FIT1045 Algorithmic Problem Solving – Workshop 1.

Objectives

The **objectives of this workshop** are:

- To become familiar with the concept of peer assisted learning and active participation in the classroom
- To understand how workshops are assessed in FIT1045
- To get familiar with the python interpreter.
- To be able to execute a file using IDLE.
- To be able to perform basic string and numerical manipulation.
- To be able to import from the math and random packages.
- To be able to do simple input/output.

Assessment

Active participation and completion As part of this unit, workshops are **assessed** with every workshop contributing 1% to your final unit mark (for a total of 12%) with each workshop being marked out of 1.
Simply participating in the classroom and working with your peers is sufficient to ensure that you get at least 0.5 marks for each workshop; If you complete a majority of the activities in a workshop you will get 0.75 marks and completing all of the tasks (aside from extension tasks) will net you 1 mark.
If you are **not** active in the class room you will be marked based on what **proportion** of the tasks you have completed (and understood).

- Completing **some** of the tasks in the workshop: **(0.25)**
- **Active participation** in the classroom or completing about **half** of the tasks in the workshop: **(0.5)**
- Completing **most** of the tasks in the workshop: **(0.75)**
- Completing **all** of the tasks in the workshop: **(1)**

We have made this decision as we want to ensure that all students recognise that any effort to work cooperatively and develop your python skills has value while still incentivising students to complete the tasks given.
We want to ensure every student has the opportunity to develop an understanding and proficiency in programming.
If you have been interacting with your demonstrator and your peers, you have already demonstrated an understanding of what you have produced; however if you have been inactive, we have to determine your level of understanding by interview.

Expectations

Participation in these workshops should be considered the **absolute bare minimum** required effort to **pass** this unit, however is by **no** means the **recommended** level of effort if you wish to excel in this unit, your future units and the workplace.

In a university setting, simply completing assigned work is not sufficient to maximise your learning outcomes, you will need to practice some self-learning as well. This is true in all units and we assist this by providing guidance on where to start your search with suggested readings and video links.

Programming is a skill and an art-form and **cannot** be done with any degree of quality **without regular and consistent** practice; You should use your non-contact hours with equal emphasis on discussion and review of the theory and time spent practicing coding.

The lecture code is a good starting point; You can also practice coding with the interactive tasks available through the codecademy python course `https://www.codecademy.com/learn/python`; Your demonstrator will be able to suggest more practice tasks for you if you request them.

Keeping a record

AKA: how not to lose your work and avoid stress and wasted time later

You will produce a lot of code as part of this unit (and your entire degree and future occupation). As such, it is ideal to avoid 're-inventing the wheel' each week. If you have produced code to do a task, there is no need for you to write it up again. You should write it up once to a very high standard and then reuse it any time you need to do the same thing.
In addition, you don't ever want to end up in the situation where you write up some code and (on the day of submission) your laptop battery dies, or you lose your flash drive, etc.
So how can we do this? How do we avoid rewriting things from scratch and make sure we never lose our work? Here are two options for what you can do!

The 'good enough' option: Google drive

As part of your Monash Student account, you have access to many google apps such as google drive. This allows you to store your work on the cloud and access it again from any machine where you can log into `my.monash`. At the very least this will allow you to keep your work in a central location and not need to worry about half a dozen different versions of the same code (and not knowing which is which).

The ideal option: a Git repository

In the future, when you get out into **industry** a **skill** which will be **expected** of you is the ability to work with actual version control systems such as **git**, this makes it a great idea to get into the habit of this early on. There is a bit of a learning curve in figuring out git but if you put in the effort now, you will reap the rewards later on.

Why use git?

Some of the great things about a real version control system like Git (or Subversion, Mercurial, etc.):

1. specifically designed for code (works well for most programming languages as well as simple files like html, css, csvs, txt, and tex files)
2. you can see easily see changes between versions of files
3. keep source files in a single common location
4. you can revert back to an older version if you prefer (and undo the reversion as well!)
5. if you modify a set of files at once (eg. where one 'talks to' another) the version control system tracks those changes as a group
6. multiple people can modify the same set of files simultaneously and have their edits combined in a non-crazy way

how to use git

Git and other version control systems can of course be done via command line, however these days you don't have to! A suggested option is to sign up for a **free** and **private** git repository via `https://bitbucket.org/` and manage changes with the software package 'gitKraken'. You don't have to do this, but if you do choose to go down the version control route you should ensure your repository is private (otherwise anyone can find your work online (which is an issue from a plagiarism perspective)). Bitbucket will allow you to have a repository of up to 1GB (which is more than enough outside serious graphics stuff)

Our friends at ENG1003 have written up support activity for learning to use git with the aid of gitKraken which we have adapted for this unit `https://www.alexandriarepository.org/module/using-git-with-gitkraken/`
Note: reusing your work will become even easier once we cover decomposition as you can always write a function for what you're doing and in later code just import that function to reuse it!

Useful Material

Introduction to Numbers and Python: `https://docs.python.org/3.0/tutorial/introduction.html`

Getting Started: This website contains a guide for installing and running python and IDLE under windows. `http://usingpython.com/running-a-python-program/`. You do not need to install python on the univeristy computers.

Things to try before your workshop.

Python, being an interpreted language has an interactive shell. This allows us to quickly and easily test small snippets of code from within a python environment without having to worry about complicated files. For simplicity we will be using an online python shell.[1] Navigate to `http://www.pythontutor.com/visualize.html`.

Task 0: A Simple Calculator

Using your keyboard, enter numbers and operators (* + - /) into the shell using the return key to execute these simple calculations.

Once you are comfortable with this you may want to use attempt to use some of the operators presented in task 1(a).

Task 1: Active participation roleplay activity

Your demonstrator will go through each of the following situations and ask you to first discuss your thoughts with the person next to you and then contribute to a class discussion.
For each of the following situations, you should consider how you can be an **active** member of the classroom and **develop** your own understanding (and that of those around you).

Situation 1:

The person next to you seems to be getting frustrated; You ask them what's wrong and they tell you they are having trouble implementing something. This happens to be something you implemented yourself and understand fairly well.

Situation 2:

You are looking at a task and aren't sure what it involves or even how to start; The person next to you is quietly working on their own code.

Situation 3:

Both you and the person next to you have just completed one of the tasks and you think your code is correct.

It is our intention that after completion of this task you feel empowered to be more effective at working with other students and being an active member of the classroom.

Task 2: Numerical Operations

Using the python interactive shell that is local to your computer, compute the first column in Table 1. Compute the other columns by creating a program that takes as input x and y and performs the operation. What does each of these operations do?

[1]Your workshop demonstrator will go through how to run a local shell in your workshop

Operator	x=3
	y=2
		x=4
		y=4
			x=5
			y=6
				Operation

$x+y$				Adds x and y together
$x-y$				
$x*y$				
$x**y$				
$x\%y$				
x/y				
$x//y$				
$x>y$				
$\cos(x/y)$				

Table 1: Numerical Operators

Task 3: Temperature Conversion

Write a program that converts the temperature in Fahrenheit to the temperature in Celsius. Your program should prompt the user for the temperature and then print "The temperature is XXX degrees Celsius".

For example:
Give the temperature in Fahrenheit? 100
The temperature is 37.7777777778 degrees Celsius.

NOTE: The conversion from F degrees Fahrenheit to C degrees Celsius is: $C = (F - 32) \times 5/9$.

Task 4: Finding The n^{th} Root

Write a program that when given as input from the user a number x and value n will find the n^{th} root of x. To make this task easier you will be requied to import pythons math library, see `https://docs.python.org/3/library/math.html`.

Background: The n^{th} root of a number x can be determined by using the natural logarithm and exponents. Let a be the n^{th} root of x, this gives $x = a^n$. Taking the logarithm of x gives,

$$\log(x) = \log(a^n) \tag{1}$$

We know from our log laws[2] that $\log(a^n) = n\log(a)$. If we divide through (1) by n gives

$$\frac{\log(x)}{n} = \log(a). \tag{2}$$

We then solve (2) for a:

$$e^{\frac{\log(x)}{n}} = e^{\log(a)} = a.$$

Example:

```
Please enter a number: 390625
Please enter a value for n: 8
The 8th root of 390625 is 4.999999999999999.
```

[2]`https://en.wikipedia.org/wiki/List_of_logarithmic_identities`

Task 5: Flipping Coins

The goal of this task will be to simulate random coin tosses for both a biased and unbiased coin. To complete this task we will need to look at importing a second library, just like we did the math library. The library we will be importing is the random library allowing the programmer to access tools to generate pseudo-random number and perform operations that use these tools. See `https://docs.python.org/3.5/library/random.html`.

Task 5(a)

Write a program that generates and prints some random numbers using the 'random.random()' command. What values are generated by this command?

Note: What happens when you type 'random.random(' in Idle. Discuss with your programming partner what this information means. Ask you workshop tutor if you are not sure.

Using Selection

Lecture 3 introduces selection. Before Workshop 2, modify your program in **Task 5(a)** and using selection complete Task **(5b)** and **(5c)**.

Task 5(b)

Write a program that simulates an unbiased coin flip. Your program should print true if the coin flip results in a head and false if the coin flip results in a tail. [3]

Task 5(c)

Now consider a biased coin. Write a program that takes a value p, with range between 0 and 1, as input from the user and tests a number of coin flips where p is the probability of the flip resulting in a result of heads.

Example

```
What kind of bias do your coins have? 0.5
Coin flip 1 has a value of heads: True
Coin flip 2 has a value of heads: False
Coin flip 3 has a value of heads: True
```

Note: This program may be made easier using the 'random.randrange(a)' function, if time permits, have a look at this function using the link above. We will be meeting the 'range(a)' function in detail next week.

Task 6: Strings

Write a program that takes as input a string representing a user's name. Your program should output the length of the name and the number of times each vowel occurs in it.

Extension Questions

All questions contained here either extend on the current week or look into topics that will be introduced in depth in the future. They may require a certain level of self study to complete during thier prescribed lab. If you have completed all other work, we highly recommend that you have go at these questions. **These problems as they appear here are not examinable. Any examinable content will be covered in the main section of later workshops.**

[3]For a problem with two possible outcomes, A and B, and a random number x we say that x belongs to outcome A if the $x < P_A$ and to outcome B otherwise.

Task 6: Strings? Or just letters?

In this task we will look at the relationship between strings and characters. We will be performing an operation known as *indexing*.

A string may be indexed by placing square brackets after the string with a value, the index (”hello”[0]). Write a program where you change the index given to the string and attempt to predict the output. Did certain values produce errors where others didn't? What about negative numbers?

Task 7: Three Sided Coins

Extend your program to flip coins that have 3 sides: heads, tails and other.

Hint: You can use numbers to represent the states rather than booleans.

```
Flipped a 3 sided coin which landed on side: 1
Flipped a 3 sided coin which landed on side: 3
Flipped a 3 sided coin which landed on side: 1
Flipped a 3 sided coin which landed on side: 2
```

Task 8: Escaping the Earth

Warning: Physics

The escape velovity of an object v_e is the velocity required to prevent the object falling back to the ground under the influence of earths gravity. We define the escape velocity such that it is the minimum required velocity. As such the object should be at rest once it has left the influnce of the earths gravity. To do this we will use two quantities, the kinetic energy,

$$E_k = \frac{1}{2} m_o v^2, \tag{3}$$

and the gravitiational potential energy,

$$U = \frac{-GM_E m_o}{r}. \tag{4}$$

From the conservation of energy, we know that the total energy before is equal to the total energy after and if the object is to remain at rest and not fall back to the surface of the earth will have a final velocity of 0.

$$(E_k + U)_b = (E_k + U)_a \tag{5}$$
$$(E_k + U)_b = 0 \tag{6}$$

By substituting (1) and (2) into (4) we can solve for v, our escape velocity.

Write a program to calculate the escape velocity of an apple using the following values.

- $G = 6.67408 \times 10^{-11}$, the universal gravitational constant.
- $M_E = 5.972 \times 10^{24}$, the mass of the earth in kilograms.
- $r = 6.371 \times 10^6$, the radius of the earth in meters.
- $m_o = 0.1$, the mass of an apple in kilograms.