**Answer Key: CMP 464-C401 Sample Final Exam, Spring 2016**

1. What will the following code draw:

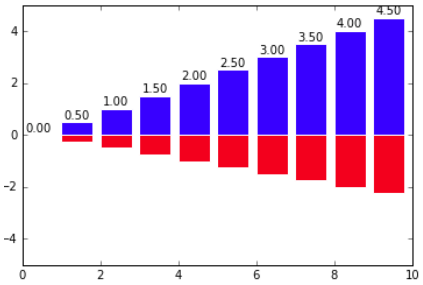
```
n = 10
X = np.arange(n)
Y1 = X/2.0
Y2 = X/4.0

plt.bar(X, +Y1, facecolor='blue')
plt.bar(X, -Y2, facecolor='red')

for x, y in zip(X, Y1):
    plt.text(x + 0.4, y + 0.05, '%.2f' % y, \
             ha='center', va='bottom')

plt.ylim(-5, +5)
```

**Answer Key:**




2. For each of the regular expressions, give a string that will matches it:

   (a) `(\d){3}\w`

   (b) `[aA]+[bB]+[cC]+`

   (c) `Bro((nx)|(oklyn)|(ther))`

   (d) `\d{3}-\d{3}-\d{4}`

   (e) `\w+@([\w\.-]+)`

**Answer Key:**

*Note: there are multiple possible answers for each:*

   (a) `123a`

   (b) `aBc`

   (c) `Bronx`

   (d) `123-456-7890`

   (e) `herbert@lehman.cuny.edu`

3. The New York City Open Data project contains all motor vehicle collisions reported to the New York Police Department. The data can be downloaded as CSV files with the following format:

```
DATE,TIME,BOROUGH,ZIP CODE,LATITUDE,LONGITUDE,LOCATION,ON STREET NAME,CROSS STREET NAME,OFF STREET
02/01/2016,0:09,BRONX,10465,40.8341548,-73.8174815,"(40.8341548, -73.8174815)",BARKLEY AVENUE,DEAN
```

All lines are formatted similarly: they start with the date, then time, the borough, zip code, latitude and longitude, and also include cross streets, types of vehicles involved, number of injuries/fatalities, and possible cause. The first line of the file gives the entries in the order they occur in the rows.

Write a program that takes a file, `bronxCollisions.csv`, and prints out all the locations that crashes occur in the 10468 zip code:

**Answer Key:**

```
import csv

#Using the dictionary reader to access by column names
f = open("bronxCollisions.csv")
reader = csv.DictReader(f)
m = [row['LOCATION'] for row in reader if int(row['ZIP CODE']) == 10468]
f.close()
print m
```

Another way to do this, using the regular csv reader (ignoring first line with column names):

```
import csv

#Note the use of 'with' for files:
with open("bronxCollisions.csv") as f:
     reader = csv.reader(f)
     reader.next()  #Ignore line with column names
     m = [row[6] for row in reader if int(row[3]) == 10468]
print m
```

4. The Center for Disease Control (CDC) provides data on the number of occurrences of Lyme Disease. Assuming you have the data stored:

```
years = [2003,2004,2005,2006,2007,2008,2009,2010,2011]
ny = [5399,5100,5565,4460,4165,5741,4134,2385,3118]
nj = [2887,2698,3363,2432,3134,3214,4598,3320,3398]
ct = [1403,1348,1810,1788,3058,2738,2751,1964,2004]
```

Write a program that will plot the percent increase in Lyme Disease occurrence with respect to the first year in the list (note some numbers plotted could be negative, since the disease occurrence has both decreased and increased from the initial observations).

**Answer Key:**

```
import matplotlib.pyplot as plt
```

```
#Data from CDC's Lyme Disease page:
years = [2003,2004,2005,2006,2007,2008,2009,2010,2011]
ny = [5399,5100,5565,4460,4165,5741,4134,2385,3118]
nj = [2887,2698,3363,2432,3134,3214,4598,3320,3398]
ct = [1403,1348,1810,1788,3058,2738,2751,1964,2004]

#We'll do the same operations for each state, so put in a function:
def scale(stateList, plt, lab,col):
     """
     Takes a list, label, and color and creates a scatter plot
     of the percentage change with respect to the first entry
     in the list.
     """
     baseNum = stateList[0]
     scaled= [i*100/baseNum-100 for i in stateList]
     plt.scatter(years, scaled, label=lab, c = col, s=75)


#Create scatter plots for each state:
scale(ny,plt,"NY", "blue")
scale(nj,plt,"NJ", "red")
scale(ct,plt,"CT", "purple")

#Set up title, axis labels, and legend, and then show:
plt.title("Lyme Disease in NY, NJ, & CT")
plt.xlabel('Years')
plt.ylabel('Percent Change')
plt.legend()
plt.show()
```

5. You are responsible for testing two different front pages for a website. The first option has aquamarine colored buttons ('Option A'), and the second option has black colored buttons ('Option B'). If 200 out of 1,000 viewers click through on Option A and 180 out of 1,000 viewers click through on Option B.

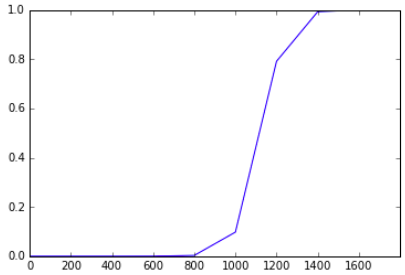
   What is the A/B test statistic for Option A and Option B (i.e. the difference of the estimated means divided by the square root of the sum of the variances)?

   **Answer Key:**

$$\begin{aligned}
\frac{\frac{180}{1000}-\frac{200}{1000}}{\sqrt{\frac{200}{1000}(1-\frac{200}{1000})/1000+\frac{180}{1000}(1-\frac{180}{1000})/1000}} &= \frac{\frac{-20}{1000}}{\sqrt{200\frac{800}{1000^3}+180\frac{820}{1000^3}}} = \frac{\frac{-20}{1000}}{\sqrt{\frac{1}{1000^3}(200\cdot 800+180\cdot 820)}} \\
&= \frac{\frac{-20}{1000}}{\sqrt{\frac{1}{1000^3}(160{,}000+147{,}600)}} = \frac{\frac{-20}{1000}}{\frac{1}{1000}\sqrt{\frac{1}{1000}(307{,}600)}} \\
&= \frac{-20}{\sqrt{307.6}}
\end{aligned}$$

6. You culled 20,000 admissions records to determine if SAT score could be used to predict admission. Using this training data, you fit the following logistic function:

$$f(x) = \frac{1}{1 + e^{200 - x/50}}$$




(a) With what probability would you expect that a student with a 500 on the SAT is admitted? Explain your answer.

(b) With what probability would you expect that a student with a 1100 on the SAT is admitted? Explain your answer.

(c) With what probability would you expect that a student with a 1500 on the SAT is admitted? Explain your answer.

(d) Say you sampled data from a single high school and found that average SAT of score of admitted students was 400 with a standard deviation of 53. Does that fit with your prediction function above? Why or why not?

**Answer Key:**

(a) `0`

(b) `50%`

(c) `100%`

(d) Our sampled data has a mean tightly clustered below 500, suggesting that all students with low SAT scores were admitted. But the logistic function predicts that students with SAT scores that low are not admitted. This suggests that the high school sampled was an outlier or that the logistic function does not fit the data well.

7. You are helping a friend find a new apartment. After going through 100 listings with them, you created a training data set of tuples of important features `(price, size, closestStop, crimeRate, like)` where:

   - `price` is the monthly rent, in dollars
   - `size` is the square footage of the apartment
   - `closestStop` is the walking distance to the nearest subway entrance
   - `crimeRate` is the number of felonies per 1000 residents over the last year
   - `like` is 0 if your friend did not like the apartment and 1 if they did.

   There are 50 new listings today for apartments. Your friend would like to know which are the 5 apartments that they would like most.

   (a) Explain how the $k$-Nearest Neighbor algorithm would work for this data:

   (b) Write a function that takes the training data, the value k, and a new apartment tuple and returns true if the majority of the k closest neighbors are liked and false if not.

   **Answer Key:**

(a) For this data, $k$-nearest neighbor would compute the closest $k$ "neighbors"– that is apartments with the most similar features. A first way to measure how similar/dissimilar two apartments $(p1, s1, w1, c1)$ and $(p2, s2, w2, c2)$ are would be comparing attributes:

$$d((p1, s1, w1, c1), (p2, s2, w2, c2)) = |p1 - p2| + |s1 - s2| + |w1 - w2| + |c1 - c2|$$

or a Euclidean distance:

$$d2((p1, s1, w1, c1), (p2, s2, w2, c2)) = \sqrt{|p1 - p2|^2 + |s1 - s2|^2 + |w1 - w2|^2 + |c1 - c2|^2}$$

This could be refined with more input from your friend. For example, if price was twice as important as the other features, it could be weighted in the distance computation:

$$d3((p1, s1, w1, c1), (p2, s2, w2, c2)) = 2 * |p1 - p2| + |s1 - s2| + |w1 - w2| + |c1 - c2|$$

(b) Very general approach (modified a bit) from the textbook:

```
def distance(v, w):
     """Pairwise differences"""
     return sum(abs(v_i - w_i) for v_i, w_i in zip(v,w))

def majority_vote(labels):
    """assumes that labels are ordered from nearest to farthest"""
    vote_counts = Counter(labels)
    winner, winner_count = vote_counts.most_common(1)[0]
    num_winners = len([count
                       for count in vote_counts.values()
                       if count == winner_count])
    if num_winners == 1:
        return winner                     # unique winner, so return it
    else:
        return majority_vote(labels[:-1]) # try again without the farthest

def knn_classify(k, labeled_points, new_point):
    """each labeled point should be a pair (point, label)"""
    # order the labeled points from nearest to farthest
    by_distance = sorted(labeled_points,
                         key=lambda (point, _): distance(point, new_point))
    # find the labels for the k closest
    k_nearest_labels = [label for _, label in by_distance[:k]]
    # and let them vote
    return majority_vote(k_nearest_labels)
```

8. You are allowed 5 colors to print an image. Write a program that uses clustering to choose the 5 colors that would best capture the image.

   - displays the original image to the screen,
   - uses clustering to choose the best 5 colors, and
   - displays the image recolored with just those 5 colors.

   You may use any method to cluster to the colors.

   **Answer Key:**

```
import matplotlib.pyplot as plt
import numpy as np

#Original image:
img = plt.imread("Midwest_Flooding_01012016_sm.png")
plt.imshow(img)
plt.show()

#Make a new image, rotated by theta and stretched by stretch:
ht = 2*img.shape[0]
wd = 2*img.shape[1]
r = np.ones((ht,wd,3))
theta = -np.pi/8
rotate = np.array([[np.cos(theta),-np.sin(theta)],[np.sin(theta),np.cos(theta)]])
stretch = np.array([[0.5,0],[0,2]])

for i in range(img.shape[0]):
     for j in range(img.shape[1]):
          (newI,newJ)= rotate.dot(stretch.dot((i,j)))
          newI,newJ = int(newI),int(newJ)
          if 0 < newI < r.shape[0] and 0 < newJ < r.shape[1]:
               r[int(newI),int(newJ),:] = img[i,j,:]
plt.imshow(r)
plt.show()
```

9. Design a program that will allow the user to visualize actual distance and transit time distance between cities.

   Inputs:

   - A dictionary of city names with values tuples (x,y) of their GIS coordinates.
   - A distance matrix of transit times between cities

   Output:

   - A map with the original cities (you may use any Python map drawing program– i.e. basemap, bokeh's mapping functions, etc.)
   - A Multidimensional Scaling (MDS) plot of the cities under the transit distances.

   Write the pseudocode and include a list of all packages you would use in the final design.

   **Answer Key:** Pseudocode: Uses the following packages:

   - `import numpy as np`
   - `import matplotlib.pyplot as plt`
   - `from mpl_toolkits.basemap import Basemap`
   - `from sklearn import manifold`

   Assume the inputs are called `cityNames` and `distances`

   (a) Initialize a basemap, m.

   (b) For each city in cityNames:

(c) (x,y) = cityNames(city)

(d) Plot a point at (x,y) labeled with the name, city.

(e) Display basemap m.

(f) Create a new plot in matplotlib.

(g) Use MDS from sk-learn on distances to create an mds instance

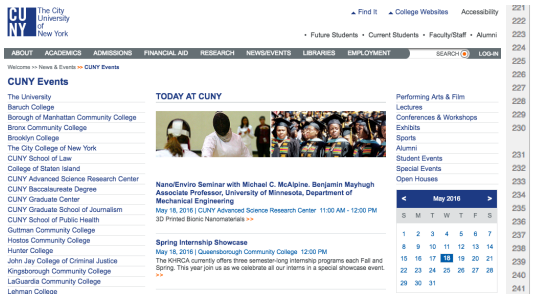
(h) Let coords be the coordinates of the embedding (i.e. coords = (mds.fit(adjust)).embedding_ )

(i) Plot the coordinates (i.e. plt.scatter(coords[:,0], coords[:,1], marker = 'o'))

(j) Display the plot.

(Note: you can avoid the first for loop by slicing lists of the values of dictionary but would need to add in a loop to add in the labels with `plt.annotate()`)

10. Design a program that scrapes the `events.cuny.edu` page and prints out the date and title of all events listed. The page and raw HTML look like:




```
                <div id="listing" class="listing">


                        <div class="block-txt">

            <div class="event-txt">
                        <!-- insert  getDetailHRef(eventId, CollegeId),  -->

                                <h3><a href="eventDetail.asp?EventId=76556">Nano/Enviro Seminar with Michael C. McAlpine.
Associate Professor, University of Minnesota, Department of Mechanical Engineering</a></h3>
                    <h4><a href="eventDetail.asp?EventId=76556">
                May 18, 2016

                | CUNY Advanced Science Research Center
                                 11:00 AM - 12:00 PM
                </a></h4>
                  <p class="short-description">
                                3D Printed Bionic Nanomaterials
                   <a href="eventDetail.asp?EventId=76556" class="suffix">&gt;&gt;</a>
                  </p>
```

**Answer Key:** We discussed two ways to do this in class:

- For the first homework, we used regular expressions to find patterns in the text and grabbed appropriate lines, and
- In Lecture #13, we used beautifulSoup to find objects in the HTML (following the Document Object Model (DOM)).

Here is pseudocode for the first approach:

- Open the webpage using `urllib2` (or `requests`).
- Use `readlines()` to read in the webpage to a list of strings, called `lines`.
- Read through each line:
- looking for the pattern: `[(January)|(February)|...|(December)]\w \d+, 2016`.
- Print out the match and the following 3 lines with the title and the tlme.
- Close file.