CS5001
Fall 2018
HW1 (data types and arithmetic operations)
Assigned: September 13, 2018
Deadline: September 19, 2018 at 11:59pm

Please review this handout (bit.ly/5001handout1) 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/5001rubric). 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 **September 24, 2018 at 11:59pm**. Email laneys@northeastern.edu to notify us you'll be cashing it in.

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?

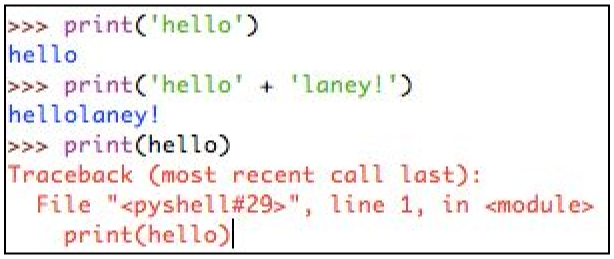
2A 7 / 5

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
>>> print('hello' + 'laney!')
hellolaney!
>>> print(hello)
Traceback (most recent call last):
  File "<pyshell#29>", line 1, in <module>
    print(hello)
```

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 all your friends have just run a road race, congratulations!

The race took place in beautiful downtown Vancouver, which is a lovely place to run. 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! No matter the distance, when the race director gives you your finishing time, it's in hours and minutes.

So you have a distance (in kilometers) and a finish time (in hours & minutes), and you want to figure out two things for you and all your buddies:

1. What was your average pace per mile?
2. What was your average miles per hour?

Write a python program that will figure this out for you. Get three inputs from the user:

1. Number of kilometers they ran
2. Hours
3. Minutes (don't bother with seconds)

Then report to the user their:

- Miles they ran (converted from kilometers)
- Average pace per mile
- Average miles per hour (MPH)

In the comments at the top of your file, list **3 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:

```
'''
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

Test case #3:
5k race, 0 hours and 17 minutes:
    3.11 miles, 5:28 pace, 10.96 MPH

'''
```

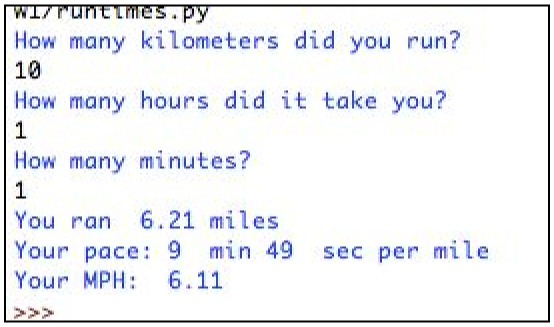
Helpful hints:

- There are 1.61 kilometers in a mile
- Python has a built-in round function, that can round off a floating-point number to 0, 1, or more decimal places. Try it out by running Python in interactive mode and trying a few things like `round(9.185)` and `round(9.185, 2)`.
- Python's math library also has a floor function, 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.

Requirements: For full credit, your program must:

- Prompt the user for the number of kilometers, hours, and minutes
- 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 80/100 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.

Example Output



```
w1/runtimes.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: `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)

You need *all* of those parts to make a complete bike. For example, if you have 3 wheels, 10 frames, and 500 links, you can make only one bike. Leftover, you would have 1 wheel, 9 frames, and 450 links.

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, e.g., no negative numbers.

In the comments at the top of your file, list **3 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:

```
'''
Test input: 2 wheels, 1 frame, 50 links
Expected output: 1 bike built. Leftover: nothing.

Test input: 3 wheels, 10 frames, 500 links
Expected output: 1 bike built. Leftover: 1 wheel, 9 frames, 450 links.

Test input: 10 wheels, 7 frames, 250 links
Expected output: 5 bikes built. Leftover: 0 wheels, 2 frames, 0 links.
'''
```

Helpful hints:

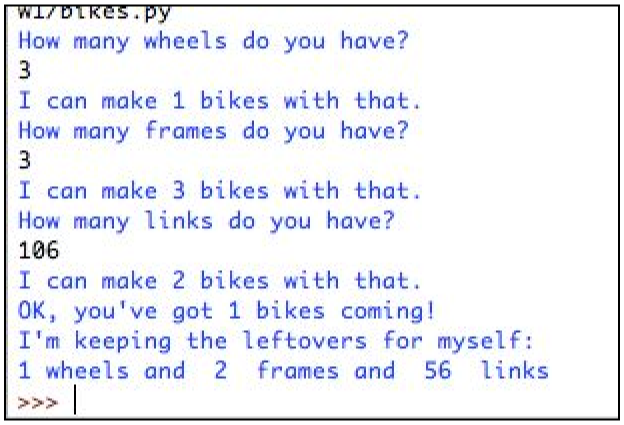
- Python has a built-in minimum function, which gives you the minimum of several numbers. Try using it with a statement like this

```
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 they have, and report how many bikes that makes.
- Prompt the user for the # of frames they have, and report how many bikes that makes.
- Prompt the user for the # of links they have, and report how many bikes that makes.
- 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.

Here's an example run of the program:



```
w1/bikes.py
How many wheels do you have?
3
I can make 1 bikes with that.
How many frames do you have?
3
I can make 3 bikes with that.
How many links do you have?
106
I can make 2 bikes with that.
OK, you've got 1 bikes coming!
I'm keeping the leftovers for myself:
1 wheels and  2  frames and  56  links
>>> 
```