CPE4040_Homework_3updated

October 16, 2024

CPE 4040: Homework 3

1. Write your answer in the cell provided under each question.
2. **You must write comments to explain your thoughts and earn full credit**.
3. **Show your execution result**.
4. Do your own work. **Do not copy-and-paste other people's (or Generative AI's) codes.**
5. Please do not use pandas since it is not covered yet.

0.0.1 Submission:

- **Submit this notebook file and the pdf version** - remember to add your name in the filename.

[1]:
```
import numpy as np
```

0.0.2 Check your Numpy Version

[2]:
```
print("Numpy version: " + np.__version__)
```

```
Numpy version: 1.26.4
```

Q1: Checker Board (10 Points) Please write NumPy codes to generate this 9x9 matrix -

[[1 0 0 1 0 0 1 0 0] [0 1 0 0 1 0 0 1 0] [0 0 1 0 0 1 0 0 1] [1 0 0 1 0 0 1 0 0] [0 1 0 0 1 0 0 1 0] [0 0 1 0 0 1 0 0 1] [1 0 0 1 0 0 1 0 0] [0 1 0 0 1 0 0 1 0] [0 0 1 0 0 1 0 0 1]]

Note: you cannot simply copy the numbers. Pay attention to the pattern. There are many ways to do this.

[3]:
```
diag = np.diag([1,1,1])
cat = np.concatenate((diag, diag, diag), axis=1)
concat = np.concatenate((cat, cat, cat))
print(concat)
```

```
[[1 0 0 1 0 0 1 0 0]
 [0 1 0 0 1 0 0 1 0]
 [0 0 1 0 0 1 0 0 1]
 [1 0 0 1 0 0 1 0 0]
 [0 1 0 0 1 0 0 1 0]
 [0 0 1 0 0 1 0 0 1]
 [1 0 0 1 0 0 1 0 0]
 [0 1 0 0 1 0 0 1 0]
 [0 0 1 0 0 1 0 0 1]]
```

Q2: Array Indexing and Slicing (12 Points)

Q2.1 Please convert this vector, `np.arange(48)`, into a 8x6 array, i.e, 8 rows and 6 columns. Let's call this array A.

[4]:
```
A = np.arange(48).reshape(8,6)
print(A)
```

```
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]
 [18 19 20 21 22 23]
 [24 25 26 27 28 29]
 [30 31 32 33 34 35]
 [36 37 38 39 40 41]
 [42 43 44 45 46 47]]
```

Q2.2 Write codes to extract the elements in columns 2, 3, and 5 from A.

[5]:
```
arr22 = A[:, [2,3,5]]
print(arr22)
```

```
[[ 2  3  5]
 [ 8  9 11]
 [14 15 17]
 [20 21 23]
 [26 27 29]
 [32 33 35]
 [38 39 41]
 [44 45 47]]
```

Q2.3 Create a new 3x5 array, with elements from rows 6, 1, 3 of A (in that order). Hint: use fancy indexing

[6]:
```
arr23 = A[[6,1,3], :5]
print(arr23)
```

```
[[36 37 38 39 40]
 [ 6  7  8  9 10]
 [18 19 20 21 22]]
```

Q2.4 Write codes to extract four elements from A with the following indices (4,2), (5,0), (1,2), (3,3).

[7]:
```
arr24 = np.array([A[4][2], A[5][0], A[1][2], A[3][3]])
print(arr24)
```

```
[26 30  8 21]
```

Q3: Array Processing (15 points)

Given a 9-by-9 matrix of random integers from 0 to 99:

```
[8]: np.random.seed(1)
     arr = np.random.randint(100,size=(9,9))
     arr
```

```
[8]: array([[37, 12, 72,  9, 75,  5, 79, 64, 16],
            [ 1, 76, 71,  6, 25, 50, 20, 18, 84],
            [11, 28, 29, 14, 50, 68, 87, 87, 94],
            [96, 86, 13,  9,  7, 63, 61, 22, 57],
            [ 1,  0, 60, 81,  8, 88, 13, 47, 72],
            [30, 71,  3, 70, 21, 49, 57,  3, 68],
            [24, 43, 76, 26, 52, 80, 41, 82, 15],
            [64, 68, 25, 98, 87,  7, 26, 25, 22],
            [ 9, 67, 23, 27, 37, 57, 83, 38,  8]])
```

Q3.1 Observe that each 9x9 matrix is essentially nine blocks of 3x3 matrix. Write a Python code to calculate the mean value for each of the 3x3 matrix.

Hint: use array slicing to index each of the 9 sub-block.

```
[9]: arr31 = arr
     arr31 = arr31.reshape(3, 3, 3, 3).mean(axis=(1,3))
     print(arr31)
```

```
[[37.44444444 33.55555556 61.        ]
 [40.         44.         44.44444444]
 [44.33333333 52.33333333 37.77777778]]
```

Q3.2 What is the maximum value in the array? What is the index of the maximum value?

Note: use argmax().

```
[10]: arr32 = arr
      print("Max value: ", arr32.max(), " Max value index: ", np.unravel_index(np.
       ↪argmax(arr32), arr32.shape))
```

```
Max value:  98  Max value index:  (7, 3)
```

Q4: Rolling Dices (20 Points)

Assume that you have a fair dice of six faces, that is, it is equally likely to get any of the 6 possible outcomes for one dice roll.

Q4.1 (5 points): Applying the same method that we used for coin tosses, write a code to simulate the probability of occurrence for each of the 6 numbers. You will perform at least 1,000,000 rolls.

The result should be presented as follows: * Number of 1's = 100,000. Probability = 16.67% * Number of 2's = 96,000. Probability = 16.0% * ……

```
[11]: n_toss = 1_000_000
die = np.array([side for side in range(1, 7)])
rolls = np.array([np.random.choice(die) for _ in range(n_toss)])

for i in range(1,7):
    print("Number of ", i, "'s =", (rolls == i).sum(), "| Probability =", 100 *␣
 ↪(rolls == i).sum() / n_toss, "%")
```

```
Number of  1 's = 166286 | Probability = 16.6286 %
Number of  2 's = 166305 | Probability = 16.6305 %
Number of  3 's = 166646 | Probability = 16.6646 %
Number of  4 's = 166557 | Probability = 16.6557 %
Number of  5 's = 167246 | Probability = 16.7246 %
Number of  6 's = 166960 | Probability = 16.696 %
```

Q4.2 (7 points): Suppose you are rolling two dices and add the two numbers together. The possible 11 outcomes range from 2 (1+1) to 12 (6+6).

Write a simulation of 1,000,000 trials to find out the probability for each of the 11 outcomes.

The result should be presented as follows: * Number of 2's = 14,000. Probability = 2.80% * Number of 3's = 27,800. Probability = 5.56% * .....

```
[12]: rolls2 = np.array([np.random.choice(die) for _ in range(n_toss)])

rolls_sum = rolls + rolls2

for i in range(2, 13):
    print("Number of ", i, "'s =", (rolls_sum == i).sum(), "| Probability =",␣
 ↪100 * (rolls_sum == i).sum() / n_toss, "%")
```

```
Number of  2 's = 27880 | Probability = 2.788 %
Number of  3 's = 54997 | Probability = 5.4997 %
Number of  4 's = 83476 | Probability = 8.3476 %
Number of  5 's = 111376 | Probability = 11.1376 %
Number of  6 's = 138915 | Probability = 13.8915 %
Number of  7 's = 166577 | Probability = 16.6577 %
Number of  8 's = 138837 | Probability = 13.8837 %
Number of  9 's = 110990 | Probability = 11.099 %
Number of  10 's = 83261 | Probability = 8.3261 %
Number of  11 's = 56010 | Probability = 5.601 %
Number of  12 's = 27681 | Probability = 2.7681 %
```

Q4.3 (8 points): Suppose you roll an uneven dice 5 times, what is the probability of getting '4' exactly 3 times? The probability for each face of the dice is as follows.

```
* P(1) = 0.10, P(2) = 0.15, P(3)=0.20, P(4) = 0.25, P(5) = 0.20, P(6) = 0.10
```

Write a simulation of 1,000,000 trials for the answer.

```
[13]: probability = np.array([.1, .15, .2, .25, .2, .1])
      rolls_loaded = np.array([np.random.choice(die, p=probability) for _ in␣
       ↪range(n_toss)])


      for i in range(1,7):
          print("Number of ", i, "'s =", (rolls_loaded == i).sum(), "| Probability␣
       ↪=", 100 * (rolls_loaded == i).sum() / n_toss, "%")
```

```
Number of  1 's = 100413 | Probability = 10.0413 %
Number of  2 's = 148814 | Probability = 14.8814 %
Number of  3 's = 201192 | Probability = 20.1192 %
Number of  4 's = 249681 | Probability = 24.9681 %
Number of  5 's = 199950 | Probability = 19.995 %
Number of  6 's = 99950 | Probability = 9.995 %
```

Q5: Picking Marbles (20 Points)

Q5.1 (10 points): You have a bag of 12 marbles, 6 are red, 4 are white, and 2 are blue. You randomly pick three marbles from the bag. What is the probability of drawing 1 red, 1 white, and 1 blue (in this order)? The theoretical result is 2/55 ~3.636%. Write a simulation of 100,000 trials to verify the answer.

Hint: You can follow the steps here: 1. Create an array of 12 elements, bag = np.array([‘r’,‘r’,‘r’,‘r’,‘r’,‘r’,‘w’,‘w’,‘w’,‘w’,‘b’,‘b’]) 2. Create the desired outcome of 3 marbels of red, white, and blue, bag3 = np.array([‘r’,‘w’,‘b’]) 3. For each of the trial, you use np.random.permutation() to shuffle the bag array 4. Create a Boolean array by comparing the first three elements of the shuffled array with the desired outcome. 5. If the values the Boolean array are all True –> you have a match

```
[14]: n_trials = 100_000
      bag = np.array(['r','r','r','r','r','r','w','w','w','w','b','b'])
      bag3 = np.array(['r','w','b'])

      count = 0

      trials = np.array([np.random.choice(bag, size=3, replace=False) for _ in␣
       ↪range(n_trials)])
      for trial in trials:
          if np.array_equal(trial, bag3):
              count += 1
      print("Probability = ", 100 * count / n_trials, "%")
```

```
Probability =  3.613 %
```

Q5.2 (10 points) With the same bag, you again pick out three marbles. However, the order does not matter this time. What is the probability of drawing 1 red, 1 white, and 1 blue? The theoretical result is 12/55 ~21.82%. Write a simulation of 100,000 trials to verify the answer.

Hint: This question is different from Q5.1 since the order of the drawing is not important as long as you have {r,w,b} in the outcome. You should look for another way to compare the desired outcome with the shuffled arrays.

[15]:
```
count2 = 0
for trial in trials:
    if 'r' in trial and 'w' in trial and 'b' in trial:
        count2 += 1


print("Probability = ", 100 * count2 / n_trials, "%")
```

```
Probability =  21.835 %
```

0.0.3 Q6: Birthday Problem & Normal Distribution (23 Points)

Q6.1 (8 points) With 30 people in the same room, what is the probability that at least three people are sharing the same birthday? Write a simulation of 100,000 trials to find the answer.

[16]:
```
n_trials = 100_000

birthdays = np.array([i for i in range(1, 366)])
trials = np.array([np.random.choice(birthdays, size=30) for _ in␣
 ↪range(n_trials)])

sum = 0

for trial in trials:
    trial.flatten()
    unique, counts = np.unique(trial, return_counts=True)
    if np.any(counts >= 3):
        sum += 1

print("Probability = ", 100 * sum / n_trials, "%")
```

```
Probability =  2.886 %
```

Q6.2 (8 points) With 30 people in the same room, what is the probability that Exactly three people are sharing the same birthday? Write a simulation of 100,000 trials to find the answer.

[17]:
```
sum = 0

for trial in trials:
    trial.flatten()
    unique, counts = np.unique(trial, return_counts=True)
    if np.any(counts == 3):
        sum += 1
```

```
print("Probability = ", 100 * sum / n_trials, "%")
```

```
Probability =  2.836 %
```

Q6.3 (7 poins) A company produces light bulbs, and the weight of each bulb is normally distributed with a mean of 50 grams and a standard deviation of 2 grams. If the company wants to package the bulbs in boxes of 10, what is the probability that the total weight of the bulbs in a box is between 495 and 505 grams? Write a simulation of 100,000 trials to answer this.

[18]:
```
n_trials = 100_000

sum = 0

for i in range(0, n_trials):
    box = np.random.normal(loc=50, scale=2, size=(10,1))
    box_weight = np.sum(box)
    if box_weight >= 495 and box_weight <= 505:
        sum += 1

print("Probability = ", 100 * sum / n_trials, "%")
```

```
Probability =  56.691 %
```