Programming Principles
Assignment 1 Work Plan

The workload and pace of this unit can be quite challenging, particularly if this is your first exposure to university-level study. This document presents some broad tips on how to approach your first assignment in a way that will benefit your learning and align with the unit's expectations, as well as providing a "timeline" of what you can do to work on your assignment each week of semester.

How to Approach Assignment 1

Focus on Learning

One of the biggest mistakes that people make in units like this is focusing on assignments, marks or "deliverables" (i.e. things you need to submit). Always remember that your assessments are there for you to *demonstrate* your understanding and ability. They should *not* be the main point where you are *developing* your understanding and ability. The learning process in this unit involves:

1. Lectures to *introduce* concepts, giving you some *knowledge* about them.
2. Workshops to practise *applying* the concepts, for you to *develop* understanding and ability.
3. Supporting resources (textbook, readings, etc.) to *deepen* your understanding and ability.
4. Assessments for you to *demonstrate* your understanding and ability.

Number 2 (and 3) are critical! They are where the learning happens. Assignments are big and challenging tasks, so they *help to deepen* your understanding and ability, but they should mainly be a *demonstration* of the understanding and ability you've *already developed*. If you are struggling with your assignment, revise the relevant part of the unit content: learn the concepts, *then* apply them.

It also helps to remember that there is *not* "one correct solution" to an assignment, and that what we are testing is *your ability to solve the problem*, not your ability to submit a perfect solution. Always remember that individual assignments should reflect *your* understanding and ability.

Review the Module 5 Case Study

After covering the fundamental concepts of programming in the first four modules, Module 5 does not introduce any new content. Instead it is dedicated to a recap and consolidation of the first four modules, and a case study where we (as a class) work through a significantly-sized task/problem.

The case study is particularly relevant, since it illustrates the process of designing and implementing a program to solve a significantly-sized task/problem – i.e. *exactly what your assignment requires*. Watch the recording of Lecture 5 and write the code alongside the video. Pay attention to the problem solving process and how the concepts we've covered (selection, iteration, variables, etc.) are used and combined to first design a solution in pseudocode and then write code to implement it.

The case study also provides a significantly-sized example of *writing good pseudocode*.

Seek Information, Help and Feedback

It's up to you to design and implement the program, but that doesn't mean you can't ask for help. Check the Blackboard discussion boards at least once a week, since extra information, guides and tips are regularly posted. If you're struggling or just want feedback on your work, email your tutor.

Assignment Work Timeline

Week 1

This module covered lots of introductory and background material, but we did introduce Python and some coding concepts. Focus on installing and familiarising yourself with your coding environment:

- Have Python installed on your computer/laptop and know how to open IDLE.
- Understand how to create, save and run a Python program in IDLE.
- Understand how to open and edit a Python program in IDLE.
- Understand the difference between the Python Shell and the Python Editor.

It is also a good idea to keep all of the code and pseudocode that you write nice and organised. Whether you are storing the files on your computer, on a thumb drive, or in online cloud storage, it really helps to have a few folders to keep things organised and be sure to give all of your programs meaningful filenames so that you can find them later on. Keep a copy of every workshop program.

Lastly, *read through the entire assignment brief.* It might seem daunting at the moment, but rest assured we will cover everything it asks for by the end of Module 4. The sooner you read the brief, the sooner you can start taking notes on things that sound particularly relevant as we cover them.

Week 2

This module covered data types and selection statements. Combined with the concepts of variables, and using built-in functions such as `print()`, `input()` and `float()` from Module 1, you can now start to work on sections of the assignment.

Look through the requirements in the brief and find parts that look like they only/mainly involve things that we've covered. Remember that *you don't need to start from Requirement 1* – you can *tackle small parts of it in any order*, bringing them together as you go.

For example, here are a couple of requirements from an assignment:

> **2.2.1.** Prompt the user to input the mark of a unit and convert the input to an integer.
>
> **2.2.2.** If the mark is below 0, set it to 0. If the mark is above 100, set it to 100.

These two requirements were inside a loop which was inside another loop, but we don't need to worry about that to complete those requirements. You can write pseudocode and a corresponding program that just does this part for now, without having covered how loops work.

Don't think of assignments as one big difficult task, think of them as collections of small simple tasks!

Lastly, try to write a first draft of your pseudocode for the entire assignment program. Look ahead to the Lecture 5 recording as mentioned on the previous page and use that as an example. Base your program design upon the steps and structure of the assignment requirements. You'll definitely revise the pseudocode later, but it is good to make a start on it early. You can even highlight parts that you've completed as you go to help keep track of your progress.

Module 3

This module covered data structures and iteration (i.e. loops). At this stage, we have covered all of the concepts needed to design and implement the program that the assignment describes:

- Getting input via the `input()` function and displaying output using the `print()` function.
- Using variables to store and refer to values throughout the program, underpinned by the concepts of data types and data structures.
- Selection statements to run blocks of code based upon conditions.
- Iteration statements to repeat blocks of code based upon conditions or counters.

The assignment is testing your ability to *combine and apply* these concepts to solve a problem. Review Reading 3.3 and then revise the pseudocode of your program to see if you can improve it.

Then, try to implement parts of the assignment that involve loops and data structures. Only write a few lines at a time before testing your code, and make sure it is working as intended before writing more. Create separate little programs to experiment with the concepts until you understand them.

The assignment involves defining/using your own functions, which has not been covered yet… Ignore those parts for now, although you can look at the details of the function in the assignment brief and try to write the code to do what the function needs in a separate program.

If you haven't sought help or feedback on your work from your tutor by this point, you should do so!

Module 4

This module covered functions and modules. Functions allow you to abstract parts of your code that perform a well-defined task, and are particularly useful when there is something that you need to do at multiple points in the code. Your assignment requires you to create at least one function.

Think of functions as separate little programs that have their own variables. The only way that they interact with your main program is by being passed values as parameters, and by returning values back to the main program. Take the time to revise Module 4 and its additional content, since this is an important concept to understand and one that students/new programmers often struggle with.

Complete the code to create your function and call it where needed in your program – the assignment brief makes these points clear. Make sure that your pseudocode is detailing your function appropriately – see the details in the assignment brief and Reading 3.3 regarding this.

Module 5 and Beyond

Use the remaining time to finish off the assignment or catch up if you've fallen behind. Aim to leave yourself enough time to go over the brief again in detail and make sure that your pseudocode and code meets the requirements. Try implementing some of the optional extra enhancements!

Remember that staff tend to receive a lot of emails when assignment due dates approach, so don't leave it until the last minute to ask for help or feedback. If you have grounds for an extension, be sure to email a request *before* the due date and include the necessary documentation.