

PS0

Max Zuo
mzuo8@gatech.edu

2020

1 Short answer problems (Python)

1

Completed

2

- (a) Stores a numpy array of length 1000 (shape (1000,)) of the integer values $[0, 999]$ (inclusive), which has a randomized order into the variable `x`. In other words, it is a permutation of $[0, 999]$.
- (b) The first line stores a numpy array of shape (3,3) into the variable `a` with the values

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

The second line slices the third row of the array `a` and stores this row as a numpy array into a variable `b`. Note: `b` does not copy the row as a new numpy array, rather is a pointer to the row within `a`. The numpy array stored at `a` is essentially unaffected.

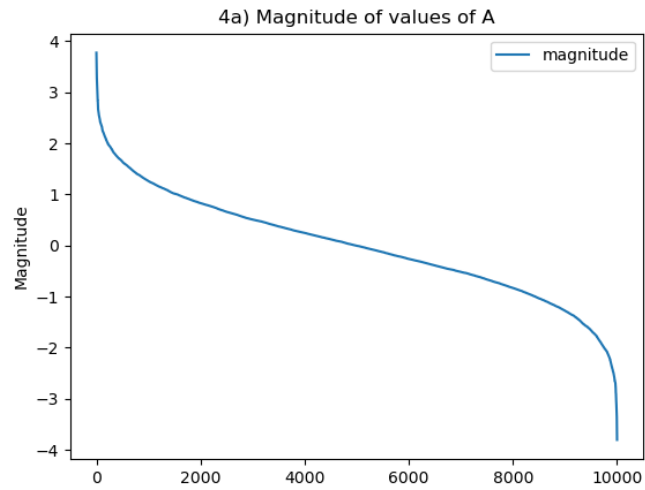
- (c) The first line does the same thing as the first line in b).
The second line takes the the numpy array `a` and flattens the shape into (9,), which it then stores into the variable `b`. Note: `b` does not copy the a new numpy array, rather is a pointer to the array at `a` with a different shape. The shape of `a` is unaffected.
- (d) The first line produces a numpy array of shape (5,1) of values sampled from the standard normal distribution $N(0, 1)$ and stores the result into a variable `f`.
The second line slices the array `f` and keeps all strictly positive values while ignoring all non-positive values and storing it in `g`. This *should* actually be producing a new array and storing it in `g`.
- (e) The first line produces a numpy array of shape (10,) in which all values are 0.5 and stores it in a variable `x`.
The second line produces a numpy array of shape (10,) in which all values are 0.5 and stores it in a variable `y`. This does so by first producing an array of shape (10,) with values of 1 then multiplying it 0.5
The third line produces a numpy array of shape (10,) in which all values are 1. and stores it in a variable `z`. This is done so by adding `x` and `y` which results in all ones of shape (10,).
- (f) The first line produces a numpy array of shape (99,) of the integer values $[1, 99]$ (inclusive) and stores this resulting array in the variable `a`.
The second line takes the array `a` and reverses its order and stores the resulting array at `b`. Note: `b` does not copy the result as a new numpy array, rather it still holds a pointer to `a`. The numpy array stored at `a` is essentially unaffected.

3

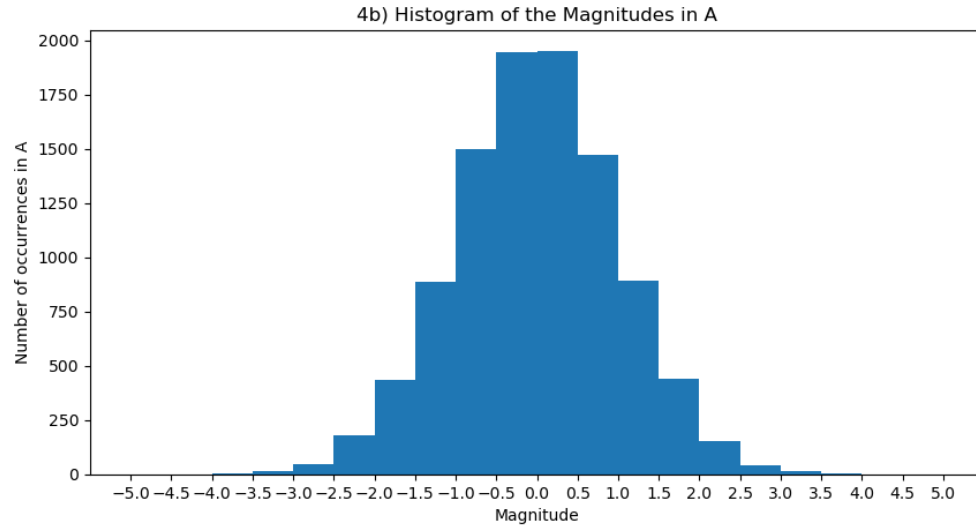
- (a) `results = (6 * np.random.rand(N) + 1).astype(int)`
 # results stores the faces (i.e. 1, 2, 3, 4, 5, 6) rolled from N different trials
- (b) `y = np.array([1, 2, 3, 4, 5, 6])`
 `z = y.reshape((3, 2))`
- (c) `r, c = np.where(z == np.max(z))`
- (d) `v = np.array([1, 8, 8, 2, 1, 3, 9, 8])`
 `x = np.sum(v == 1)`

4

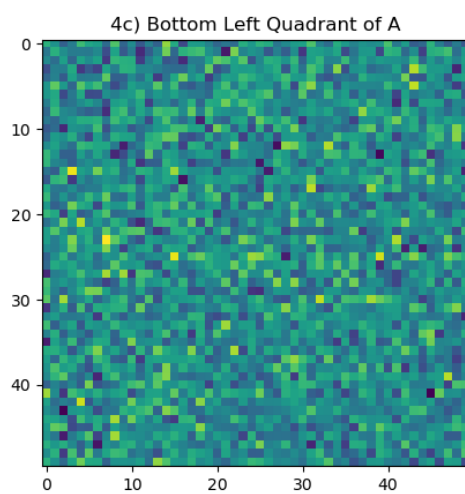
(a) 4a.png



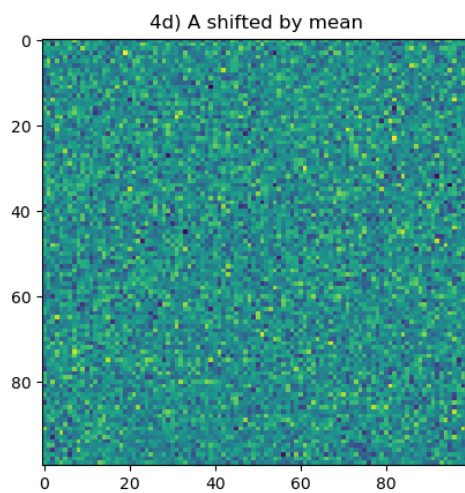
(b) 4b.png



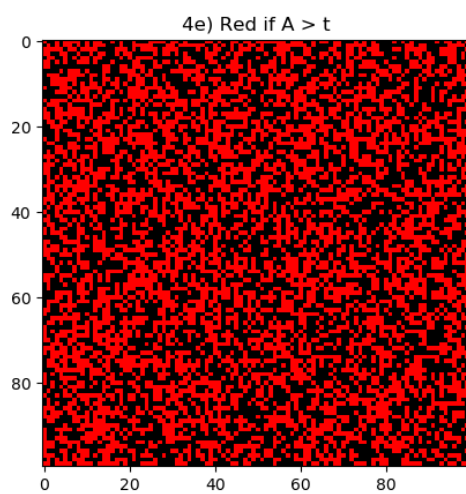
(c) 4c.png



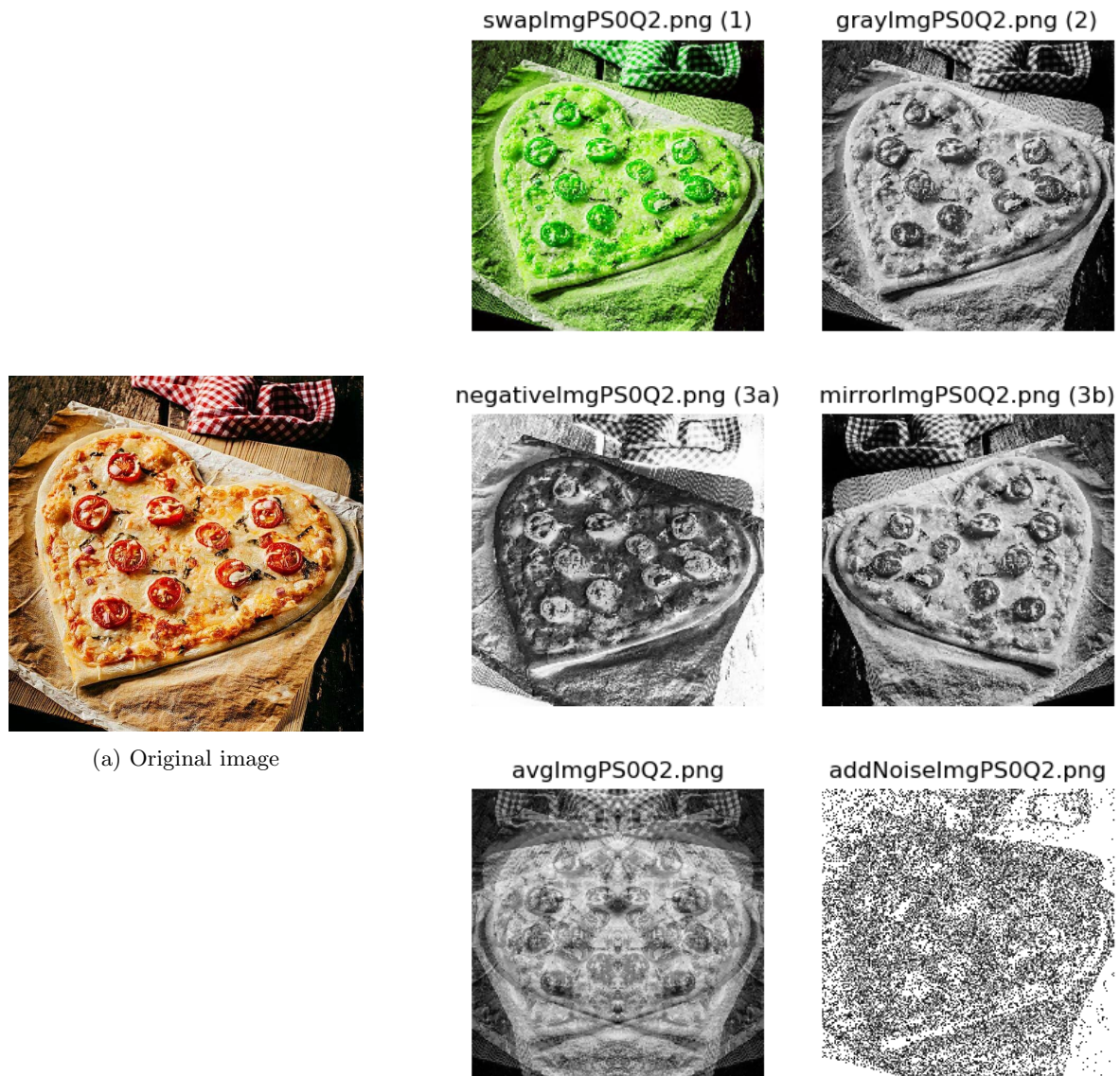
(d) 4d.png



(e) outputZPS0Q1.png



2 Short programming example (Python)



(b) Results

Figure 1: Results for the short programming assignment