Python: More Data Annotation and Numpy

Name: Rudy Martinez

abc123: Lpe528

Blank notebook to be used for class exercises.

Exercise 1

Given the provided numpy array, write code that prints the following:

- Print the third row of the array
- Print the second column of the array
- Print the fourth row's third column (this should print a single number).

In [2]:
```
import numpy as np

grades = [[79, 95, 60],
          [95, 60, 61],
          [99, 67, 84],
          [76, 76, 97],
          [91, 84, 98],
          [70, 69, 96],
          [88, 65, 76],
          [67, 73, 80],
          [82, 89, 61],
          [94, 67, 88]] # "grades" is a list of lists.

grades = np.array(grades) # Converts "grades" into a numpy array
```

In [38]:
```
print(f"Third Row Values: {grades[2,:]}\n")
print(f"Second Column Values: {grades[:,1]}\n")
print(f"Fourth Row's Third Column Value: {grades[3, 2]}")
```

```
Third Row Values: [99 67 84]

Second Column Values: [95 60 67 76 84 69 65 73 89 67]

Fourth Row's Third Column Value: 97
```

Exercise 2

Given the provided array in the Lab file, write code the prints the following:

- The average grade per test (column) (this should print an array with 3 items)
- The average grade per row, i.e., the average test grade per student
- Print the max grade per test
- Print the average across all students (rows) and tests (columns)

In [22]:
```
import numpy as np

grades = [[79, 95, 60],
          [95, 60, 61],
          [99, 67, 84],
          [76, 76, 97],
          [91, 84, 98],
          [70, 69, 96],
          [88, 65, 76],
          [67, 73, 80],
          [82, 89, 61],
          [94, 67, 88]] # "grades" is a list of lists.

grades = np.array(grades) # Converts "grades" into a numpy array
```

In [65]:
```
print(f"Average grade per test {grades.mean(axis = 0)}\n")
print(f"Average grade per row {grades.mean(axis = 1).round(2)}\n")
print(f"Max grade per test {grades.max(axis = 0)}\n")
print(f"Average across all students {grades.mean().round(2)}")
```

```
Average grade per test [84.1 74.5 80.1]

Average grade per row [78.   72.   83.33 83.   91.   78.33 76.33 73.33 77.33 83.  ]

Max grade per test [99 95 98]

Average across all students 79.57
```

Exercise 3

Write code that prints the grades of all students that have an average grade less than 90.

In [39]:
```
import numpy as np

grades = [[79, 95, 60],
          [95, 60, 61],
          [99, 67, 84],
          [76, 76, 97],
          [91, 84, 98],
          [70, 69, 96],
          [88, 65, 76],
          [67, 73, 80],
          [82, 89, 61],
          [94, 67, 88]] # "grades" is a list of lists.

grades = np.array(grades) # Converts "grades" into a numpy array
```

In [54]:
```
avg = grades.mean(axis = 1)
bool_index = avg < 90
print(grades[bool_index,:])
```

```
[[79 95 60]
 [95 60 61]
 [99 67 84]
 [76 76 97]
 [70 69 96]
 [88 65 76]
 [67 73 80]
 [82 89 61]
 [94 67 88]]
```

Exercise 4

Write code to adds 10 points every student's test grade **if** their average test grade is greater than 90.

In [68]:
```
import numpy as np

grades = [[79, 95, 60],
          [95, 60, 61],
          [99, 67, 84],
          [76, 76, 97],
          [91, 84, 98],
          [70, 69, 96],
          [88, 65, 76],
          [67, 73, 80],
          [82, 89, 61],
          [94, 67, 88]] # "grades" is a list of lists.
```

```
grades = np.array(grades) # Converts "grades" into a numpy array
```

In [71]:
```
new_grades = grades.copy()

bool_index = grades.mean(axis = 1) > 90

new_grades[bool_index] += 10
new_grades
```

Out[71]:
```
array([[ 79,  95,  60],
       [ 95,  60,  61],
       [ 99,  67,  84],
       [ 76,  76,  97],
       [101,  94, 108],
       [ 70,  69,  96],
       [ 88,  65,  76],
       [ 67,  73,  80],
       [ 82,  89,  61],
       [ 94,  67,  88]])
```

Exercise 5

Write a function that takes two 1-dimensional arrays as input and returns the euclidean distance between the two arrays.

Euclidean distance is defined as

$$EDist = \sqrt{(x_0 - v_0)^2 + (x_1 - v_1)^2 + \cdots + (x_{D-1} - v_{D-1})^2}$$

The square root of a number in numpy can be calculated as np.sqrt(x), where x is a number or array.

Try to complete this exercise with for loops and with vector notation.

In [73]:
```
def edist(vec_1, vec_2):
    return np.sqrt(((vec_1-vec_2)**2).sum())

vec_1 = np.array([1,2,3,4])
vec_2 = np.array([5,6,7,8])

calc = edist(vec_1, vec_2)
calc
```

Out[73]:
```
8.0
```

Exercise 6

Write code to load the data in the "iris.csv" into numpy arrays.

The frst 4 columns are the features/attributes. The last column is the class. Simply load the class as a list of strings. Don't forget to convert the dataset into a numpy array. You can use either DictVectorizer or the CSV method on the previous slide to load the features.

In [3]:
```
with open('./iris.csv') as in_file:
    count = 0
    for row in in_file:
        print(row.strip())
        count += 1
        if count == 10:
            break
```

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

In [21]:
```
#Method 1
import csv
import numpy as np

X = [] #Features
Y = [] #Classes

with open('iris.csv') as file:
    csv = csv.reader(file, delimiter = ',')

    for row in csv:
        X.append([float(x) for x in row[0:4]])
        Y.append(row[4])

X = np.array(X)
Y = np.array(Y)
```

```
print(X.shape)
print(Y.shape)
```

```
(150, 4)
(150,)
```

In [23]:

```
#Method 2
import csv
import numpy
from sklearn.feature_extraction import DictVectorizer

X_list_dict = []
Y = []

with open('iris.csv') as file:
    csv = csv.reader(file, delimiter = ',')

    for row in csv:
        features = {'col_1': float(row[0]),
                    'col_2': float(row[1]),
                    'col_3': float(row[2]),
                    'col_4': float(row[3])}
        X_list_dict.append(features)
        Y.append(row[4])

vec = DictVectorizer(sparse = False)
X = vec.fit_transform(X_list_dict)
Y = np.array(Y)

print(X.shape)
print(Y.shape)
```

```
(150, 4)
(150,)
```

In [ ]: