

---

**Due** Feb 1 by 11:59pm      **Points** 100**Available** Jan 23 at 8:30am - Feb 3 at 11:59am 11 days

---

[Home](#)[Resources](#)

---

## Homework 1

### Data & Arithmetic Operations

Please review document: [How to approach a homework](#) before doing your homework. To submit your solution, compress all files together into one .zip file. Login to gradescope with your account, click on the appropriate assignment, and upload that single zip file. You may submit multiple times right up until the deadline; we will grade only the most recent submission.

Your solution will be evaluated according to our CS5001 [grading rubric](#). The written component accounts for 20% of your overall HW1 score; the programming component for 80%. We'll give written feedback when appropriate as well; please come to office hours with any questions about grading (email to set up a time if no current office hours work for you).

You are permitted three "late day" tokens in the semester; each one allows you submit a homework up to 48 hours late. You may apply ONE to this homework to get an extra 48 hours to work on the assignment. To use a token, email Prof. Keith **before** the assignment due date.

*Note: This is an introductory course, and we want to make sure that everyone has a solid understanding of the fundamentals, so we'll often rule some Python tools in or out. For this*

*homework, do not use conditionals, lists, or loops -- we haven't gotten there yet, and you don't need them!*

---

## Written Component (20% of your grade)

Please open a **plain-text** file (you can do this in IDLE or any plaintext editor you like, such as TextEdit or NotePad (please do NOT submit PDF, RTF or .DOC files) and type out your answers to the questions below. You can type your answers into Python to confirm, but answer the questions first!

**File to submit:** `written.txt`

### Written #1

What are the types of the following values?

- 1A            50.1
- 1B            4
- 1C            -5.0
- 1D            "hello"
- 1E            '15.5'

### Written #2

What do the following expressions evaluate to?

- 2A            7 / 5
- 2B            7.0 / 5.0
- 2C            7 // 5
- 2D            7 % 5
- 2E            1 \*\* 5

---

## Programming Component (80% of your grade)

## Code Structure and Style

Read each part of the programming portion carefully as to not miss any details.

Note that we're **requiring you to submit flowcharts for programming parts 1 and 3, and test cases for part 2**. These items will be counted as part of your documentation score in the grading rubric. Don't forget to include these elements if you want to receive full credit for that part of the rubric.

Make sure you review the [Python Style Guide](#). A percentage of your score for every homework is for writing good, clear, readable code. There's a lot in there. Read the style guide sections on them and make sure your comments and variables are helping to make your code nicely written.

**For Homework #1, you will be graded on your adherence to Style Guide sections CS1 - CS4.**

We'll write *all* our Python programs using the same structure, with the heart of our code inside a main function. Before you write any other code, type **def main():** at the very top and **main()** at the very bottom (don't forget the **if \_\_name\_\_ == "\_\_main\_\_":** too). Your program's code will go in between.

```
def main():  
  
    # Your code goes here!  
  
    # Make sure you indent one tab over  
  
if __name__ == "__main__":  
    main()
```

*For Homework #1, you may assume “good” input. In other words, we will not be using input to purposely “break” your code (e.g. if you ask for numeric value, we will enter that rather than a string or other data type).*

*Note: If you are submitting a .zip archive for your assignment, be sure to zip (compress) only your solution files. Do NOT zip the enclosing folder. Our auto-grader looks for your solution code to be at the top-level (root) directory so to pass the auto-tests, your code must be at that top-level.*

*Note 2: Gradescope allows you to simply drag-n-drop your .py (Python), .png (Graphics) and .txt (Plain-text) files to the submission upload. In other words, it is NOT mandatory for you to zip your assignments if you don't want to. Just ensure that if you use drag-n-drop that you multi-select ALL the files you want to submit to Gradescope. If you've omitted a file and press the "submit" button, you cannot "add" to the submission...you'll need to resubmit the whole package again.*

## Program 1 (25%)

---

**Filename for submission: bottles.py**

In many jurisdictions a small deposit is added to drink containers to encourage people to recycle them. In one particular jurisdiction, drink containers holding one liter or less have a \$0.10 deposit, and drink containers holding more than one liter have a \$0.25 deposit.

Write a program that reads the number of containers of each size from the user. Your program should continue by computing and displaying the refund that will be received for returning those containers. Format the output so that it includes a dollar sign and two digits to the right of the decimal point.

**Note:**

For this assignment (and the remaining two parts below) pay attention to your output (spelling, etc.) to match the specification precisely. If you fail to do so, your code will not pass the automated testing. You may copy the following prompts directly into your code so you don't make a spelling or spacing mistake:

How many containers that are 1 liter or less?

How many containers that are more than 1 liter?

Your total refund will be

**REMINDER:** Don't forget: Include a flowchart that describes your program's operation (this will be counted as part of your grade as 'documentation' in the grading rubric)

Sample output

```
'''  
= RESTART: /Users/keithbagley/Dropbox/NORTHEASTERN  
omework/HW1/bottles.py  
How many containers that are 1 liter or less? 2  
How many containers that are more than 1 liter? 3  
Your total refund will be $0.95  
>>>
```

## Program 2 (30%)

---

**Filename for submission: atm.py**

Consider the software that runs in an Automated Teller Machine (ATM) machine. One task that must be able to be performed is to determine the type (denomination) of currency and the numbers of bills (aka "banknotes" for those of you not from North America) for each type dispensed. A

requirement is that our ATM use the fewest number of bills that it can to dispense the specified amount of money. It only dispenses fifties, twenties, tens, fives, and ones.

Design and implement program that writes out what bills would be dispensed if this machine existed for the amount of money that was asked for by the user.

**Note:** This assignment requires you to pay particular attention to your output formatting. **Be precise in your spelling and spacing.** Match the textual prompts exactly as you see in the example below in order to pass our automated testing suite. You may copy the prompts given here to use in your code:

Welcome to PDQ Bank! Amount to withdraw? \$

Cha-ching! You asked for \$

That breaks down to:

*No flowchart required for this part of the assignment, but...Before you begin coding -- write test cases!*


### Test Cases

In the comments at the top of your file, list **3 test cases** that you came up with. Something like this but choose your own examples. You should write your test cases in such a way that, if they all pass, you can be fairly confident your program will work all the time. These test cases will count towards your "documentation" grade for this assignment.

```
'''
Test case #1:
Input: $1
Output: 0 fifties, 0 twenties, 0 tens, 0 fives, 1 ones

Test case #2:
Input: $22
Output: 0 fifties, 1 twenties, 0 tens, 0 fives, 2 ones
'''
```

Sample Output (**Note:** I'm running from the command-line instead of in IDLE so the program output starts at "Welcome to PDQ Bank..."):



```
Welcome to PDQ Bank! Amount to withdraw? $ 105
Cha-ching! You asked for $ 105
That breaks down to:
  2 fifties
  0 twenties
  0 tens
  1 fives
  0 ones
```

```
[keithbagley@Keiths-MacBook-Pro HW_1 % python3 atm.py
Welcome to PDQ Bank! Amount to withdraw? $ 237
Cha-ching! You asked for $ 237
That breaks down to:
  4 fifties
  1 twenties
  1 tens
  1 fives
  2 ones
```

```
[keithbagley@Keiths-MacBook-Pro HW_1 % python3 atm.py
Welcome to PDQ Bank! Amount to withdraw? $ 1
Cha-ching! You asked for $ 1
That breaks down to:
  0 fifties
  0 twenties
  0 tens
  0 fives
  1 ones
```

```
[keithbagley@Keiths-MacBook-Pro HW_1 % python3 atm.py
Welcome to PDQ Bank! Amount to withdraw? $ 252
Cha-ching! You asked for $ 252
That breaks down to:
  5 fifties
  0 twenties
  0 tens
  0 fives
  2 ones
```

## Program 3 (25%)

---

### Filename for submission: field\_of\_dreams.py

Create a program that reads the length and width of a farmer's field from the user in feet. Display the area of the field in acres. Ensure your output displays 3 digits after the decimal place. As with the other parts of this assignment, pay attention to your output (spelling, etc.) to match the specification precisely. If you fail to do so, your code will not pass the automated testing

**Prompts** (You may copy these directly and use them in your code)

Enter the length of the field in feet:

Enter the width of the field in feet:

The area of the field is

**Hint:** There are 43,560 square feet in an acre.

- **Don't forget:** Include a flowchart that describes your program's operation (this will be counted as part of your grade as 'documentation' in the grading rubric)

Sample Output:

```
= RESTART: /Users/keithbagley/Dropbox/NORTHEAST1  
omework/HW1/mysubmission/field_of_dreams.py  
Enter the length of the field in feet: 400  
Enter the width of the field in feet: 1000  
The area of the field is 9.183 acres  
>>>
```