

15-110 Check1 - Written Portion

Name:

AndrewID:

Complete the following problems in the fillable PDF, or print out the PDF, write your answers by hand, and scan the results.

Also complete the programming problems in the starter file check1.py from the course website.

When you are finished, upload your check1.pdf to **Check1 - Written** on Gradescope, and upload your check1.py file to **Check1 - Programming** on Gradescope. Make sure to check the autograder feedback after you submit!

Written Problems

[#1 - Writing Algorithms - 36pts](#)

[#2 - Running Code - 24pts](#)

[#3 - Basic Programming Syntax - 20pts](#)

Programming Problems

[#1 - Data Types - 20pts](#)

Written Problems

#1 - Writing Algorithms - 36pts

Can attempt after Algorithms and Abstraction lecture

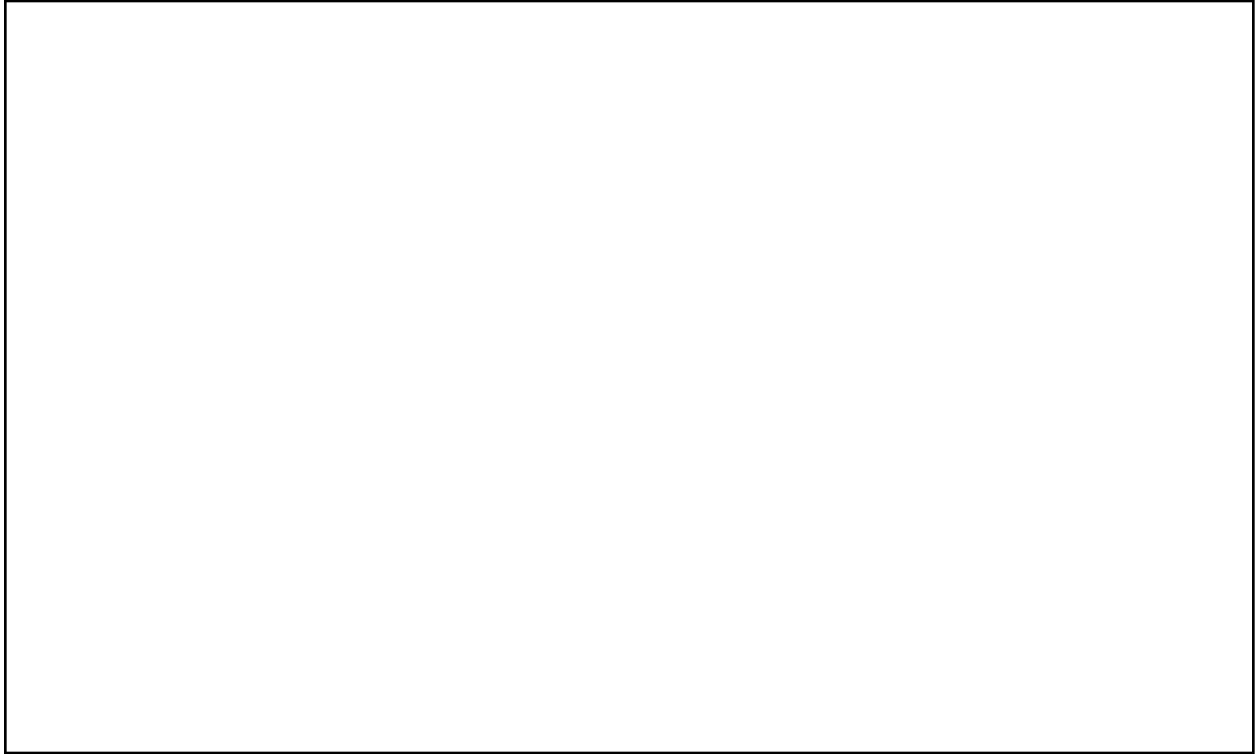
In this problem, you will write plain-language algorithms (not code!) at three levels of abstraction. Assume all of your instructions will be provided verbally (no pictures).

Do not write more than 100 words per question (and can write much less!).

First, write an algorithm at a **low** level of abstraction that instructs a person on how to write the capital letter L. Assume the person you are instructing has almost no prior knowledge- they know directions (up/down/left/right/etc) but nothing else about writing.



Second, write an algorithm at a **medium** level of abstraction that instructs a person on how to write the word 'ALL' in English, in all capital letters. This time you can assume the person you're instructing has a little more prior knowledge- what paper and a pen/pencil are, how to draw straight lines, etc.



Finally, if you wanted to provide an algorithm on how to write the word 'ALL' in English at a **high** level of abstraction, what additional starting knowledge would you give the person being instructed?



#2 - Running Code - 24pts

Can attempt after Programming Basics lecture

The following question is intended to make you feel more comfortable with running code and encountering errors. In each of the following examples, copy the line of code into the interpreter (next to `>>>`) and press Enter to run it. Then copy the output in the interpreter into the space below the code and check a box below that to indicate whether the code ran successfully or raised an error.

`5 / (4 - 2)`

Ran Successfully Raised an Error

`"Hello World"`

Ran Successfully Raised an Error

`(8 + 3) < (5 * 2)`

Ran Successfully Raised an Error

`8 + "two"`

Ran Successfully Raised an Error

#3 - Basic Programming Syntax - 20pts

Can attempt after Programming Basics lecture

Assume you've created some device to detect a medical condition. You've run a bunch of tests on study participants to determine how often the device detects the condition vs. not, and how often it gets the answer right vs. not. In other words, you've calculated the **true positive**, **false positive**, **false negative**, and **true negative** rates of this device.

Read more here: https://en.wikipedia.org/wiki/Binary_classification

You've now set up four variables in your code that hold those four rates. You want to calculate the **recall** and **precision** of your device using these variables (not using numbers directly).

```
truePos = 15 # true positive - condition detected correctly
falseNeg = 8 # false negative - device did not detect condition that existed
falsePos = 2 # false positive - device detected a condition that didn't exist
trueNeg = 75 # true negative - 'no condition' detected correctly
```

Recall is calculated as $\frac{\text{true positive}}{\text{true positive} + \text{false negative}}$ (in other words, how many of the existing cases were detected?). Write a line of code here to calculate the recall of the device and store it in a variable **recall**.

Precision is calculated as $\frac{\text{true positive}}{\text{true positive} + \text{false positive}}$ (in other words, how accurate is the device when it gets a positive result?). Write a line of code here to calculate the precision of the device and store it in a variable **precision**.

Using these two variables, you can calculate the **F-score** of the device. This is a more general measure of the device's accuracy. The F-score is calculated as:

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Write a line of code to calculate and display the F-score of the device. This should use the two variables you defined and print "Accuracy: " followed by the result. For example, if the accuracy is 0.75, the code should print "Accuracy: 0.75".

Programming Problems

For each of these problems (unless otherwise specified), write the needed code directly in the Python file, under the comment and print statement that correspond to the problem. Do not delete the provided print statements- we're using them to autograde.

If you find yourself struggling to get your code to work, remember your resources! Office hours in particular are useful for debugging problems.

Before you submit: click 'Run File as Script' to make sure your code runs without raising an error message. Any syntax or runtime errors left in the code will result in a deduction on the assignment grade. You should do this for all future programming assignments as well.

#1 - Data Types - 20pts

Can attempt after Programming Basics lecture

Under the line `print(' ---1---')`, write Python code to:

1. Assign the integer 15 to the variable **a**.
2. Assign the float 3.14 to the variable **b**.
3. Assign the string "20" to the variable **c**.
4. Assign the boolean True to the variable **d**.
5. Evaluate 5 minus 1.7 and assign that expression to the variable **e**.
6. Check whether 8 is less than 5 and assign that expression to the variable **f**.
7. Reassign the variable **a** to hold the value 45.
8. Concatenate **c** and "21" and assign the result to variable **g**. Don't change the value in **c**.

Feel free to print any of these variables to check your work.