

MSU CSE 803 Computer Vision, Fall 2022

Homework 0

The goal of this assignment is to incentivize learning to write reasonably good python/numpy code. This is so that:

- You don't learn it on your own and discover some super useful function at the end of the semester
- If you're doing something like calculating eigenvectors wrong, you find out in a low-stakes way
- You get credit for spending time doing this

You do NOT need to submit your solutions of this homework. A good introduction to python and Numpy from Dr. Justin Johnson is here [Numpy Tutorial](#). The problems are prepared by Dr. David Fouhey.

Instructions

Each assignment requires you to fill in the blank in a function (in `tests.py` and `warmup.py`) and return the value described in the comment for the function.

The code will start with:

```
def sample1(L):
    #Given:
    #    a list L
    #Return:
    #    the 1st entry of L (counting like humans, not computers)
    #Hint: No hints for you

    return None
```

You can then fill in:

```
def sample1(L):
    #Given:
    #    a list L
    #Return:
    #    the 1st entry of L (counting like humans, not computers)
    #Hint: No hints for you
```

```
return L[0]
```

You can test your implementation by running the test script.

```
python run.py --test b1           #Test problem b1
python run.py --allwarmups        #Test all the warmup problems
python run.py --alltests          #Test all the test problems
python run.py --alltests --pdb    #Test all the test problems, and
launch the                        #pdb debugger if things don't
match so you                      #can find the differences
```

This will show:

```
$ python run.py --allwarmups
Running b1
Running b2
...
Running b20
Ran warmup tests
20/20 = 100.0
```

Warmup Problems

You should solve all 20 warmup problems (in `warmup.py`). These are all solvable in one line.

Tests

You can solve any 15 of the 20 problems (in `tests.py`). Many are not solvable in one line. Only one (p10) should be done with a for loop. You are free to choose any 15 that you want but you are highly encouraged to do all 20. It may pay off to know how to do: p2, p11, p12, p14, and one of (p18, p19, p20)

Here is one example:

```
def p4(t):
    #Given:
    #   a tuple of 3x3 rotation matrix R
    #   Nx3 matrix M
    #Return:
    #   a Nx3 matrix of the rotated vectors
    #Par: 3 lines
    #Instructor: 1 line
    #Hint:
    #   1) Applying a rotation to a vector is right-multiplying the
rotation
    #       matrix with the vector
```

```
# 2) .T transposes; this may make your life easier
# 3) np.dot matrix-multiplies
R, M = t #unpack
```

```
return None
```

For each, we provide:

- *Given*: The arguments that are given as input
- *Return*: What you have to return
- *Par*: How many lines it might take. If it takes more than this, there's a better way. Unless the hints suggest a for loop, you should not use one.
- *Instructor*: How many lines I (David) took to write it the first time around. See if you can beat me (no semicolons though)!
- *Hint*: functions you're likely to find useful and tips for writing the function.