EMS412U: Python Practice Questions Toolkit

Muhie Al Haimus, Silvia Valls Santafe, Dr. Rehan Shah

January 2026

Please email **m.alhaimus@se23.qmul.ac.uk** for any additional comments or corrections for this toolkit.

1 Introduction

This toolkit serves as a refresher to learning the basics of Python programming useful for EMS726P. This resource should be used as a **supplementary** resource to get familiar with Python as a programming language for this module.

2 Practice questions for Weeks 1-6

2.1 Week 1: User input

1) Write a program to read the user's name and then display a message to welcome them by name (e.g. "Hello Bob!").

Fun fact: When you type your name, your computer will say hi back, like a mini conversation between you and Python.

2) Write a program to read in two numbers and print their sum and average. Run your program with the numbers 12 and 15 to show that it works.

2.2 Week 1: Data types

1) Write a program to read in the amount of the bill and the amount of money tendered. Calculate and display the amount of change due to the customer. Run your program to show that it works with a bill amount of £8.50 and an amount tendered of £10.

Fun fact: Next time you visit a restaurant, you can try using your Python program to compute your bill (factoring in how generous you feel with your tip) and calculate the change.

2) Write a program to read in the amount of a restaurant bill and then ask

the user how much as a percentage of the total bill they would like to tip and then output the tip amount. Run your program to show the amounts on a bill of £87 with a tip of 12.5%. When outputting your total tip amount also print out the type of the variable by adding an additional print statement using the *type*() function. What do you notice about the type of the tip?

2.3 Week 2: Selection and iteration

1) Imagine you are at the cinema, and the ticket machine needs to decide which ticket to print for you: child or adult. Write a program that asks the user to input their age. If the age is 15 or less, display the message "You are entitled to purchase a child's ticket". Otherwise, display "You must buy an adult ticket!" Show that the program works correctly with the inputs of 14, 15, and 16.

Hint: Every time you test it, you are seeing conditional logic in action, Python checks a rule and decides.

2) Write a program using a `for` loop that counts from 1-100 that prints out if the variable increasing after iteration is a multiple of five.

Hint: Start at 1 → check if divisible by 5 → print message → go to next number → repeat... until 100. Use the modulus operator.

3) Write a program that uses a `while` loop to ask the user to type in 'ducky' and run the loop endlessly until the user types it in. Show it working with the words: 'cup', 'teapot', 'cake', and 'ducky' entered in sequence.

2.4 Week 3: Using libraries and functions

> Think of a **library** as a giant toolbox filled with ready made gadgets. Need to pick a random number, draw a chart, or calculate something complicated? Instead of building the tool yourself, you just borrow it from Python's library shelf. All you have to say is: import `library` (e.g. `import random`).

1) Write a program that generates a random number between 1 and 100 and then asks the user to guess it. Allow them 5 guesses before the game is "lost". Help them by displaying whether the guess is too high or too low after each attempt. Make sure that if they guess correctly they WIN!

Hint: Use the library `random` *to generate random numbers!*

Once you start writing bigger programs, you will notice something: the same pieces of code appear again and again. That's when **functions** become your best friends. A **function** is like a machine you build once and use as many times as you want. You feed it some information (called parameters), it does its job inside, and it hands you back a result (return value). Think of it this way: The **function's name** tells Python what job to do, **the parentheses ( )** tell it what information it needs and the word **return** hands back the final answer.

2) A function in Python allows a programmer to minimise the amount of duplicated code in a codebase. Using a function, create a Fahrenheit to Centigrade converter and call this function three times to show that you don't have to copy and paste the code blocks or re-run the code. In the first function call, convert 150F to Celsius but do not store or output the result. In the next function call print out the value of 43F. Lastly, store the value of 89F in a variable named *answer_to_89* and then print out *answer_to_89*. Temperatures can be converted from Fahrenheit to Centigrade using the following formula, where F is the temperature in Fahrenheit and C is the temperature in Centigrade: $C = 5(F - 32)/9$. Write a function to convert Fahrenheit to Centigrade.

Extension: Modify the output to display the temperature to two decimal places, use Google if needed!

3) Write a program, to take a number from the user and display its factorial e.g.
Factorial of 2 = 2 x 1 = 2
Factorial of 3 = 3 x 2 x 1 = 6
Factorial of 4 = 4 x 3 x 2 x 1 = 24

The program should comprise of three functions: *get_number* - function that gets and returns the number from the user. $factorial$ - function that passes a number (previously entered) as a parameter and returns one value which is the factorial of this number. *main* – the main function that calls *get_number* and $factorial$ and displays the result.

2.5 Week 4: Getting Started with Mathematics in Python

1) Sum to N using a function called *sum_to_n* return the value of $\sum_{r=5}^{20} 15r^3 + 2r^2 + 3$.

2) Write a Python program to get the length and the angle of a complex number.

Hint: The user inputs a complex number, and the function outputs the modulus and arctan of the complex number.

3) Write a Python program to multiply two complex numbers together.

Hint: Use principles/methods similar to those used in the previous question to input.

2.6 Week 5: Getting familiar with numpy

1) Create a 3x2 matrix, initially filling it with zeros, then ask the user to input a value for each element in the matrix (the inputs should only be floats). You can make sure that the input type is correct by creating a separate method *validate_inputs* and rejects any rouge value. Finally, print out the matrix.

2) 3D rotations: Ask the user to input an XYZ point and then ask which axis the user would like to rotate around. Then output the rotated point to the screen.

For an anticlockwise rotation around the x-axis:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix}$$

For an anticlockwise rotation around the y-axis:

$$\begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}$$

For an anticlockwise rotation around the z-axis:

$$\begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.7 Week 6: pandas, numpy, matplotlib

1) First, download the Nvidia stock data set here.
Then upload the file to the Juypterhub editor in your working directory; you may also need to download *pandas* to do this make a new cell and type `!pip3`

`install` *pandas*. Then using *pandas*, open the file, separate each column, and store them as a variable.

a) First plot a bar chart of close vs open against time.

b) Then make a separate scatter (line) graph of the percentage difference between close and open (this will require extra processing as the dataset does not include this metric).

c) Plot a scatter graph of year-on-year stock growth.

2.8 Bonus Questions

1) Matrix transform: Make a program that allows you to create any 2D shape (and plot it on a *matplotlibpyplot*), but start with a rectangle represented by four 2x1 matrices and then ask if the user would like to move the rectangle either clockwise or anticlockwise and by what angle. The matrices for these rotations are as follows:

For a clockwise rotation about the origin:

$$\begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix}$$

For an anticlockwise rotation about the origin:

$$\begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$$

Then ask the user if they would like to scale the shape. Plot all of the points on the *pyplot*. This is a great visualization of matrix rotations for computer graphics and 2D games!

2) Create a Bubble Sort Algorithm

3) Create a Binary Search Algorithm

4) Create a Quicksort Algorithm

Hint: Use Google to find out how these sort and search algorithms work.

3 Solutions

Answers to these questions can be found in the **Python Toolkit: 'Squashing bugs and not snakes'** section on QMPlus.