PYTHON FOR DATA SCIENCE COURSE

First Assignment

Pedro Almada

13 of November 2024

Oxford, United Kingdom

Formative Assignment (amended)

Deadline: 12 pm - Friday 15th of November 2024

There are 5 exercises in these assignment. Each completed exercise is awarded 7 points for a total of 35 points. In order to pass the final assigment you will need to reach at least 40 points out of 100.

Please send your solutions to the Weekly Classes Office (weeklyclasses@conted.ox.ac.uk) with a signed DoA (Declaration of Authorship) form. Please send the notebook with your name appended to the file name.

Key things:

- remember to comment your code!
- when it comes to syntax, google and numpy/pandas documentations are your friends.

In [ ]:
```
## Let's import the libraries we are going to use later on
import numpy as np
import pandas as pd

# Let's set the precision for NumPy and Pandas
np.set_printoptions(precision=3)
pd.options.display.float_format = '{:,.2f}'.format
```

Exercises on Data Structures

1. Compute statistics on a Python list [7 points]

Given the following list

```
l = [12, 45, 12, 999, 10, 8, 76, 20, 10, 10, 7, 70, 17]
```

Compute the mean, the median and the standard deviation of the values in the list, using only built-in functionalities of python (no imported libraries, no NumPy or Pandas)

In [ ]:
```
l = [12, 45, 12, 999, 10, 8, 76, 20, 10, 10, 7, 70, 17]

# Write your solution here

#//////////#

print("Answers:")

#Calculating the MEAN (average):
#I started by calculating the lenght of the list.
```

```
#I decided to include the "float" function just to get use to it, because
# Then, calculate the sum of all the elements in the list.
#Finally, divide the total sum by the number of elements in the list.

mean = sum(l) / float(len(l))
print("Mean/Average: " +str(mean))

#I found an alternative method that could be:

#n = len(l)
#get_sum = sum(l)
#mean = get_sum / n
#print("Mean: " +str(mean))

#//////////#


#Calculating the MEDIAN:
# I Start by defining the list of numbers and calculate the length of the
# Sort the list in ascending order to arrange the elements from smallest
# Check if the length of the list is even or odd by checking the remainde
# If the number of elements is even, find the two middle elements and cal
# If the number of elements is odd, find the middle element in the sorted


sorted_list = sorted(l)#sorting the list to find the median
n = len(sorted_list)

if n % 2 == 1:
    median = sorted_list[n // 2] # odd lenght: middle element

else:
    median = (sorted_lisr[n // 2 - 1] + sorted_list[n // 2]) /2
print("Median: " + str(median))

#I found an alternative method that could be:
#n = len(l)
#l.sort() #sorting the list to find the median

#if n % 2 == 0:
    #median1 = l[n//2]
    #median2 = l[n//2 - 1]
    #median = (median1 + median2)/2
#else:
    #median = l[n//2]
#print("Median: " + str(median))

#only difference is in which condition they prioritize, odd or even.

#//////////#

# Calculating the STANDARD DEVIATION:
# Start by calculating the mean of the list, as this will be used to find
# For each element in the list, calculate the squared difference from the
# Sum all the squared differences to get the total squared deviations.
# Divide the total squared deviations by the number of elements to get th
# Finally, take the square root of the variance to obtain the standard de

variance = sum((x - mean) ** 2 for x in l) / len(l)
```

```
std_dev = variance ** 0.5
print("Standard Deviation: " +str(std_dev))



#///////////#
```

```
Answers:
Mean/Average: 99.6923076923077
Median: 12
Standard Deviation: 260.6044407837537
```

2A. Create a dictionary from a list of tuples (records) [3 points]

Given the following list of tuples, each one holding a record

```
records = [
    ('author', 'title', 'publication_year', 'page_count'),
    ('J. R. R. Tolkien', 'The Fellowship of the Ring',
1954, 398),
    ('J. K. Rowling', 'Harry Potter and the Philosopher\'s
stone', 1996, 223),
    ('Evelyn Waugh', 'Brideshead Revisited', 1945, 402),
    ('Philip K. Dick', 'Ubik', 1969, 202),
    ('Thomas Pynchon', "Gravity's Rainbow", 1973, 760),
    ('Stephen King', 'The Stand', 1978, 829)
]
```

Convert them into a list of dictionaries, using the first tuple as keys for all the dictionaries and all the other tuples as values, one per dictionary. Expected result:

```
books = [{'author': 'J. R. R. Tolkien',
  'title': 'The Fellowship of the Ring',
  'publication_year': 1954,
  'page_count': 398},
 {'author': 'J. K. Rowling',
  'title': "Harry Potter and the Philosopher's stone",
  'publication_year': 1996,
  'page_count': 223},
 {'author': 'Evelyn Waugh',
  'title': 'Brideshead Revisited',
  'publication_year': 1945,
  'page_count': 402},
 {'author': 'Philip K. Dick',
  'title': 'Ubik',
  'publication_year': 1969,
  'page_count': 202},
 {'author': 'Thomas Pynchon',
  'title': "Gravity's Rainbow",
  'publication_year': 1973,
  'page_count': 760},
 {'author': 'Stephen King',
  'title': 'The Stand',
```

```
 'publication_year': 1978,
 'page_count': 829}]
```

In [ ]:
```
records = [
    ('author', 'title', 'publication_year', 'page_count'),
    ('J. R. R. Tolkien', 'The Fellowship of the Ring', 1954, 398),
    ('J. K. Rowling', 'Harry Potter and the Philosopher\'s stone', 1996,
    ('Evelyn Waugh', 'Brideshead Revisited', 1945, 402),
    ('Philip K. Dick', 'Ubik', 1969, 202),
    ('Thomas Pynchon', "Gravity's Rainbow", 1973, 760),
    ('Stephen King', 'The Stand', 1978, 829)
]

## Write your solution here
# Rememnber: A list is a data structure that is mutable and has an order
#A Tuple is a collection of python objects that are comma separated that
# Tuple use parenthesis and lists uses square brackets.

#Starting by extracting the first tuple as the keys for the dictonaries.

keys = records[0]

#Creating a list of dictonaries by combing keys with each subsequent tupl
# For each tuple in records, (meaning all the yuples after the first) pai
#Keys are basically like the column names on a list. Values are what is a
#zip is a buint-in python function that combines multiple iterables eleme
books = [dict(zip(keys, values)) for values in records[1:]]

#Printing the results
print("Answer: ")
print("Books= ", books)
```

```
Answer:
Books=  [{'author': 'J. R. R. Tolkien', 'title': 'The Fellowship of the Ri
ng', 'publication_year': 1954, 'page_count': 398}, {'author': 'J. K. Rowli
ng', 'title': "Harry Potter and the Philosopher's stone", 'publication_yea
r': 1996, 'page_count': 223}, {'author': 'Evelyn Waugh', 'title': 'Bridesh
ead Revisited', 'publication_year': 1945, 'page_count': 402}, {'author':
'Philip K. Dick', 'title': 'Ubik', 'publication_year': 1969, 'page_count':
202}, {'author': 'Thomas Pynchon', 'title': "Gravity's Rainbow", 'publicat
ion_year': 1973, 'page_count': 760}, {'author': 'Stephen King', 'title':
'The Stand', 'publication_year': 1978, 'page_count': 829}]
```

2B. Dictionary manipulation [4 points]

Write a function that takes the dictionary like `books` and return the author whose book has the highest page count.

Apply the function to `books` to verify that it works correctly

In [ ]:
```
## Write your solution here.
#I staert by fdefining a function to find the author with highest page co
#Then, use max() with a key to find the dictionary with the highest page
#lambda is a function that helps create a small, anonumous function in a
#The syntax for lambda is= lambda arguments: expressions
#Specify key=lambaa book... so that max() compares the fictionaries based
#Then, return the author of that book.
```

```
def author_with_highest_page_count(books):
    book_with_max_pages = max(books, key=lambda book: book['page_count'])
    return book_with_max_pages['author']

#Applying the functon to get the author that has the highest page count.
highest_page_count_author = author_with_highest_page_count(books)

#Result
print("Answer: ")
print("Author with the highest paage count: ", highest_page_count_author)
```

```
Answer:
Author with the highest paage count:  Stephen King
```

Exercises on Numpy

For these exercises we will work on the IRIS dataset, a very popular multivariate dataset for data analysis, first introduced by Ronald Fisher in 1936.

See more about the IRIS dataset heare: https://archive.ics.uci.edu/ml/datasets/Iris

The data source is the file `iris.data` retrieved from the Web. It contains the data for this example in comma separated values (CSV) format. The number of columns is 5 and the number of rows is 150. The data set is composed of 50 samples from each of three species of Iris Flower: Iris Setosa, Iris Virginica and Iris Versicolor. Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species one from each other.

The variables are:

```
sepal_length: Sepal length, in centimeters.
sepal_width: Sepal width, in centimeters.
petal_length: Petal length, in centimeters.
petal_width: Petal width, in centimeters.
```

The three categories of Irises are: 'Iris Setosa', 'Iris Versicolor', and 'Iris Virginica'.

Let's first import the `iris` dataset as a 1-dimensional array of tuples.

In [ ]:

```
# Input:
# Import iris keeping the text column intact
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iri
iris = np.genfromtxt(
    url, delimiter=',',
    names=['sepal length', 'sepal width', 'petal length', 'petal width',
    dtype=[np.float64, np.float64, np.float64, np.float64, 'U15']
)
iris
```

```
Out[ ]: array([(5.1, 3.5, 1.4, 0.2, 'Iris-setosa'),
               (4.9, 3. , 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. , 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. , 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'),
               (5.4, 3.7, 1.5, 0.2, 'Iris-setosa'),
               (4.8, 3.4, 1.6, 0.2, 'Iris-setosa'),
               (4.8, 3. , 1.4, 0.1, 'Iris-setosa'),
               (4.3, 3. , 1.1, 0.1, 'Iris-setosa'),
               (5.8, 4. , 1.2, 0.2, 'Iris-setosa'),
               (5.7, 4.4, 1.5, 0.4, 'Iris-setosa'),
               (5.4, 3.9, 1.3, 0.4, 'Iris-setosa'),
               (5.1, 3.5, 1.4, 0.3, 'Iris-setosa'),
               (5.7, 3.8, 1.7, 0.3, 'Iris-setosa'),
               (5.1, 3.8, 1.5, 0.3, 'Iris-setosa'),
               (5.4, 3.4, 1.7, 0.2, 'Iris-setosa'),
               (5.1, 3.7, 1.5, 0.4, 'Iris-setosa'),
               (4.6, 3.6, 1. , 0.2, 'Iris-setosa'),
               (5.1, 3.3, 1.7, 0.5, 'Iris-setosa'),
               (4.8, 3.4, 1.9, 0.2, 'Iris-setosa'),
               (5. , 3. , 1.6, 0.2, 'Iris-setosa'),
               (5. , 3.4, 1.6, 0.4, 'Iris-setosa'),
               (5.2, 3.5, 1.5, 0.2, 'Iris-setosa'),
               (5.2, 3.4, 1.4, 0.2, 'Iris-setosa'),
               (4.7, 3.2, 1.6, 0.2, 'Iris-setosa'),
               (4.8, 3.1, 1.6, 0.2, 'Iris-setosa'),
               (5.4, 3.4, 1.5, 0.4, 'Iris-setosa'),
               (5.2, 4.1, 1.5, 0.1, 'Iris-setosa'),
               (5.5, 4.2, 1.4, 0.2, 'Iris-setosa'),
               (4.9, 3.1, 1.5, 0.1, 'Iris-setosa'),
               (5. , 3.2, 1.2, 0.2, 'Iris-setosa'),
               (5.5, 3.5, 1.3, 0.2, 'Iris-setosa'),
               (4.9, 3.1, 1.5, 0.1, 'Iris-setosa'),
               (4.4, 3. , 1.3, 0.2, 'Iris-setosa'),
               (5.1, 3.4, 1.5, 0.2, 'Iris-setosa'),
               (5. , 3.5, 1.3, 0.3, 'Iris-setosa'),
               (4.5, 2.3, 1.3, 0.3, 'Iris-setosa'),
               (4.4, 3.2, 1.3, 0.2, 'Iris-setosa'),
               (5. , 3.5, 1.6, 0.6, 'Iris-setosa'),
               (5.1, 3.8, 1.9, 0.4, 'Iris-setosa'),
               (4.8, 3. , 1.4, 0.3, 'Iris-setosa'),
               (5.1, 3.8, 1.6, 0.2, 'Iris-setosa'),
               (4.6, 3.2, 1.4, 0.2, 'Iris-setosa'),
               (5.3, 3.7, 1.5, 0.2, 'Iris-setosa'),
               (5. , 3.3, 1.4, 0.2, 'Iris-setosa'),
               (7. , 3.2, 4.7, 1.4, 'Iris-versicolor'),
               (6.4, 3.2, 4.5, 1.5, 'Iris-versicolor'),
               (6.9, 3.1, 4.9, 1.5, 'Iris-versicolor'),
               (5.5, 2.3, 4. , 1.3, 'Iris-versicolor'),
               (6.5, 2.8, 4.6, 1.5, 'Iris-versicolor'),
               (5.7, 2.8, 4.5, 1.3, 'Iris-versicolor'),
               (6.3, 3.3, 4.7, 1.6, 'Iris-versicolor'),
               (4.9, 2.4, 3.3, 1. , 'Iris-versicolor'),
               (6.6, 2.9, 4.6, 1.3, 'Iris-versicolor'),
               (5.2, 2.7, 3.9, 1.4, 'Iris-versicolor'),
```

```
(5. , 2. , 3.5, 1. , 'Iris-versicolor'),
(5.9, 3. , 4.2, 1.5, 'Iris-versicolor'),
(6. , 2.2, 4. , 1. , 'Iris-versicolor'),
(6.1, 2.9, 4.7, 1.4, 'Iris-versicolor'),
(5.6, 2.9, 3.6, 1.3, 'Iris-versicolor'),
(6.7, 3.1, 4.4, 1.4, 'Iris-versicolor'),
(5.6, 3. , 4.5, 1.5, 'Iris-versicolor'),
(5.8, 2.7, 4.1, 1. , 'Iris-versicolor'),
(6.2, 2.2, 4.5, 1.5, 'Iris-versicolor'),
(5.6, 2.5, 3.9, 1.1, 'Iris-versicolor'),
(5.9, 3.2, 4.8, 1.8, 'Iris-versicolor'),
(6.1, 2.8, 4. , 1.3, 'Iris-versicolor'),
(6.3, 2.5, 4.9, 1.5, 'Iris-versicolor'),
(6.1, 2.8, 4.7, 1.2, 'Iris-versicolor'),
(6.4, 2.9, 4.3, 1.3, 'Iris-versicolor'),
(6.6, 3. , 4.4, 1.4, 'Iris-versicolor'),
(6.8, 2.8, 4.8, 1.4, 'Iris-versicolor'),
(6.7, 3. , 5. , 1.7, 'Iris-versicolor'),
(6. , 2.9, 4.5, 1.5, 'Iris-versicolor'),
(5.7, 2.6, 3.5, 1. , 'Iris-versicolor'),
(5.5, 2.4, 3.8, 1.1, 'Iris-versicolor'),
(5.5, 2.4, 3.7, 1. , 'Iris-versicolor'),
(5.8, 2.7, 3.9, 1.2, 'Iris-versicolor'),
(6. , 2.7, 5.1, 1.6, 'Iris-versicolor'),
(5.4, 3. , 4.5, 1.5, 'Iris-versicolor'),
(6. , 3.4, 4.5, 1.6, 'Iris-versicolor'),
(6.7, 3.1, 4.7, 1.5, 'Iris-versicolor'),
(6.3, 2.3, 4.4, 1.3, 'Iris-versicolor'),
(5.6, 3. , 4.1, 1.3, 'Iris-versicolor'),
(5.5, 2.5, 4. , 1.3, 'Iris-versicolor'),
(5.5, 2.6, 4.4, 1.2, 'Iris-versicolor'),
(6.1, 3. , 4.6, 1.4, 'Iris-versicolor'),
(5.8, 2.6, 4. , 1.2, 'Iris-versicolor'),
(5. , 2.3, 3.3, 1. , 'Iris-versicolor'),
(5.6, 2.7, 4.2, 1.3, 'Iris-versicolor'),
(5.7, 3. , 4.2, 1.2, 'Iris-versicolor'),
(5.7, 2.9, 4.2, 1.3, 'Iris-versicolor'),
(6.2, 2.9, 4.3, 1.3, 'Iris-versicolor'),
(5.1, 2.5, 3. , 1.1, 'Iris-versicolor'),
(5.7, 2.8, 4.1, 1.3, 'Iris-versicolor'),
(6.3, 3.3, 6. , 2.5, 'Iris-virginica'),
(5.8, 2.7, 5.1, 1.9, 'Iris-virginica'),
(7.1, 3. , 5.9, 2.1, 'Iris-virginica'),
(6.3, 2.9, 5.6, 1.8, 'Iris-virginica'),
(6.5, 3. , 5.8, 2.2, 'Iris-virginica'),
(7.6, 3. , 6.6, 2.1, 'Iris-virginica'),
(4.9, 2.5, 4.5, 1.7, 'Iris-virginica'),
(7.3, 2.9, 6.3, 1.8, 'Iris-virginica'),
(6.7, 2.5, 5.8, 1.8, 'Iris-virginica'),
(7.2, 3.6, 6.1, 2.5, 'Iris-virginica'),
(6.5, 3.2, 5.1, 2. , 'Iris-virginica'),
(6.4, 2.7, 5.3, 1.9, 'Iris-virginica'),
(6.8, 3. , 5.5, 2.1, 'Iris-virginica'),
(5.7, 2.5, 5. , 2. , 'Iris-virginica'),
(5.8, 2.8, 5.1, 2.4, 'Iris-virginica'),
(6.4, 3.2, 5.3, 2.3, 'Iris-virginica'),
(6.5, 3. , 5.5, 1.8, 'Iris-virginica'),
(7.7, 3.8, 6.7, 2.2, 'Iris-virginica'),
(7.7, 2.6, 6.9, 2.3, 'Iris-virginica'),
(6. , 2.2, 5. , 1.5, 'Iris-virginica'),
```

```
          (6.9, 3.2, 5.7, 2.3, 'Iris-virginica'),
          (5.6, 2.8, 4.9, 2. , 'Iris-virginica'),
          (7.7, 2.8, 6.7, 2. , 'Iris-virginica'),
          (6.3, 2.7, 4.9, 1.8, 'Iris-virginica'),
          (6.7, 3.3, 5.7, 2.1, 'Iris-virginica'),
          (7.2, 3.2, 6. , 1.8, 'Iris-virginica'),
          (6.2, 2.8, 4.8, 1.8, 'Iris-virginica'),
          (6.1, 3. , 4.9, 1.8, 'Iris-virginica'),
          (6.4, 2.8, 5.6, 2.1, 'Iris-virginica'),
          (7.2, 3. , 5.8, 1.6, 'Iris-virginica'),
          (7.4, 2.8, 6.1, 1.9, 'Iris-virginica'),
          (7.9, 3.8, 6.4, 2. , 'Iris-virginica'),
          (6.4, 2.8, 5.6, 2.2, 'Iris-virginica'),
          (6.3, 2.8, 5.1, 1.5, 'Iris-virginica'),
          (6.1, 2.6, 5.6, 1.4, 'Iris-virginica'),
          (7.7, 3. , 6.1, 2.3, 'Iris-virginica'),
          (6.3, 3.4, 5.6, 2.4, 'Iris-virginica'),
          (6.4, 3.1, 5.5, 1.8, 'Iris-virginica'),
          (6. , 3. , 4.8, 1.8, 'Iris-virginica'),
          (6.9, 3.1, 5.4, 2.1, 'Iris-virginica'),
          (6.7, 3.1, 5.6, 2.4, 'Iris-virginica'),
          (6.9, 3.1, 5.1, 2.3, 'Iris-virginica'),
          (5.8, 2.7, 5.1, 1.9, 'Iris-virginica'),
          (6.8, 3.2, 5.9, 2.3, 'Iris-virginica'),
          (6.7, 3.3, 5.7, 2.5, 'Iris-virginica'),
          (6.7, 3. , 5.2, 2.3, 'Iris-virginica'),
          (6.3, 2.5, 5. , 1.9, 'Iris-virginica'),
          (6.5, 3. , 5.2, 2. , 'Iris-virginica'),
          (6.2, 3.4, 5.4, 2.3, 'Iris-virginica'),
          (5.9, 3. , 5.1, 1.8, 'Iris-virginica')],
         dtype=[('sepal_length', '<f8'), ('sepal_width', '<f8'), ('petal_le
ngth', '<f8'), ('petal_width', '<f8'), ('species', '<U15')])
```

In [ ]:

```
#I decided to visualize the data in a data frame forman for easier viewin
# For this you need to use pandas
#import pandas ad pd


iris_df = pd.DataFrame(iris) #This is to convert Numpy structured array t
print(iris_df.head()) #This displays the first 5 rows using the Data Fram

#visualizaing data like this helps with the next exercises.
```

```
   sepal_length  sepal_width  petal_length  petal_width      species
0          5.10         3.50          1.40         0.20  Iris-setosa
1          4.90         3.00          1.40         0.20  Iris-setosa
2          4.70         3.20          1.30         0.20  Iris-setosa
3          4.60         3.10          1.50         0.20  Iris-setosa
4          5.00         3.60          1.40         0.20  Iris-setosa
```

3A. How to convert a 1d array of tuples to a 2d numpy array [4 points]

Exercise: convert the `iris` 1D array to a numeric-only 2D array `iris_data` by omitting the species text field. Create a `iris_label` 1D array containing only the species text field. Keep the same indexing/order as in the original array.

In [ ]:
```
## Write your solution here


#For this, I need to structure array into two separate arrays.
#The goal is to split the structured data into two parts:
#Numeric data array(data array): A 2D numpy array contains only numeric d
#2D array will exclude any categorical labels or text fields. Also each r
#Label Arrat(label array) a 1D array contains only categorical labels, in
#1D arrat will have the same number of number of rows in the numeric data



#I start by converting the numeric fields into a 2D numpy array (iris dat
iris_data = np.array([row.tolist()[:4] for row in iris])
#For each row, you need to convert the row to a list with row.tolist()
# [:4] is used to select onoy the first 4 elements (the numneric fields)
#Wrapping trge list comprehension in np.array(...) gives a 2D array.

#Convrting the species filed into a 1D array (iris label)
iris_label = np.array([row[4] for row in iris])
#To extract Species label, you nee to use a list comprehension to get jus
#By wrapping this in np.array(...) gives a 1D array.


#Results
print("Answer: ")
print("Numeric Data(iris_data):")
print(iris_data[:5])

print("Species Labels (iris_label):")
print(iris_label[:5])
```

```
Answer:
Numeric Data(iris_data):
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
Species Labels (iris_label):
['Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa']
```

3B. Split the IRIS dataset by Label [3 points]

Split the dataset in `iris_data` according to the labels `iris_labels`.

In [ ]:
```
# Write your solution here

#First, Identying unique labels

unique_labels = np.unique(iris_label) #Extracts rge unique species names,

#then moving to split  data by each label
#We need to create a dictionary to store the split data.
#Use fictionary comprehension to create split_data, where each key is a u

split_data = {label: iris_data[iris_label == label] for label in unique_l
#iris_data[iris_label == label] uses boolean indexing to select only rows
```

```
print("Answer: ")

#Result
for label, data in split_data.items():
    print(f"label: {label}")
    print(data[:5]) #Showing the first 5 entries for each label for visua
    print()
```

```
Answer:
label: Iris-setosa
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]

label: Iris-versicolor
[[7.  3.2 4.7 1.4]
 [6.4 3.2 4.5 1.5]
 [6.9 3.1 4.9 1.5]
 [5.5 2.3 4.  1.3]
 [6.5 2.8 4.6 1.5]]

label: Iris-virginica
[[6.3 3.3 6.  2.5]
 [5.8 2.7 5.1 1.9]
 [7.1 3.  5.9 2.1]
 [6.3 2.9 5.6 1.8]
 [6.5 3.  5.8 2.2]]
```

4. Compute statistics with Numpy [7 points]

1. For each flower species compute the key statistics for `sepal width`:

- mean
- median
- standard deviation

Answer the following questions:

```
a) Which is the flower type with the largest mean value
for `sepal_width`?
b) Which is the flower type with the smallest median value
for `sepal_width`?
```

2. Compute the correlation matrix between `sepal_width` and `petal_length` for the three scpecies. Which is the species that shows the highest correlation among the two parameters?

In [ ]:

```
### Compute mean, median, and standard deviation for `sepal width` here:
#I create a unique label for each of the species.
unique_labels = np.unique(iris_label)

#Initialioze a dictionary to store statistics for each species
stats = {}
```

```
#Calculating the mean, median and standard deviation sepal width

for label in unique_labels:
    sepal_width =  iris_data[iris_label == label][:, 1] #'[:, 1]' is to s
   #getting the mean median and standard div. using numpy:
    mean_value = np.mean(sepal_width)
    median_value = np.median(sepal_width)
    std_dev_value = np.std(sepal_width)

    #then store the statistics in the dictionary
    stats[label] = {
        'mean': mean_value,
        'median': median_value,
        'std_dev': std_dev_value
    }

#Displaying the statistics
for label, values in stats.items():
    print(f"{label} - Sepal Width Statistics:")
    print(f"Mean: {values['mean']:.2f}, Median: {values['median']:.2f}, S

largest_mean_label = max(stats, key=lambda x: stats[x]['mean'])
smallest_median_label = min(stats, key=lambda x: stats[x]['median'])

print("Answers: ")
print(f"a) The flower species with the largest mean sepal width is: {larg
print(f"b) The flower species with the smallest median sepal width is: {s
```

```
Iris-setosa - Sepal Width Statistics:
Mean: 3.42, Median: 3.40, Std Dev: 0.38

Iris-versicolor - Sepal Width Statistics:
Mean: 2.77, Median: 2.80, Std Dev: 0.31

Iris-virginica - Sepal Width Statistics:
Mean: 2.97, Median: 3.00, Std Dev: 0.32

Answers:
a) The flower species with the largest mean sepal width is: Iris-setosa
b) The flower species with the smallest median sepal width is: Iris-versic
olor
```

In [ ]:

```
### Compute the correlation coefficients here



#Let's start with initializing a dictionary to store the correlation coef
correlations = {}

#We need to calculate the correlation between sepal width and petal lengh
for label in unique_labels:
    #Filter data for the current species
    sepal_width = iris_data[iris_label == label][:, 1] #Sepal width colum
    petal_lenght = iris_data[iris_label == label][:, 2] #Petal lenght col

    #Computing the correlation coeficient
    correlation = np.corrcoef(sepal_width, petal_lenght)[0, 1]


    #Storing the correlation in the dictionary
```

```
    correlations[label] = correlation

#Computes correlations
for label, corr in correlations.items():
    print(f"{label} - Correlation between Sepal Width and Petal Length: {
    # The `f"{label} - Correlation between Sepal Width and Petal Length:
    # the output to include the label name and the correlation value.
    # `{corr:.2f}` formats the correlation value to 2 decimal places for

#Finding the species with the higjest correlations
highest_corr_label = max(correlations, key=correlations.get)
print(" ")
print("Answer: ")
print(f"The species with the highest correlation between sepal width and
```

```
Iris-setosa - Correlation between Sepal Width and Petal Length: 0.18
Iris-versicolor - Correlation between Sepal Width and Petal Length: 0.56
Iris-virginica - Correlation between Sepal Width and Petal Length: 0.40

Answer:
The species with the highest correlation between sepal width and petal len
gth is: Iris-versicolor
```

5. How to create a new column from existing columns of a numpy array [7 points]

Create a new column for the flower volume V in `iris`, where volume is computing according to this formula (the assumption here is that the shape of the iris flower is approximately conical):

$$Volume = \frac{\pi \times SepalLength^2 \times PetalLength}{3}$$

In [ ]:

```
# Input
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iri
iris_2d = np.genfromtxt(url, delimiter=',', dtype='object')

# Write your solution here

#The first thing is to loading the dataset as a 2D arrat of objects (text

#Start by convertsing necessary columns to floayts for computation
sepal_lenght = iris_2d[:, 0].astype(float) #Sepal lenght (1st column)
petal_lenght = iris_2d[:, 2].astype(float) #Petal lenght (3rd column)

#Computing volume for each row using the forumala
#Volume = (pi * Sepal Lenght ^2 * petal lenght) /3

volumes = (np.pi * (sepal_lenght ** 2) * petal_lenght)/3

#Adding the new column to iris_2d by stacking it horizontally
#Conversting volumes to a column if ibjects to match dtype

volumes_column = volumes.reshape(-1, 1).astype(object) #This reshaped to
iris_with_volume = np.hstack([iris_2d, volumes_column])

#Result (first reowsn for visual verification)
```

```
print("Answer: ")
print(iris_with_volume[:5])
```

```
Answer:
[[b'5.1' b'3.5' b'1.4' b'0.2' b'Iris-setosa' 38.13265162927291]
 [b'4.9' b'3.0' b'1.4' b'0.2' b'Iris-setosa' 35.200498485922445]
 [b'4.7' b'3.2' b'1.3' b'0.2' b'Iris-setosa' 30.0723720777127]
 [b'4.6' b'3.1' b'1.5' b'0.2' b'Iris-setosa' 33.238050274980004]
 [b'5.0' b'3.6' b'1.4' b'0.2' b'Iris-setosa' 36.65191429188092]]
```

In [ ]:

```
#visualizing using pandas.
iris_volume_df = pd.DataFrame(iris_with_volume) #This is to convert Numpy
print(iris_volume_df.head())
```

```
        0       1       2       3               4      5
0  b'5.1'  b'3.5'  b'1.4'  b'0.2'  b'Iris-setosa' 38.13
1  b'4.9'  b'3.0'  b'1.4'  b'0.2'  b'Iris-setosa' 35.20
2  b'4.7'  b'3.2'  b'1.3'  b'0.2'  b'Iris-setosa' 30.07
3  b'4.6'  b'3.1'  b'1.5'  b'0.2'  b'Iris-setosa' 33.24
4  b'5.0'  b'3.6'  b'1.4'  b'0.2'  b'Iris-setosa' 36.65
```

UNIVERSITY OF OXFORD

Declaration of Authorship

First Assignment - Python for Data Science Course

I, Pedro Jorge Almada Melendez, confirm that the work I am submitting for the assignment titled "First-Assignment_Pedro_Almada" is my own work and has been completed by the requirements of the assignment.

Date: 13/11/2024

Signature: