

BIOS2016 Patchwork 1 2020/21

- Screenshot your code AND console output as appropriate.
- Include the source code screenshot and the output screenshot as appropriate
- Remember to comment your code and think about code style.
- You will submit as a PDF. It shouldn't have to be very long (as a guide, number of lines of code in my answers is given in [brackets] for the longer questions)
- If you are unsure how to code a part, try to explain in pseudocode what you are trying to do for partial marks
- Remember to answer as much as you can for each part and question. Partial marks will be given if you are on the right track but haven't finished a question.

Part A – Basic Python knowledge (worth 50% of Patch 1)

1. **A *for* loop to print a times-table.** *For* loops are useful when we know how many times we want to run a chunk of code. Write a *for* loop which prints out a number starting at 0 and counts up in 8's (ie. the "eight times table"!). Print out up to and including 10 x 8. Format the output so each line reads :

X times 8 is Y

where X and Y depend on the current loop.

[5%]

2. **Translate the following pseudocode into equivalent Python code:**

For x = count from 100 to 104 (inclusive)

Print x

Create variable y, make y equal to value of x

Double value of y

Print "y = " [value of y]

[10%]

Print "Finished."

3. Using a *while* loop to produce a sequence of numbers. [Guide: my answer = 10 lines long]

Create a *while* loop which will generate numbers in the Fibonacci sequence (see below), and stop after it has produced a value greater than 1000. **You cannot use a list to do this!**

Background: The Fibonacci sequence is a famous sequence of numbers which starts 0, 1, 1, 2, 3, 5, 8, 13, 21, 34...

It is generated by the following formula:

$$F_0 = 0, F_1 = 1$$

$$\text{From then, } F_n = F_{n-1} + F_{n-2}$$

So, the first few terms are:

0 (Given by F_0)

1 (Given by F_1)

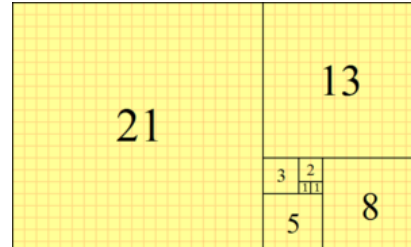
1 (Calculated by $0+1$)

2 (Calculated by $1+1$)

3 (Calculated by $1+2$)

5 (Calculated by $2+3$)

.... *Note: you may have to handle the first two terms in a special way* [15%]



The Fibonacci sequence appears in lots of patterns.
https://en.wikipedia.org/wiki/Fibonacci_number

4. A Lottery simulation. [Guide: my complete answer = 25 lines long]



Your task is to use the `random` library to help simulate a lottery competition.

In this competition there are 10,000 tickets, each with a unique ID number. Every week there is one draw, and there are 10 winners (ie. 10 winning numbers are picked).

Each ticket is identified by a unique integer number between 1 and 10,000. In this lottery, a ticket CAN win twice (unusually!), i.e. we replace a ticket back in the draw after it has been pulled out.

- Write a function which takes as a parameter a ticket ID number, and simulates a lottery draw by picking 10 random winners, and returns True if the specified ticket is a winner, and False otherwise (True and False are Boolean datatypes in Python)
- Use this function to simulate a million (ie. 1,000,000) lottery draws. Assume each time a different ticket is purchased. How many times did you win in this simulation?
- Discuss the following two points:
 - Do you think it makes a difference if the person buys a *different* ticket number each week, or if they keep the *same* number for the million draws? (use your simulation to provide evidence to your answer if you can)
 - How could you improve your answer to part (b) to give you more confidence in the amount of wins you would expect after a million draws?

[20%]

Part B – Working with data (worth 50% of Patch 1)

1. Loading and sorting a simple data file [40%]

- **Load** the “ages.txt” file from Moodle – read the numbers into a list. You can code this manually (as we did in class) or you can find a function (e.g. NumPy *loadtxt*) to help you do this.
- **Print** out the **number of items** in the file
- **Sort** the items into ascending order. You can use a sort function which we saw in class, write your own, or find a library function to do this.
- **Print** out the sorted list.

Screenshot your code, and the console output, and add to your patchwork.

2. Loading and plotting a more complex data file [60%] [Guide: my complete answer = 25 lines of code]

- Download historic weather data for Sutton Bonington Weather Station
 - **This is on Moodle:** suttonboningtondata_moodle.csv.
If you're interested, originally it came from <https://www.metoffice.gov.uk/public/weather/climate-historic/#?tab=climateHistoric>. I tidied it up in Excel, replacing special characters like '*' and '---', and saved it as CSV format.
 - **If you get stuck loading the file**, you can further pre-process the data into a format you are more comfortable reading into Python, perhaps using Excel. **Please explain** in the patchwork what you have done to the data. This will lose some marks, but will allow you to attempt all the sections below.
- Write a python program to:
 - **Load** the data file. I would recommend the NumPy *loadtxt* function to help you do this. You will need to look up how to use this function's parameters.
 - **Print to screen** how many items (rows of data, not including headers) are loaded from the data file
 - Work out the **minimum temperature** recorded in this data. Write your own code or use a library function to help you. **Print out this value.**
 - The Sun column is currently in hours. **Recalculate this column** so the values are in days (assume a day is 24 hours).
 - **Re-save** this file as a new filename– you may want to use *numpy.savetxt*, but you don't have to.*
 - Lastly, **plot the 2017 monthly max temperatures** as a line graph, using *matplotlib*. Remember to label axes, add a title etc. *Add a copy of this figure to your PDF*

**Also please upload your new saved file, if you attempted that element*