CS5001
Spring 2019
HW1 (data types and arithmetic operations)
Assigned: January 10, 2019
Deadline: January 17, 2019 at 9:00am

Please review this handout (bit.ly/align-homework) before doing your homework. To submit your solution, compress all files together into one .zip file. Login to `handins.ccs.neu.edu` with your CCIS 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 (bit.ly/cs5001rubric). 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 Laney's office hours with any questions about grading (email to set up a time if no current office hours work for you).

If you apply your late token, your new deadline is **January 21 at 9:00am**. Email laneys@northeastern.edu to notify us you'll be cashing it in.

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
Please open a plaintext file (you can do this in IDLE or any plaintext editor you like, such as TextEdit or NotePad) and type out your answers to the questions below. You can type your answers into Python to confirm, but answer the questions first!

Written #1

- Filename: `written.txt`

What are the types of the following values?

1A	48.25
1B	48
1C	-1
1D	-5.0
1E	"hello"

1F ‘and goodbye!’

1G ‘15’

Written #2
What do the following expressions evaluate to?

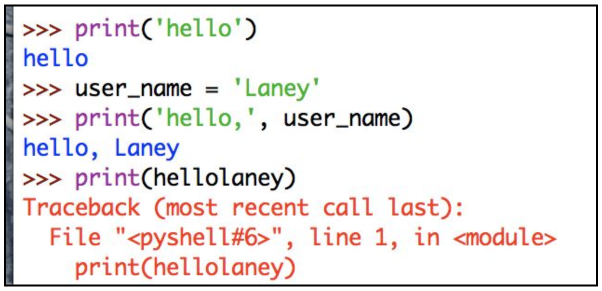
2A 7 / 5

2B 7.0 / 5.0

2C 7 // 5

2D 7 % 5

Written #3
I’ve just opened up my Python terminal and typed the following 3 statements, all using the print function. In your own words, explain why the third one would give me an error:



```
>>> print('hello')
hello
>>> user_name = 'Laney'
>>> print('hello,', user_name)
hello, Laney
>>> print(hellolaney)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    print(hellolaney)
```

Written #4
Suppose you have a variable named *amount*, and you know that its data type is float. Write one line of Python of code that will print the value in *amount* with exactly two digits after the decimal point.

Write this line of code just in your text file; don’t write an executable Python program. (We’re getting some practice here writing code without executing it. Becoming confident that our code will work is an important part of our skill set!)

Programming Component

Code Structure and Style
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, so for this homework focus on two things: comments and variable names. Read the style guide sections on them and make sure your comments and variables are helping to make your code nicely written.

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. Your program's code will go in between.

```
def main():
      # Your code goes here!
      # Make sure you indent one tab over

main()
```

Programming #1

- Filename: `racepace.py`

You and your friends have just run a road race, congratulations!

The race took place in beautiful downtown Vancouver. There were a bunch of different distance options, so some of your friends ran a 10k, some ran a 5k, and some of them even ran a 50k or more! How many miles are those distances? They didn't say, because Canada.

No matter the distance, though, we have finishing times in hours and minutes.

So you have a distance (in kilometers) and a finish time (in hours & minutes), and you want to figure out three things:

1. Miles they ran, converted from kilometers.
2. What was the average pace per mile?
3. What was the average miles per hour?

Write a python program that will figure this out for you and your running buddies. Your program should prompt the user for three inputs:

1. Number of kilometers they ran, a floating-point number
2. Number of hours, a whole number
3. Number of minutes, a whole number

You can use any wording you like to gather information for the user, it doesn't have to match our example below. However, the **order** in which you prompt them must be the same. For grading, we test your code using auto-generated input in a specific order. Ask the user for (1) distance, then (2) hours, then (3) minutes.

Before you begin coding -- write test cases!
In the comments at the top of your file, list **5 test cases** that you came up with. Something like this, but choose your own examples, and make sure you have a few edge cases in there -- make sure your test for a very long / very short race, a distance like .1 kilometers, etc.:

```
'''
      Test case #1:
      10k race, 1 hour and 1 minute:
      6.21 miles, 9:49 pace, 6.11 MPH

      Test case #2:
      25k race, 2 hours and 13 minutes:
      15.53 miles, 8:34 pace, 7.01 MPH
'''
```

Helpful hints:

- There are 1.61 kilometers in a mile
- You'll want the number of miles rounded off to two decimal places, similar to what we did in lab this week, but not identical -- if it's a **whole** <==(whole, not even. Sorry for the typo!) number of miles, print 10.0 not 10.00.
- Python's math library has a floor function should you need it, which will tell you the largest integer value less than or equal to a number. You need to import the math library to use it, so you need two steps to make this part happen:
  - At the top of your .py file, include the line `import math`
  - When you need to know the floor of a number, use the function math.floor, as in `math.floor(10.85)`, which would give you back just a plain old 10.
- You may assume that the user gives you "good" input -- all of the data are non-negative. The distance is a float and is always greater than zero. Hours and minutes are both whole numbers, and either one may be zero, but the *total* time is never zero.

Requirements: For full credit, your program must:

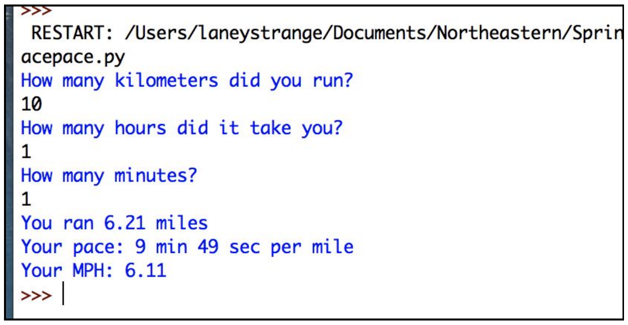
- Prompt the user for the number of kilometers, hours, and minutes -- in that order.
- Compute the total number of miles ran, rounded off to two decimal places
- Compute the minutes and seconds pace-per-mile (**head's up!!!** If you do some calculation and end up with, say, 10.8 minutes, that does *not* mean 10 minutes and 80 seconds... it means 10 minutes and 8/10 of a minute. You need to make that conversion into minutes and seconds.)
- Compute the average miles per hour, rounded off to two decimal places
- Report all of this information to the user

- As always with CS5001 assignments, your user interaction must be useful, friendly, and informative.

AMAZING points: In addition to meeting the requirements, you earn the final 2 points with...

- Use a constant variable to store the number of kilometers in a mile.
- Use round or format functions *only* for printing out, and do not save any values that have been rounded off, floored, etc.

Example Output



```
>>>
 RESTART: /Users/laneystrange/Documents/Northeastern/Sprin
acepace.py
How many kilometers did you run?
10
How many hours did it take you?
1
How many minutes?
1
You ran 6.21 miles
Your pace: 9 min 49 sec per mile
Your MPH: 6.11
>>> |
```

Programming #2

- Filename: `change.py`

Write a Python program that, for a given amount of money (given as a float), breaks the amount down into different denominations. Report the output in terms of:

- Number of dollars
- Number of quarters
- Number of dimes
- Number of nickels
- Number of pennies

Your program must report these outputs using the minimum number of coins/dollars possible. This is called an **optimization problem**. We could turn the amount into all pennies, and that's technically correct -- it converts an amount into a breakdown of denominations. But, it's not what we're going for here; instead, we want a correct solution, with the fewest coins/dollars we can possible come up with.

The input from the user is a float. If they enter more than two digits after the decimal place, round to the closest 100th.

Before you begin coding -- write test cases!

In the comments at the top of your Python file, list **5 test cases** that you came up with (which, naturally, you devised before you began writing your code). Something like this, but choose your own examples, and make sure you have a few edge cases in there -- someone enters 0, the output would be all pennies, etc.

```
'''
```

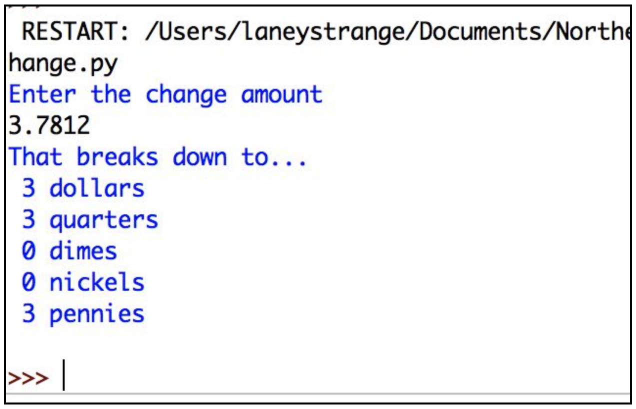
```
        Test case #1:
        $4.50
        4 dollars, 2 quarters, 0 dimes, 0 nickels, 0 pennies

        Test case #2:
        $.2879
        0 dollars, 1 quarter, 0 dimes, 0 nickels, 4 pennies
'''
```

AMAZING points: In addition to meeting the requirements, you earn the final 2 points with...

- Any value you use more than once is saved in a variable, not used as a literal.
- Use only one print statement for the final output, but without going over 80 column-width in your Python code. Look at the very bottom of the style guide for an example.

Example Output



```
 RESTART: /Users/laneystrange/Documents/Northe
hange.py
Enter the change amount
3.7812
That breaks down to...
 3 dollars
 3 quarters
 0 dimes
 0 nickels
 3 pennies

>>> |
```

Programming #3

- Filename: `bikes.py`

Suppose you are a bike shop owner who puts together bikes from spare parts. Your customers come to you with parts they've found in their garages, from scrap metal heaps, etc., and they ask you to build them all the bikes you can make with the parts they give you. You build the bikes, charge them a big pile of money for the privilege, and keep the leftover parts for yourself.

It takes the following parts to create one bicycle:

- 2 wheels
- 1 frame
- 50 links (to make a chain)

Write a Python program to calculate how many bikes you can make from the parts the user gives you. You may assume they always give you "good" input, i.e., all whole numbers, and no negatives. Your output can print simply "bikes" or "bike(s)" each time -- don't worry about singular/plural.

Before you begin coding -- write test cases!
In the comments at the top of your Python file, list **5 test cases** that you came up with. Again, make sure you have a few edge cases, like you can't make any bikes but have a lot of leftovers, you have links to make an absolute ton of bikes but only a few wheels and frames, etc.

Helpful hints:

- Python has a built-in minimum function, which gives you the minimum of several numbers. Try using it with a statement like this

```
bikes_possible = min(3, 5, 1)
# the variable bikes_possible will have the value 1
```

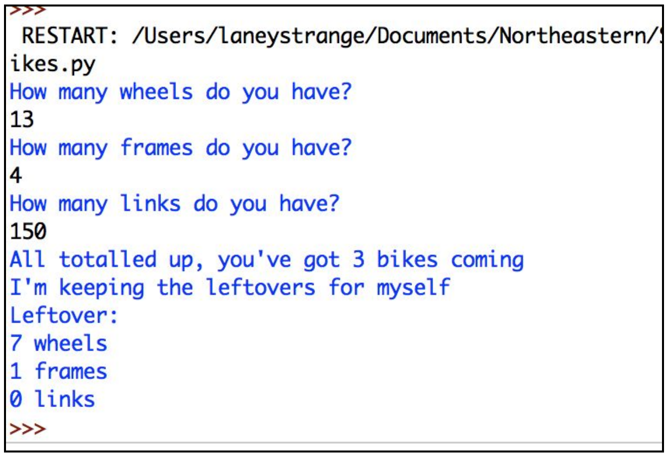
Requirements: For full credit, you must...

- Prompt the user for the # of wheels # of frames they have, and # of links they have. You can use any wording you like, but you must prompt them for the information in that order so we can test your solution.
- Report the total number of bikes you can make, and the leftover parts.
- Save all of these inputs in well-named variables of correct data types.
- As with all CS5001 programs, we expect user interaction to be friendly and informative. Don't give the user an option that doesn't exist, and make sure they always know what to do next.

AMAZING points: In addition to meeting the requirements, you earn the final 2 points with...

- Do not use any literals in your main function. Any values that need to be saved are in constants variables, initialized at the top of your program, above main.

Here's an example run of the program:



```
>>>
 RESTART: /Users/laneystrange/Documents/Northeastern/
ikes.py
How many wheels do you have?
13
How many frames do you have?
4
How many links do you have?
150
All totalled up, you've got 3 bikes coming
I'm keeping the leftovers for myself
Leftover:
7 wheels
1 frames
0 links
>>>
```