deadline will count. However, in borderline cases or exceptional circumstances work done after the deadline may be considered, with or without penalty.

Marking scheme

This problem set will be worth 50% of your marks for this module. The following marking scheme will be used to mark your submission out of 100%. The examiner's overall impression of the assignment may influence marks in each individual component.

25%	**Research**	Investigation of each problem and its solution, as evidenced by clean, efficient solutions.
25%	**Development**	Clear, well-written, and efficient code with appropriate comments.
25%	**Consistency**	Good planning and pragmatic attitude to work as evidenced by commit history.
25%	**Documentation**	Concise descriptions of solutions.

Advice for students

- You must be able to explain your assignment during and after its completion. If you had trouble understanding something in the first place, you will likely have trouble explaining it a couple of weeks later. Write a short explanation of it into your submission, so that you can jog your memory later.
- Everyone is susceptible to procrastination and disorganisation. You are expected to be aware of this and take reasonable measures to avoid them.
- Students have problems from time to time. Some of these are unavoidable, such as acute family issues or illness. In such cases allowances regarding assignments can sometimes be made. Students should be able to show that up until an issue arose they had completed a reasonable and proportionate amount of work and took reasonable steps to avoid preventable issues.
- Go easy on yourself — this is one assignment in one module. It will not define you or your life. A higher overall course mark should not be determined by a single assignment.

Questions

1. Write a program that takes any positive integer and outputs the sum of all numbers between 1 and that number.

```
$ python sumupto.py
Please enter a positive integer: 10
55
```

2. Write a program that outputs whether or not today is a day that begins with the letter T. An example of running this program on a Thursday is as follows.

```
$ python begins-with-t.py
Yes - today begins with a T.
```

An example of running it on a Wednesday is as follows.

```
$ python begins-with-t.py
No - today does not begin with a T.
```

3. Write a program that prints all numbers between 1,000 and 10,000 that are divisible by 6 but not 12.

```
$ python divisors.py
1002
1014
1026
etc
9990
```

4. Write a program that asks the user ot input any positive integer and outputs the successive values of the following calculation performed on it up to and including when the value becomes 1. The calculation is: if the number is even, divide it by two, if it is odd then multiply it by three and add one.

```
$ python collatz.py
Please enter a positive integer: 10
10 5 16 8 4 2 1
```

5. Write a program that asks the user to input a positive integer and tells the user whether or not the number is a prime.

```
$ python primes.py
Please enter a positive integer: 19
That is a prime.
```

6. Write a program that takes a user input string and outputs every second word.

```
$ python secondstring.py
Please enter a sentence: The quick brown fox jumps over the lazy dog.
The brown jumps the dog
```

7. Write a program that that takes a positive floating point number as input and outputs an approximation of its square root.

```
$ python squareroot.py
Please enter a positive number: 14.5
The square root of 14.5 is approx. 3.8.
```

8. Write a program that outputs today's date and time in the format "Monday, January 10th 2019 at 1:15pm".

```
$ python datetime.py
Monday, January 10th 2019 at 1:15pm
```

9. Write a program that reads in a text file and outputs every second line.

10. Write a program that displays a plot of the functions x, x^2 and 2^x in the range $[0, 4]$.