# CSC2410: Assignment 1 (20%)

## Academic misconduct

Please ensure that you do not share any information about your assignment with anyone. Do not accidentally leave printouts or USB drives that contain your assignment details where someone else can access them. Make sure not to store your assignments on a computer where another student can access your assignment.

Academic misconduct is unacceptable and includes plagiarism, collusion and cheating. You will find further explanation at the link below: <https://www.usq.edu.au/library/referencing/plagiarism>.

* This assignment must be all your own work
* The source of all information must be correctly cited and referenced

Assignment that do not adhere to these requirements have the potential to be deemed to be the result of academic misconduct. Please make sure to read the USQ policies.

Should you have any queries regarding this assignment, please do not hesitate to email the course examiner.

**Assessment policy**

Please ensure to check the USQ assessment procedure <http://policy.usq.edu.au/documents/14749PL>.

**Late submission of Assignments**

Students can apply for an extension of time to submit an Assignment at any time up to the deadline. Students are strongly advised to make a request for an extension as soon as their need becomes apparent. Delay in making a request involves the risk that there will be insufficient time (with consequential loss of Marks) if the request is refused.

Please email the course examiner if you require an assignment extension. Any requests for extension must follow the USQ policies. Further information is available at these links:

* How do I apply for an assignment extension: <https://usqassist.custhelp.com/app/answers/detail/a_id/2950>
* What are the penalties for late assignment submission?  
  <https://usqassist.custhelp.com/app/answers/detail/a_id/5955>

# CSC2410: Assignment 1: Part A (40/100 marks)

Assignment 1 has two parts. Part A focuses on concepts introduced in the course readings. Part B focuses on implementation of modelling and programming concepts covered in tutorials.

For Assignment 1 you will submit two files:

* Part A: submit a **Word** (.doc, .docx) or **PDF** file which includes your essays for questions A1-A4 below.
* Part B: complete and submit the A1-TooBike.ipynb file provided.

## Modelling & Simulation concepts

### A1: Generalisation (8 marks)

What is **generalisation**? How can it improve a functions usability? Write a short essay answering this question. Provide examples to illustrate your points.

Generalization is the process of adding parameters to a function, parameters being variables whose values are set when the function is called. The point is that they are set outside of the function, not inside it.

The first advantage is that we can use the same function with different arguments without doing any extra coding[[1]](#footnote-1). In other words if a function is designed to model the amount of damage a car does when it crashes into another car, we can call the function multiple times increasing the speed with each call to get different results as the below example demonstrates:

def crashDamage(speed, mass)

crashDamage(40,1000)

crashDamage(60, 1000)

In the above example we have defined a function that uses speed and mass as parameters, and then called that function to test the different output at 40 and 60 km/ph respectively. The function has been generalized as it allows a more broad range of outputs, and is therefore more general than if speed and mass had been defined within the function itself[[2]](#footnote-2).

The second way generalization can improve usability is that the code base of the simulation can have a section at the top of the code that pre-defines the parameters. This in turn simplifies testing different scenarios as it means you can make changes to the parameters in one place and effect all the code. If these values had been hard coded into the function then changing them would require going and finding every instance of that value and replacing it with the new value.

For example, if there were a function designed to calculate Force=mass x acceleration then we could define force, mass and acceleration as parameters so that when mass and acceleration changed we could change those parameters at the top of the code as follows:

Force = 0

Mass = 15

Acceleration = 5

def forceCalculator(force,mass,acceleration):

force = mass \* acceleraton

return force

We could then call the function with:

forceCalculator(Force,Mass,Acceleration)

This would return the force with whatever changes we wanted to make to the Mass and Acceleration variables at the top of the code.

In conclusion, generalization involves adding parameters to a function that thereby make that function easier to use for modified inputs. Because the functions values are not defined with hard coding, they are considered general.

### A2: Abstraction (8 marks)

Write a short essay discussing the relevance of **abstraction** when trying to model and simulate a physical system. Provide examples to illustrate your points

Abstraction is the process of removing details from a system to create a model. In the real world there are so many details that impact on a system that taking them all into account is often impossible[[3]](#footnote-3). Instead, we can accept a certain margin of error as noise and create a model that works well enough. This sentiment is best expressed with the following aphorism: “All models are wrong, but some are useful”[[4]](#footnote-4). In other words: we know that we cannot take every little thing relevant to a system into account but we can still make models that offer useful predictions in spite of their limitations in perfectly describing reality.

To illustrate this point we can use the example of trying to model a length of time a plane’s flight will take. In the real physical system a lot of factors may play into this such as air pressure, weather, engine power and wear. We can take into account a plethora of details about the pilot such as mood, experience and even propensity to be distracted. Furthermore we could even have factors at play such as the logistics and administration behind the airport. We could get very precise about things and take into account the individual weights of customers with their clothing and luggage. There is almost no limit to the details that may be involved.

The question then becomes: how accurate does our model need to be to be useful? How much of the aforementioned complexity can we do without? What details can we be rid of to save time and effort? One way that we could apply abstraction to the plane flight times is by averaging flight times between destinations and using that as our model to predict flight times. It does not take into account details, but it may offer quite accurate and useful predictions. We would be abstracting all of those details into an average flight time.

In conclusion, abstraction is the process of simplifying a physical system to efficiently create a useful model.

### A3: Iterative modelling (8 marks)

Write a short essay discussing how **iterative modelling** is used to model a physical system. Provide examples to illustrate your points.

Iterative modelling is essentially about building a model that starts very simple and with a high level of abstraction, then improving that model incrementally with more advanced features[[5]](#footnote-5). Another way of wording this is that we aim to make the model less wrong with each revision, including more factors into consideration to make predictions or improving the underlying mathematical model.

To illustrate this point take the example of a system that calculates a jets maximum speed in the air. We could start with something basic such as thrust of the jet and use that to devise a speed as our first iteration of the model:

maxSpeed = thrust

We could then get more specific by taking into account wind resistance for our second iteration:

maxSpeed = thrust - windResistance

We could also begin adding details about the ability of the pilot, the pilot ability coefficient for the third iteration:

maxSpeed = (thrust – windResistance) \* pilotAbilityCoefficient

We could continue this trend of adding more and more detail to the model, making the model more accurate. We then test the model to see if it works better than the previous one. With each iteration the model becomes more complex, accurate and less wrong. In other words, it more closely resembles the physical system it is attempting to model. This is the essence of iterative modelling.

In conclusion, iterative modelling is about starting simply and getting more complex so that a model becomes more accurate with every revision.

### A4: Incremental development and scaffolding (8 marks)

What are the fundamental steps of **incremental development**? What part does **scaffolding** play in incremental development? Write a short essay answering these questions. Provide examples to illustrate your points.

Incremental development is the process of developing a program starting with a working program and making small changes that are then tested. It differs itself from iterative modelling in that it is about adding new features to code rather than necessarily adding new details to the modelling techniques, though it could include that. It has three fundamental steps as follows[[6]](#footnote-6):

1) Always start with a working program.

2) Make one small testable change at a time.

3) Run the program to see if the change worked.

The first step, always starting with a working program, is designed to start from something that is functioning. The benefit of this is that you start with a working codebase so it is a given that the program was working before the change was made.

The second step, make one small testable change at a time, is designed to make debugging and error finding easier. If we make a small change and test before making another we can spot the error immediately, whereas if we make a large change finding the exact source of the error requires going over all the changes.

The third step, run the program to see if the change worked, is essentially the testing phase. We run the program to test if the change worked as expected. If it has not worked as expected we can go back and try to fix the issue before further changes are made.

Scaffolding is writing code to test a change before another change that it may depend on is added to the system[[7]](#footnote-7). This applies in cases where you may have an expected input but not have the part of the program that generates that input properly.

If we use the example of an airplane flying we could create a model that takes into account real-time weather data by fetching it from the Bureau of Meteorology, but that system may not yet be up and running. As such we will have to build a fake version that outputs reasonable mock data from the Bureau of Meteorology. The benefit being that we test that the system handles the code properly before adding more.

In conclusion, incremental development is a process designed to decrease the difficulty of debugging by starting with a working program, creating a small change and then testing.

### References:

Downey, A. (2017). Modeling and Simulation in Python. 2nd ed. Needham, Massachusetts: Green Tea Press.

1. (Downey, 2017, p. 4) [↑](#footnote-ref-1)
2. (Downey, 2017, p. 4) [↑](#footnote-ref-2)
3. (Downey, 2017, p. 1) [↑](#footnote-ref-3)
4. (George Box cited in Downey, 2017, p. 4) [↑](#footnote-ref-4)
5. (Downey, 2017, p. 3) [↑](#footnote-ref-5)
6. (Downey, 2017, p. 32) [↑](#footnote-ref-6)
7. (Downey, 2017, p. 33) [↑](#footnote-ref-7)