HW1

February 4, 2021

1 HW1

Notebook containes 25 simple Numpy, Pandas, and Matplotlib exercises and aims to refresh your knowledge of those tools The questions are of different complexity, however they will all be graded equaly

2 Numpy

[64]:
```
import numpy as np
```

3 1

Write a function to create a numpy array using only the input: start, length, and step.

Use the function to create an array of length 10, starting from 5 and has a step of 3 between consecutive numbers

[15]:
```
def create_arr(start, length, step):
    return np.array([i*step + start for i in range(length)])

create_arr(5, 10, 3)
```

[15]:
```
array([ 5,  8, 11, 14, 17, 20, 23, 26, 29, 32])
```

4 #2

Convert numpy's `datetime64` object to datetime's `datetime` object

[27]:
```
from datetime import datetime
dt64 = np.datetime64('2019-05-14 22:10:10')
dt64.astype(datetime)
```

[27]:
```
datetime.datetime(2019, 5, 14, 22, 10, 10)
```

5 #3

Drop all nan values from a 1D numpy array

[35]:
```
a = np.array([1,2,3,np.nan,5,6,7,np.nan])
a[~np.isnan(a)]
```

[35]:
```
array([1., 2., 3., 5., 6., 7.])
```

6 #4

Compute the maximum for each row in the given array

[3]:
```
np.random.seed(100)
a = np.random.randint(1,10, [5,3])
print(a, '\n')
a.max(axis=1)
```

```
[[9 9 4]
 [8 8 1]
 [5 3 6]
 [3 3 3]
 [2 1 9]]
```

[3]:
```
array([9, 8, 6, 3, 9])
```

7 #5

Find the most frequent value of petal length (3rd column) in iris dataset

[63]:
```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = np.genfromtxt(url, delimiter=',', dtype='object')
names = ('sepallength', 'sepalwidth', 'petallength', 'petalwidth', 'species')

unique, counts = np.unique(iris[:, 2], return_counts=True)
unique[np.argmax(counts)]
```

[63]:
```
b'1.5'
```

8 #6

Sort the iris dataset based on sepallength column

[16]:
```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = np.genfromtxt(url, delimiter=',', dtype='object')
names = ('sepallength', 'sepalwidth', 'petallength', 'petalwidth', 'species')
iris[iris[:, 0].argsort()]
```

[16]:
```
array([[b'4.3', b'3.0', b'1.1', b'0.1', b'Iris-setosa'],
       [b'4.4', b'3.2', b'1.3', b'0.2', b'Iris-setosa'],
       [b'4.4', b'3.0', b'1.3', b'0.2', b'Iris-setosa'],
```

```
[b'4.4', b'2.9', b'1.4', b'0.2', b'Iris-setosa'],
[b'4.5', b'2.3', b'1.3', b'0.3', b'Iris-setosa'],
[b'4.6', b'3.6', b'1.0', b'0.2', b'Iris-setosa'],
[b'4.6', b'3.1', b'1.5', b'0.2', b'Iris-setosa'],
[b'4.6', b'3.4', b'1.4', b'0.3', b'Iris-setosa'],
[b'4.6', b'3.2', b'1.4', b'0.2', b'Iris-setosa'],
[b'4.7', b'3.2', b'1.3', b'0.2', b'Iris-setosa'],
[b'4.7', b'3.2', b'1.6', b'0.2', b'Iris-setosa'],
[b'4.8', b'3.0', b'1.4', b'0.1', b'Iris-setosa'],
[b'4.8', b'3.0', b'1.4', b'0.3', b'Iris-setosa'],
[b'4.8', b'3.4', b'1.9', b'0.2', b'Iris-setosa'],
[b'4.8', b'3.4', b'1.6', b'0.2', b'Iris-setosa'],
[b'4.8', b'3.1', b'1.6', b'0.2', b'Iris-setosa'],
[b'4.9', b'2.4', b'3.3', b'1.0', b'Iris-versicolor'],
[b'4.9', b'2.5', b'4.5', b'1.7', b'Iris-virginica'],
[b'4.9', b'3.1', b'1.5', b'0.1', b'Iris-setosa'],
[b'4.9', b'3.1', b'1.5', b'0.1', b'Iris-setosa'],
[b'4.9', b'3.1', b'1.5', b'0.1', b'Iris-setosa'],
[b'4.9', b'3.0', b'1.4', b'0.2', b'Iris-setosa'],
[b'5.0', b'3.5', b'1.3', b'0.3', b'Iris-setosa'],
[b'5.0', b'3.4', b'1.6', b'0.4', b'Iris-setosa'],
[b'5.0', b'3.3', b'1.4', b'0.2', b'Iris-setosa'],
[b'5.0', b'3.2', b'1.2', b'0.2', b'Iris-setosa'],
[b'5.0', b'3.5', b'1.6', b'0.6', b'Iris-setosa'],
[b'5.0', b'2.0', b'3.5', b'1.0', b'Iris-versicolor'],
[b'5.0', b'3.4', b'1.5', b'0.2', b'Iris-setosa'],
[b'5.0', b'2.3', b'3.3', b'1.0', b'Iris-versicolor'],
[b'5.0', b'3.6', b'1.4', b'0.2', b'Iris-setosa'],
[b'5.0', b'3.0', b'1.6', b'0.2', b'Iris-setosa'],
[b'5.1', b'3.8', b'1.9', b'0.4', b'Iris-setosa'],
[b'5.1', b'3.8', b'1.6', b'0.2', b'Iris-setosa'],
[b'5.1', b'2.5', b'3.0', b'1.1', b'Iris-versicolor'],
[b'5.1', b'3.5', b'1.4', b'0.2', b'Iris-setosa'],
[b'5.1', b'3.4', b'1.5', b'0.2', b'Iris-setosa'],
[b'5.1', b'3.5', b'1.4', b'0.3', b'Iris-setosa'],
[b'5.1', b'3.3', b'1.7', b'0.5', b'Iris-setosa'],
[b'5.1', b'3.7', b'1.5', b'0.4', b'Iris-setosa'],
[b'5.1', b'3.8', b'1.5', b'0.3', b'Iris-setosa'],
[b'5.2', b'4.1', b'1.5', b'0.1', b'Iris-setosa'],
[b'5.2', b'3.4', b'1.4', b'0.2', b'Iris-setosa'],
[b'5.2', b'3.5', b'1.5', b'0.2', b'Iris-setosa'],
[b'5.2', b'2.7', b'3.9', b'1.4', b'Iris-versicolor'],
[b'5.3', b'3.7', b'1.5', b'0.2', b'Iris-setosa'],
[b'5.4', b'3.0', b'4.5', b'1.5', b'Iris-versicolor'],
[b'5.4', b'3.9', b'1.7', b'0.4', b'Iris-setosa'],
[b'5.4', b'3.4', b'1.7', b'0.2', b'Iris-setosa'],
[b'5.4', b'3.4', b'1.5', b'0.4', b'Iris-setosa'],
```

```
[b'5.4', b'3.7', b'1.5', b'0.2', b'Iris-setosa'],
[b'5.4', b'3.9', b'1.3', b'0.4', b'Iris-setosa'],
[b'5.5', b'3.5', b'1.3', b'0.2', b'Iris-setosa'],
[b'5.5', b'2.6', b'4.4', b'1.2', b'Iris-versicolor'],
[b'5.5', b'4.2', b'1.4', b'0.2', b'Iris-setosa'],
[b'5.5', b'2.3', b'4.0', b'1.3', b'Iris-versicolor'],
[b'5.5', b'2.4', b'3.7', b'1.0', b'Iris-versicolor'],
[b'5.5', b'2.4', b'3.8', b'1.1', b'Iris-versicolor'],
[b'5.5', b'2.5', b'4.0', b'1.3', b'Iris-versicolor'],
[b'5.6', b'3.0', b'4.1', b'1.3', b'Iris-versicolor'],
[b'5.6', b'2.8', b'4.9', b'2.0', b'Iris-virginica'],
[b'5.6', b'3.0', b'4.5', b'1.5', b'Iris-versicolor'],
[b'5.6', b'2.5', b'3.9', b'1.1', b'Iris-versicolor'],
[b'5.6', b'2.7', b'4.2', b'1.3', b'Iris-versicolor'],
[b'5.6', b'2.9', b'3.6', b'1.3', b'Iris-versicolor'],
[b'5.7', b'2.6', b'3.5', b'1.0', b'Iris-versicolor'],
[b'5.7', b'2.9', b'4.2', b'1.3', b'Iris-versicolor'],
[b'5.7', b'2.8', b'4.1', b'1.3', b'Iris-versicolor'],
[b'5.7', b'4.4', b'1.5', b'0.4', b'Iris-setosa'],
[b'5.7', b'2.8', b'4.5', b'1.3', b'Iris-versicolor'],
[b'5.7', b'2.5', b'5.0', b'2.0', b'Iris-virginica'],
[b'5.7', b'3.8', b'1.7', b'0.3', b'Iris-setosa'],
[b'5.7', b'3.0', b'4.2', b'1.2', b'Iris-versicolor'],
[b'5.8', b'2.7', b'4.1', b'1.0', b'Iris-versicolor'],
[b'5.8', b'4.0', b'1.2', b'0.2', b'Iris-setosa'],
[b'5.8', b'2.6', b'4.0', b'1.2', b'Iris-versicolor'],
[b'5.8', b'2.8', b'5.1', b'2.4', b'Iris-virginica'],
[b'5.8', b'2.7', b'5.1', b'1.9', b'Iris-virginica'],
[b'5.8', b'2.7', b'3.9', b'1.2', b'Iris-versicolor'],
[b'5.8', b'2.7', b'5.1', b'1.9', b'Iris-virginica'],
[b'5.9', b'3.0', b'5.1', b'1.8', b'Iris-virginica'],
[b'5.9', b'3.0', b'4.2', b'1.5', b'Iris-versicolor'],
[b'5.9', b'3.2', b'4.8', b'1.8', b'Iris-versicolor'],
[b'6.0', b'2.9', b'4.5', b'1.5', b'Iris-versicolor'],
[b'6.0', b'2.7', b'5.1', b'1.6', b'Iris-versicolor'],
[b'6.0', b'3.0', b'4.8', b'1.8', b'Iris-virginica'],
[b'6.0', b'3.4', b'4.5', b'1.6', b'Iris-versicolor'],
[b'6.0', b'2.2', b'4.0', b'1.0', b'Iris-versicolor'],
[b'6.0', b'2.2', b'5.0', b'1.5', b'Iris-virginica'],
[b'6.1', b'3.0', b'4.9', b'1.8', b'Iris-virginica'],
[b'6.1', b'2.6', b'5.6', b'1.4', b'Iris-virginica'],
[b'6.1', b'2.8', b'4.0', b'1.3', b'Iris-versicolor'],
[b'6.1', b'2.9', b'4.7', b'1.4', b'Iris-versicolor'],
[b'6.1', b'2.8', b'4.7', b'1.2', b'Iris-versicolor'],
[b'6.1', b'3.0', b'4.6', b'1.4', b'Iris-versicolor'],
[b'6.2', b'2.2', b'4.5', b'1.5', b'Iris-versicolor'],
[b'6.2', b'2.9', b'4.3', b'1.3', b'Iris-versicolor'],
```

```
[b'6.2', b'3.4', b'5.4', b'2.3', b'Iris-virginica'],
[b'6.2', b'2.8', b'4.8', b'1.8', b'Iris-virginica'],
[b'6.3', b'2.5', b'4.9', b'1.5', b'Iris-versicolor'],
[b'6.3', b'2.7', b'4.9', b'1.8', b'Iris-virginica'],
[b'6.3', b'2.5', b'5.0', b'1.9', b'Iris-virginica'],
[b'6.3', b'3.3', b'4.7', b'1.6', b'Iris-versicolor'],
[b'6.3', b'2.8', b'5.1', b'1.5', b'Iris-virginica'],
[b'6.3', b'3.3', b'6.0', b'2.5', b'Iris-virginica'],
[b'6.3', b'2.3', b'4.4', b'1.3', b'Iris-versicolor'],
[b'6.3', b'3.4', b'5.6', b'2.4', b'Iris-virginica'],
[b'6.3', b'2.9', b'5.6', b'1.8', b'Iris-virginica'],
[b'6.4', b'2.8', b'5.6', b'2.2', b'Iris-virginica'],
[b'6.4', b'2.8', b'5.6', b'2.1', b'Iris-virginica'],
[b'6.4', b'3.1', b'5.5', b'1.8', b'Iris-virginica'],
[b'6.4', b'3.2', b'4.5', b'1.5', b'Iris-versicolor'],
[b'6.4', b'3.2', b'5.3', b'2.3', b'Iris-virginica'],
[b'6.4', b'2.9', b'4.3', b'1.3', b'Iris-versicolor'],
[b'6.4', b'2.7', b'5.3', b'1.9', b'Iris-virginica'],
[b'6.5', b'3.0', b'5.8', b'2.2', b'Iris-virginica'],
[b'6.5', b'3.0', b'5.5', b'1.8', b'Iris-virginica'],
[b'6.5', b'3.0', b'5.2', b'2.0', b'Iris-virginica'],
[b'6.5', b'2.8', b'4.6', b'1.5', b'Iris-versicolor'],
[b'6.5', b'3.2', b'5.1', b'2.0', b'Iris-virginica'],
[b'6.6', b'2.9', b'4.6', b'1.3', b'Iris-versicolor'],
[b'6.6', b'3.0', b'4.4', b'1.4', b'Iris-versicolor'],
[b'6.7', b'3.1', b'4.7', b'1.5', b'Iris-versicolor'],
[b'6.7', b'3.1', b'5.6', b'2.4', b'Iris-virginica'],
[b'6.7', b'2.5', b'5.8', b'1.8', b'Iris-virginica'],
[b'6.7', b'3.0', b'5.0', b'1.7', b'Iris-versicolor'],
[b'6.7', b'3.1', b'4.4', b'1.4', b'Iris-versicolor'],
[b'6.7', b'3.3', b'5.7', b'2.5', b'Iris-virginica'],
[b'6.7', b'3.0', b'5.2', b'2.3', b'Iris-virginica'],
[b'6.7', b'3.3', b'5.7', b'2.1', b'Iris-virginica'],
[b'6.8', b'3.2', b'5.9', b'2.3', b'Iris-virginica'],
[b'6.8', b'2.8', b'4.8', b'1.4', b'Iris-versicolor'],
[b'6.8', b'3.0', b'5.5', b'2.1', b'Iris-virginica'],
[b'6.9', b'3.1', b'5.4', b'2.1', b'Iris-virginica'],
[b'6.9', b'3.1', b'5.1', b'2.3', b'Iris-virginica'],
[b'6.9', b'3.1', b'4.9', b'1.5', b'Iris-versicolor'],
[b'6.9', b'3.2', b'5.7', b'2.3', b'Iris-virginica'],
[b'7.0', b'3.2', b'4.7', b'1.4', b'Iris-versicolor'],
[b'7.1', b'3.0', b'5.9', b'2.1', b'Iris-virginica'],
[b'7.2', b'3.0', b'5.8', b'1.6', b'Iris-virginica'],
[b'7.2', b'3.2', b'6.0', b'1.8', b'Iris-virginica'],
[b'7.2', b'3.6', b'6.1', b'2.5', b'Iris-virginica'],
[b'7.3', b'2.9', b'6.3', b'1.8', b'Iris-virginica'],
[b'7.4', b'2.8', b'6.1', b'1.9', b'Iris-virginica'],
```

```
       [b'7.6', b'3.0', b'6.6', b'2.1', b'Iris-virginica'],
       [b'7.7', b'2.8', b'6.7', b'2.0', b'Iris-virginica'],
       [b'7.7', b'2.6', b'6.9', b'2.3', b'Iris-virginica'],
       [b'7.7', b'3.8', b'6.7', b'2.2', b'Iris-virginica'],
       [b'7.7', b'3.0', b'6.1', b'2.3', b'Iris-virginica'],
       [b'7.9', b'3.8', b'6.4', b'2.0', b'Iris-virginica']], dtype=object)
```

9 7

What is the value of second longest *unique* petallength of species `setosa`

[34]:
```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = np.genfromtxt(url, delimiter=',', dtype='object')
names = ('sepallength', 'sepalwidth', 'petallength', 'petalwidth', 'species')
np.unique(iris[iris[:, 4] == b'Iris-setosa'][:, 2])[-2]
```

[34]:
```
b'1.7'
```

10 #8

Replace all occurrences of nan with 0 in numpy array

[54]:
```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris_2d = np.genfromtxt(url, delimiter=',', dtype='float', usecols=[0,1,2,3])
np.random.seed(100)
iris_2d[np.random.randint(150, size=20), np.random.randint(4, size=20)] = np.nan
iris_2d[np.isnan(iris_2d)] = 0
iris_2d
```

[54]:
```
array([[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],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 0. , 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [0. , 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
```

```
[5.1, 3.8, 1.5, 0.3],
[5.4, 3.4, 1.7, 0.2],
[5.1, 3.7, 1.5, 0.4],
[4.6, 3.6, 1. , 0.2],
[5.1, 3.3, 1.7, 0.5],
[4.8, 3.4, 0. , 0. ],
[5. , 3. , 1.6, 0.2],
[5. , 3.4, 1.6, 0.4],
[5.2, 3.5, 1.5, 0.2],
[5.2, 3.4, 1.4, 0.2],
[4.7, 3.2, 1.6, 0.2],
[4.8, 3.1, 1.6, 0.2],
[5.4, 3.4, 1.5, 0.4],
[5.2, 4.1, 1.5, 0.1],
[5.5, 4.2, 1.4, 0.2],
[0. , 3.1, 1.5, 0.1],
[5. , 3.2, 1.2, 0.2],
[5.5, 3.5, 1.3, 0.2],
[4.9, 3.1, 1.5, 0.1],
[4.4, 3. , 1.3, 0.2],
[5.1, 3.4, 1.5, 0.2],
[5. , 3.5, 1.3, 0.3],
[4.5, 2.3, 1.3, 0.3],
[4.4, 3.2, 1.3, 0.2],
[5. , 3.5, 1.6, 0.6],
[5.1, 3.8, 1.9, 0.4],
[4.8, 3. , 1.4, 0.3],
[5.1, 3.8, 1.6, 0.2],
[4.6, 3.2, 1.4, 0.2],
[5.3, 3.7, 1.5, 0.2],
[5. , 3.3, 1.4, 0.2],
[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, 0. , 4. , 1.3],
[6.5, 2.8, 4.6, 1.5],
[5.7, 2.8, 4.5, 1.3],
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 0. , 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 0. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
```

```
[5.6, 3. , 4.5, 0. ],
[5.8, 2.7, 0. , 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[0. , 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[0. , 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 0. , 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 0. , 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[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, 0. ],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 0. ],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
```

```
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 0. , 5.5, 1.8],
       [6. , 3. , 4.8, 0. ],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 0. ],
       [6.7, 3.3, 5.7, 0. ],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```

11 9

Write a function to calculate the difference between the maximum and the minimum values of a given array along the second axis, using NumPy

[63]:
```
def calc_min_max_diff(x):
    return x.max(axis=1) - x.min(axis=1)

calc_min_max_diff(np.arange(12).reshape((2, 6)))
```

[63]:
```
array([5, 5])
```

12 #10

Write a function to compute the mean, standard deviation, and variance of a given array using NumPy. Do each operation in two different ways (by formula and using inbuilt function), and check using NumPy's `allclose` function that the values are similar. Use the function on an NumPy array of your choice.

[100]:
```
#your code here

def calc_stats(x):
    m1 = x.mean()
    m2 = x.sum() / x.shape[0]
    assert np.allclose(m1, m2)
    print("\nMean: ", m1)
    s1 = x.std()
    s2 = np.sqrt(np.square(np.abs(x - m1)).sum() / x.shape[0])
    assert np.allclose(s1, s2)
    print("\nstd: ", s1)
    v1 = x.var()
    v2 = np.square(x - m1).sum() / x.shape[0]
    assert np.allclose(v1, v2)
    print("\nvariance: ", v1)

calc_stats(np.arange(6))
```

```
Mean:  2.5

std:  1.707825127659933

variance:  2.9166666666666665
```

[104]:
```
calc_stats(np.linspace(1.0, 100.0, num=10))
```

```
Mean:  50.5

std:  31.595094555959157

variance:  998.25
```

13 Pandas

[1]:
```
import pandas as pd
```

14 Question #1

Print the unique values in a pandas series `series`, and the number of appearances in the series. Use one line of code.

[113]:
```
series = pd.Series(np.take(list('abcdefghijklmnop'), np.random.randint(16,␣
 ↪size=500)))
series.value_counts()
```

[113]:
```
i    39
k    37
f    36
p    34
c    34
b    33
l    33
a    32
o    31
n    29
e    29
h    28
m    27
d    26
g    26
j    26
dtype: int64
```

15 Question #2

Stack two pandas series: 1. horizontally - final shape (5,2) 2. Vertically - final shape (10,1)

[128]:
```
series_q2_1 = pd.Series(['a', 'b', 'c', 'd', 'e'])
series_q2_2 = pd.Series([1,2,3,4,5])
print("Horizontally")
print(pd.concat([series_q2_1, series_q2_2], axis=1))
print("Vertically")
print(pd.concat([series_q2_1, series_q2_2], axis=0))
```

```
Horizontally
   0  1
0  a  1
1  b  2
2  c  3
3  d  4
```

```
4  e  5
Vertically
0    a
1    b
2    c
3    d
4    e
0    1
1    2
2    3
3    4
4    5
dtype: object
```

16 Question #3

Calculate the number of characters in each value in a pandas series

[3]:
```
series_q3 = pd.Series(['This', 'is', 'a', 'string', 'with', 'words'])
series_q3.str.len()
```

[3]:
```
0    4
1    2
2    1
3    6
4    4
5    5
dtype: int64
```

17 Question #4

Get the day of month, week number, day of year and day of week from a pandas series (Hint: you can use the dateutil.parser package)

[43]:
```
series_q4 = pd.Series(['01 Jan 2019', '02-02-2019', '20150303', '2013/04/04',␣
 ↪'2012-05-05', '2013-06-06T12:20'])
date = pd.DatetimeIndex(series_q4)
pd.DataFrame({
    "date": series_q4,
    "day": date.day,
    "week": date.week,
    "dayofyear": date.dayofyear,
    "dayofweek": date.dayofweek,
})
```

[43]:

	date	day	week	dayofyear	dayofweek
0	01 Jan 2019	1	1	1	1
1	02-02-2019	2	5	33	5

```
2          20150303    3    10         62          1
3        2013/04/04    4    14         94          3
4        2012-05-05    5    18        126          5
5  2013-06-06T12:20    6    23        157          3
```

18 Question #5

Out of the following series, how many values are "Apples"? Remember to use a vector-like calculation

[55]:
```
import random
nouns = ["Apples", "Bananas", "Peaches"]
series_q5 = pd.Series([nouns[random.randrange(0, len(nouns))] for i in␣
 ↪range(500)])
series_q5.value_counts()[nouns[0]]
```

[55]:
```
160
```

19 Question #6

Change the first character of each word to upper case in each word of the given pandas series

[61]:
```
series_q6 = pd.Series(['we', 'are', 'at', 'the', 'campus', 'learning',␣
 ↪'pandas'])
series_q6.str.capitalize()
```

[61]:
```
0          We
1         Are
2          At
3         The
4      Campus
5    Learning
6      Pandas
dtype: object
```

20 Question #7

Turn the given array into a pandas dataframe with 5 rows and 3 columns

[71]:
```
arr = np.arange(15)
pd.DataFrame(arr.reshape(5, 3))
```

[71]:
```
    0   1   2
0   0   1   2
1   3   4   5
2   6   7   8
3   9  10  11
4  12  13  14
```

21 Question #8

Read the data from https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data into a pandas dataframe. Do not download the file.

Then print the amount of columns of each datatype.

[104]:
```
df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/
↪iris/iris.data", header=None)
df.dtypes.value_counts()
```

[104]:
```
float64    4
object     1
dtype: int64
```

22 Question #9

How many of the colums in the dataframe from previous question has missing values?

[105]:
```
sum(df.isna().sum() > 0)
```

[105]:
```
0
```

23 Question #10

[165]:
```
df = pd.read_csv('../data/hw1/titanic_train.csv', index_col=None)
df.head()
```

[165]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Group df by pclass and calculate survival rate and mean, max, min fare for each pclass group

[168]:
```
df.groupby("Pclass").agg({
    "Fare": ["mean", "min", "max"],
    "Survived": {
        "rate": lambda x: x.value_counts(normalize=True)[1],
    },
})
```

[168]:

	Fare			Survived
	mean	min	max	rate
Pclass				
1	84.154687	0.0	512.3292	0.629630
2	20.662183	0.0	73.5000	0.472826
3	13.675550	0.0	69.5500	0.242363

24 Matplotlib

[169]:
```
import matplotlib.pyplot as plt
import warnings
import pandas as pd
import numpy as np

warnings.filterwarnings("ignore", category=RuntimeWarning, module="matplotlib")
%matplotlib inline
```

Let's prepare the data:

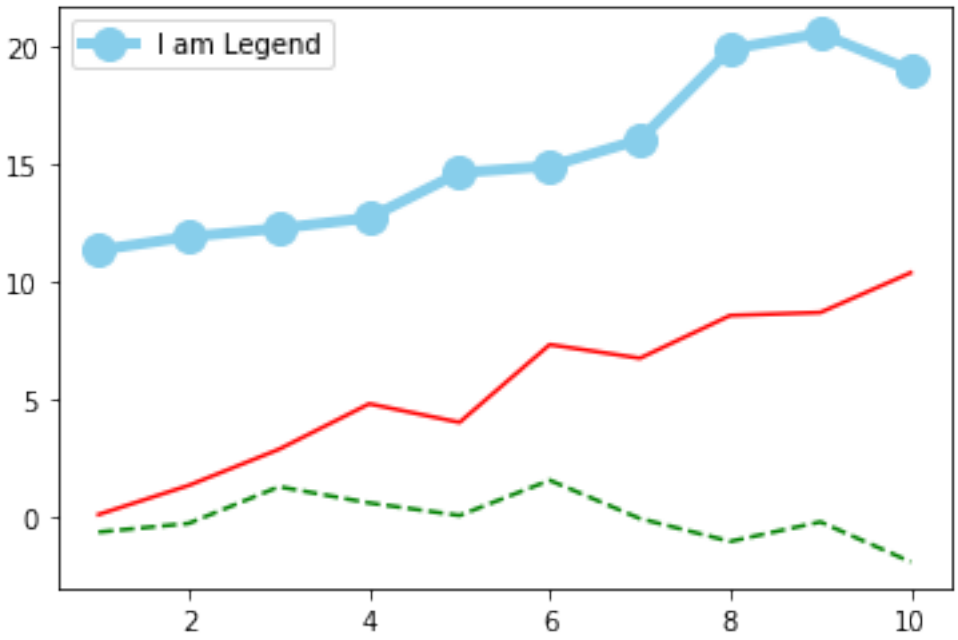
[170]:
```
df = pd.DataFrame({'x': range(1,11), \
                   'y_1': np.random.randn(10), \
                   'y_2': np.random.randn(10) + range(1,11), \
                   'y_3': np.random.randn(10) + range(11,21) })
print(df)
```

	x	y_1	y_2	y_3
0	1	-0.645142	0.099865	11.341369
1	2	-0.276158	1.339071	11.912430
2	3	1.282321	2.874565	12.266391
3	4	0.594372	4.807342	12.711665
4	5	0.060737	4.017617	14.632378
5	6	1.556410	7.324465	14.910755
6	7	-0.070918	6.745680	16.034678
7	8	-1.043214	8.563203	19.888963
8	9	-0.212634	8.690443	20.553221
9	10	-1.912156	10.380730	18.992138

25 Question #1

From the given data, create a line plot (plot all data points, connected by a line) with the following trends on it: 1. y_1 according to x, color green, no marker, dashed line with width 2 2. y_2 according to x, color red, no marker, regular line with width 2 3. y_3 according to x, color skyblue, regular line of width 4, with blue 'o' markers of size 12, and a label displayed as legend "I am Legend".

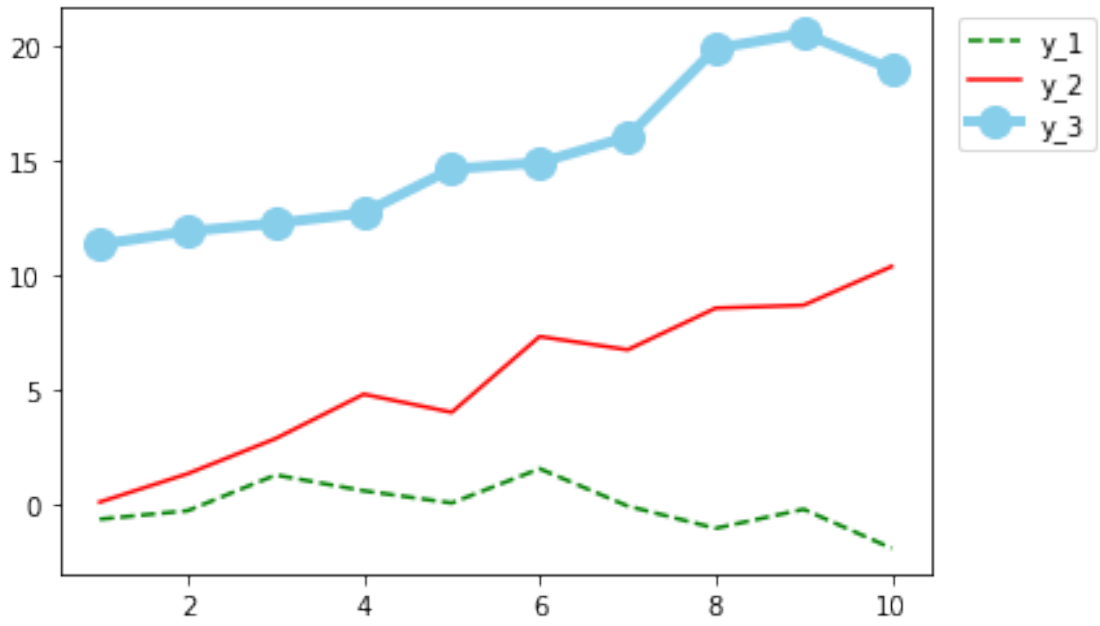
[273]:
```
plt.plot(df.x, df.y_1, "g--")
plt.plot(df.x, df.y_2, "r")
plt.plot(df.x, df.y_3, color="skyblue", marker='o', linewidth=4, markersize=12,␣
 ↪label="I am Legend")
plt.legend();
```



26 Question #2

Take the plot from Question #1, labels all lines with 'y_1', 'y_2', 'y_3' and put the legend on the upper right corner of the plot, outside the plot itself - use `loc` parameter

[272]:
```
plt.plot(df.x, df.y_1, "g--", label="y_1")
plt.plot(df.x, df.y_2, "r", label="y_2")
plt.plot(df.x, df.y_3, color="skyblue", marker='o', linewidth=4, markersize=12,␣
 ↪label="y_3")
plt.legend(loc=(1.03, 0.75));
```

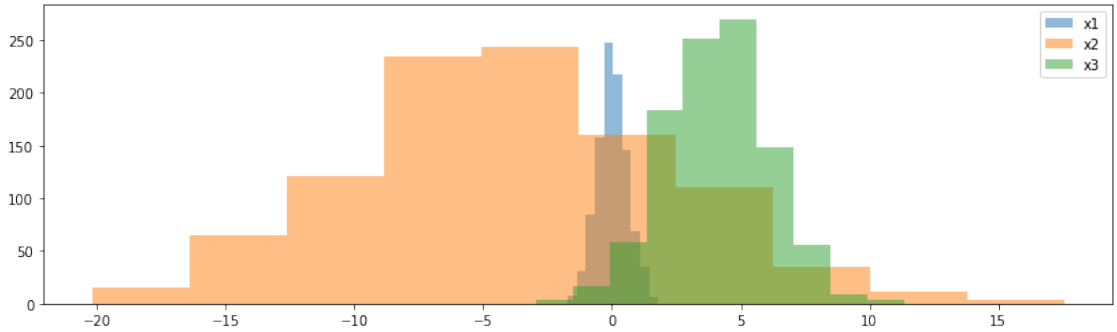


27 Question #3

Let's prepare the data:

[248]:
```
x1 = np.random.normal(0, 0.6, 1000)
x2 = np.random.normal(-4, 6, 1000)
x3 = np.random.normal(4, 2, 1000)
```

[274]:
```
plt.figure(figsize=(14, 4))
plt.hist(x1, alpha=0.5, label="x1")
plt.hist(x2, alpha=0.5, label="x2")
plt.hist(x3, alpha=0.5, label="x3")
plt.legend();
```



Print one plot, containing the distribution plots of each of the 3 given distribution. Enlarge the graph and pass transparency parameters so that all of the parts of each distribution are visible. Add a matching legend.

28 Question #4

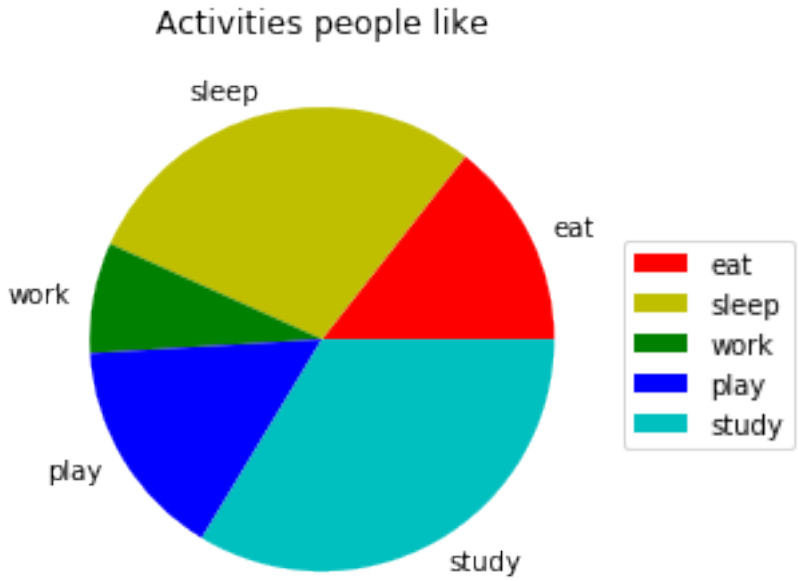
Let's prepare the data:

[275]:
```
activities = ['eat', 'sleep', 'work', 'play', 'study']

slices = [15, 30, 8, 16, 35]

colors = ['r', 'y', 'g', 'b', 'c']
```

[292]:
```
plt.pie(slices, colors = ['r', 'y', 'g', 'b', 'c'], labels = ['eat', 'sleep',␣
↪'work', 'play', 'study'])
plt.title("Activities people like")
plt.legend(loc=(1.02, 0.31));
```



Create a pie-chart using the given data, to show how many people like each activity. - Add a legend and an appropriate title. - Locate the legend and the title so that they are not on top of the plot.

29 Question #5

Correct the plot from Question #4. Add more information to the pie chart so it delivers much more information - a label next to each pie-slice (if you haven't already done so), and the percentage out

of all on the pie-slice itself.

[300]:
```
plt.pie(slices, colors = ['r', 'y', 'g', 'b', 'c'], labels = ['eat', 'sleep',␣
 ↪'work', 'play', 'study'], autopct='%.1f%%')
plt.title("Activities people like")
plt.legend(loc=(1.02, 0.31));
```