# **Process for Algorithm Development in Python**

## **Step 1: Read the problem description.**

Read the question in detail. If the question references something else (like and another file or the Moodle), make sure you read that too.

## **Step 2: Analyze the problem.**

What data is available to us and how is it represented?

What are the inputs, outputs, and important variables?

When does the program end?

What assumptions can I make? Ask for clarification if you aren’t sure.

What keywords are in the problem that suggest how we might solve it (if, decide, compare, repeat, remove, …)?

Think about how you would intuitively solve the problem without a formal step by step process. This might involve drawing or writing down thoughts and examples.

## **Step 3: Write an algorithm in English.**

Write out the steps of the program in plain language, like you were explaining something in a lot detail to another person.

## **Step 4: Write a detailed algorithm.**

Write out the algorithm in pseudocode. That means writing it out in a mix of language and code. Here you should think about how you can translate the steps into the 6 basic programming operations (input, procession, output, sequence, selection, repetition, and the bonus 7th one modularity).

## **Step 5: Start writing the algorithm in Python.**

We are finally writing our code into the computer. Start to translate the algorithm you wrote in step 4 into the syntax needed to get it working in Python. It is often useful to write some pieces of the code and test them on their own (instead of trying to write everything all at once). Choose variable names will make your program easy to read and make sure you add comments to clarify any complicated parts.

## **Step 6: Test your code.**

Test your code. The first thing to check is if it runs without an error. Error messages print a line number, the problem is usually on that line (although sometimes it is a problem from the line of code before that). Go back to step 5 and adjust your Python code until it run without a syntax or runtime error.

If the code runs without errors, then it is time to see if the logic works. We do this by giving it a set of test variables for that program. Do you get the outputs you expect for each one? In some questions I will provide the tests, in others you will have to think of your own set of tests to see if your code works for all the expected values. You may have to revisit your high-level or detailed algorithm to find problems with the logic.

## **Step 7: Document your code.**

Make sure the filename matches the instructions. Make sure that you have a header at the top of the file with the filename, program description, your name, and the date. Make sure that the description is detailed. Double check that you have used reasonable variable names throughout the code and adjust them if they are not clear.