COMP7035

Python for Data Analytics and Artificial Intelligence

Seaborn

Renjie Wan, Xue Wei

24/10/2022

香港浸會大學
HONG KONG BAPTIST UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE
計算機科學系

What we will learn?

Topic		Hours
I.	Python Fundamentals A. Program control and logic B. Data types and structures C. Function D. File I/O	12
II.	Numerical Computing and Data Visualization Tools and libraries such as A. NumPy **B. Matplotlib** C. Seaborn	9
III.	Exploratory Data Analysis (EDA) with Python Tools and libraries such as A. Pandas B. Sweetviz	9
IV.	Artificial Intelligence and Machine Learning with Python Tools and libraries such as A. Keras B. Scikit-learn	9

Comments for your test

5. Write a Python program that prints all the numbers from 0 to 6 except 3 and 6.

```
for x in range(6):
    if (x == 3 or x==6):
        continue
    print(x)
print("\n")
```

Correct

```
num_list = []
for i in range(7):
    if(i!=3 and i!=6): Correct
        num_list.append(i)
print(num_list)
```

Correct

Comments for your test

```
num_list = []
for i in range(7):
    if(i!=3 or i!=6):
        num_list.append(i)
print(num_list)
```

```
for i in range(7):
    if i == 0:
        print(i)
    if i%3!=0:
```

```
list1 = [i for i in range(7)]
print(list1)
for i in list1:
    print(i)
    if i%3 == 0:
        list1.pop(i)
print(list1)
```

```
for i in range(6):
    if i%3!=0: Wrong!!
        print(i)
```

```
for i in range(6):
    if i%3!=0:
        print(i)
```

```
for i in range(7):
    if i%3 ==0: Wrong!!
        break
    else:
        print(a)
```

Comments for your test

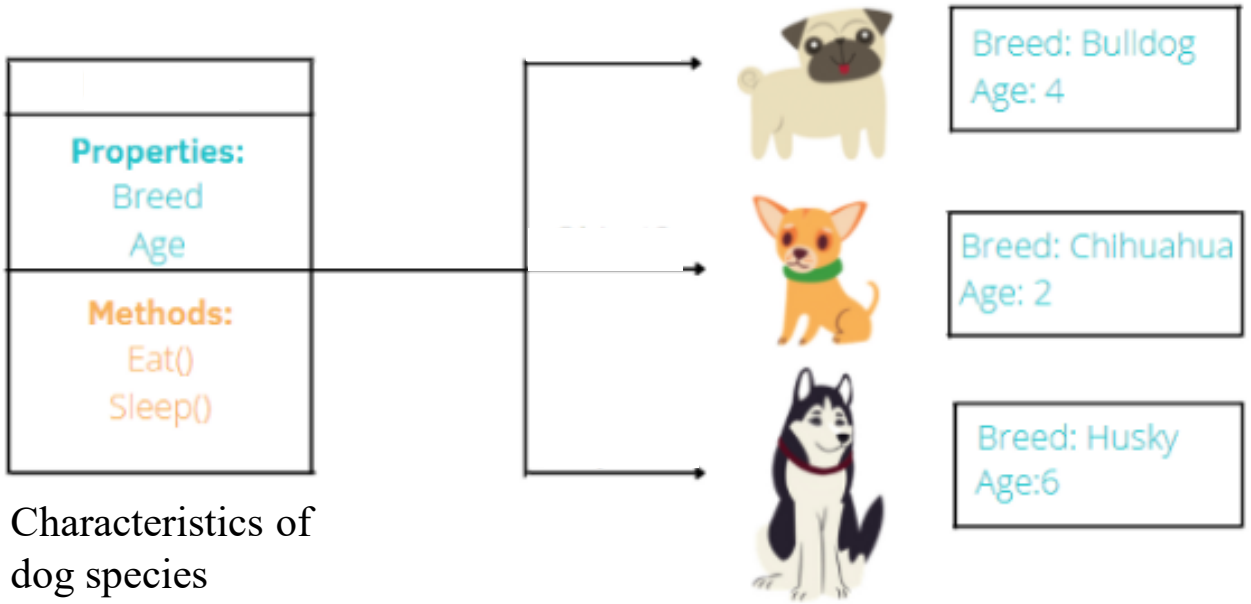
3. What is printed by the Python code?

```
for z in [2, 4, 7, 9]:
    print(z - 1)
```

Result: 1 3 6 8

Class

- Almost everything in Python is an object, with its properties and methods.
- A Class is like an object constructor, or a "blueprint" for creating objects.

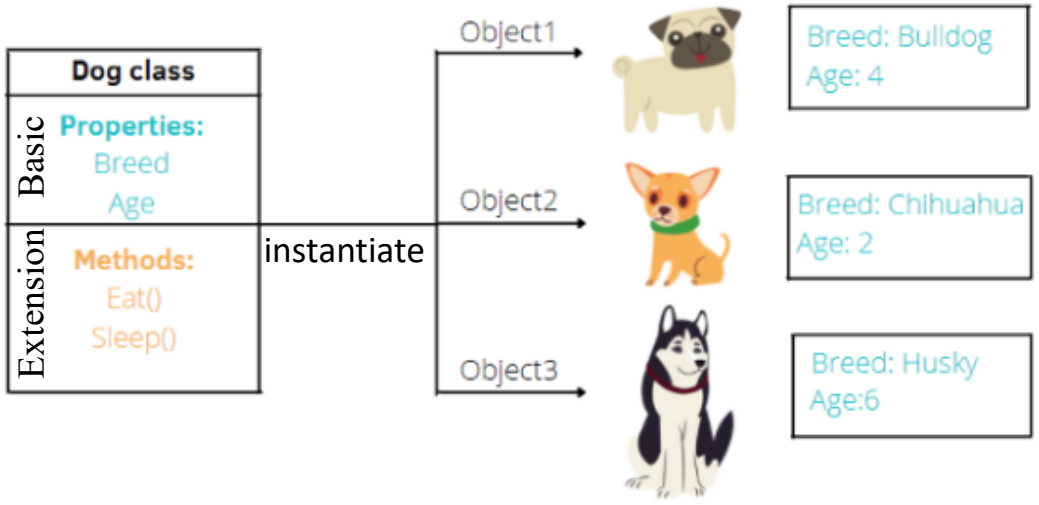


Characteristics of dog species

Different kinds of dogs with those characteristics

Class

- Almost everything in Python is an object, with its properties and methods.
- self represents the instance of the class. By using the “self” we can access the attributes and methods of the class in python.



Class

- Classes have a function called __init__(), which is always executed when the class is being initiated.
- Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

```
class Dog:
  def __init__(self, breed, age):
    self.breed = breed
    self.age = age
  def Eating(self, times):
    print(self.breed + ' Eating ' + str(times) + " times every day")
  def Sleeping(self, hours):
    print(self.name + ' Sleeping ' + str(hours) + " hours erery day")
```

define basic properties

define extension functions

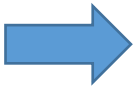
How to use class in Python

```
class Dog:
  def __init__(self, breed, age):
    self.breed = breed
    self.age = age
  def Eating(self, times):
    print(self.breed + ' Eating ' + str(times) + " times every day")
  def Sleeping(self, hours):
    print(self.name + ' Sleeping ' + str(hours) + " hours erery day")
```

```
object1 = Dog("Bulldog", 4)
object2 = Dog("Chihuahua", 2)
object3 = Dog("Husky", 6)
```

instantiate

```
object1.Eating(5)
object2.Eating(4)
object3.Eating(3)
```



```
Bulldog Eating 5 times every day
Chihuahua Eating 4 times every day
Husky Eating 3 times every day
```

Result

A small exercise for you

- Write a class for Person
- Basic Properties: Age, Name, Sex.
- Extension Properties: Working, Sleepings, just consider the hours they work and sleep everyday.
- Then, instantiate the two classes into to different persons

We have used it before

```
import matplotlib.pyplot as plt

fig = plt.figure()
```

You have instantiate a class by using plt.figure()

```
# Plot first figure
ax = fig.subplots()
```

Now you can call the subplots function inside plt.figure()

```
ax.plot([0, 1], [0, 1])
ax.plot([1, 2], [0, 1])
plt.show()
```

We have used it before

numpy.ndarray

```
class numpy.ndarray(shape, dtype=float, buffer=None, offset=0, strides=None, order=None)
```

np.array is a function that can help you instantiate the ndarray classes

```
a = np.array([1, 2, 3, 4, 5, 6])
b = np.array([3, 4, 5, 6, 7, 6])
```

You create two array classes using this way.
They are all arrays. They have similarities, while they still present different characteristics.

Just like the Dog defined in previous slides.

The class you may encounter

```
CLASS  torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1,
       groups=1, bias=True, padding_mode='zeros', device=None, dtype=None) [SOURCE]
```

m = nn.Conv2d(16, 33, 3, stride=2)

```
CLASS  torch.optim.SGD(params, lr=<required parameter>, momentum=0, dampening=0,
       weight_decay=0, nesterov=False, *, maximize=False, foreach=None) [SOURCE]
```

optimizer = torch.optim.SGD(model.parameters(), lr=0.1, momentum=0.9)

Numpy Array Padding

- **numpy.pad(*array, pad_width, mode='constant', **kwargs*)**
- This function is used to pad an array.

```
import numpy as np
a = [[1, 2], [3, 4]]
print(a)
a_pad = np.pad(a, ((2, 3), (3, 3)), 'constant')
print(a_pad)
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Pad three columns of zero along the horizontal direction, before the second axis

Pad three columns of zero along the horizontal direction, after the second axis

Pad two rows of zero along the vertical direction, before the first axis

```
[[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 1 2 0 0 0]
 [0 0 0 3 4 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]
```

Pad three rows of zero along the vertical direction, after the first axis

A small exercise for you

```
[[0 0 0 0 0 0 0 0]
 [0 0 0 1 2 0 0 0]
 [0 0 0 3 4 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]
```

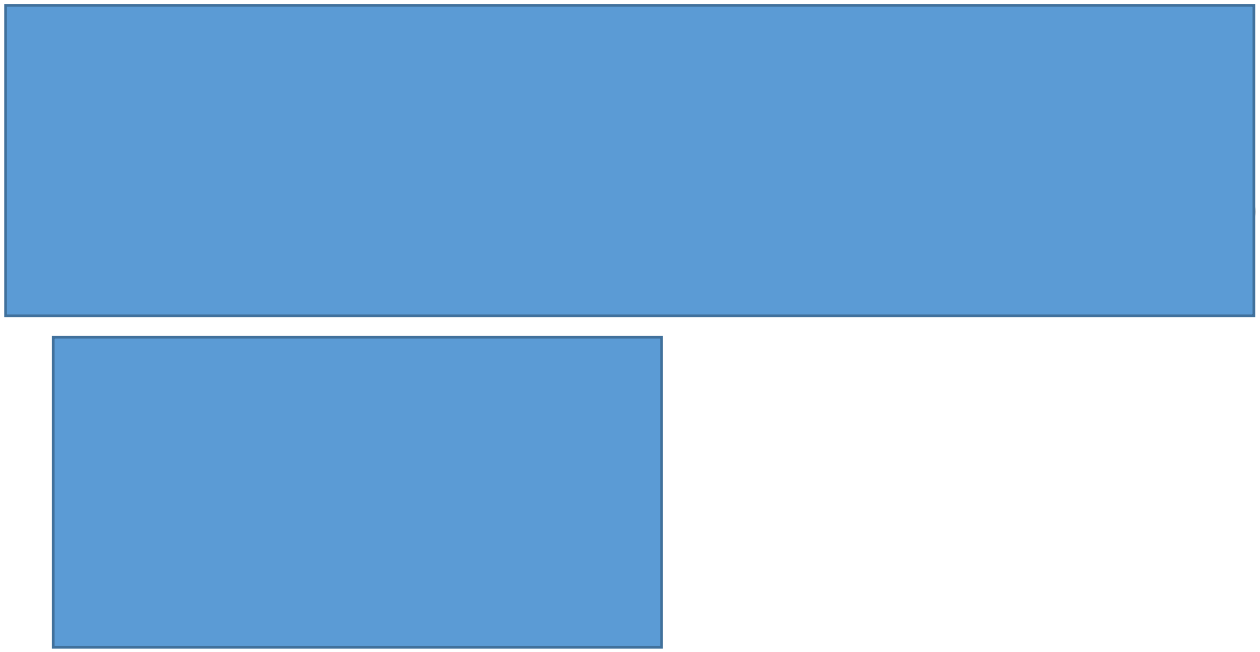
1. Create a matrix like the left
2. Create a matrix like the below

```
[[1. 0. 0. 0.]
 [0. 2. 0. 0.]
 [0. 0. 3. 0.]
 [0. 0. 0. 4.]]
```

A small exercise for you

```
[[0 0 0 0 0 0 0 0]
 [0 0 0 1 2 0 0 0]
 [0 0 0 3 4 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]
```

Create a matrix like the left



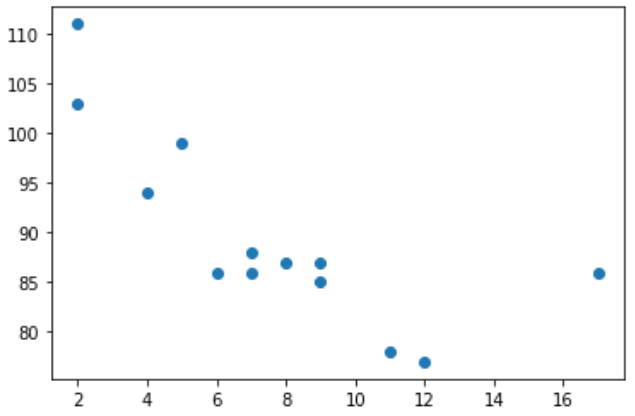
PyPlot: Scatter

- The scatter() function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis:

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

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.show()
```



PyPlot: Scatter

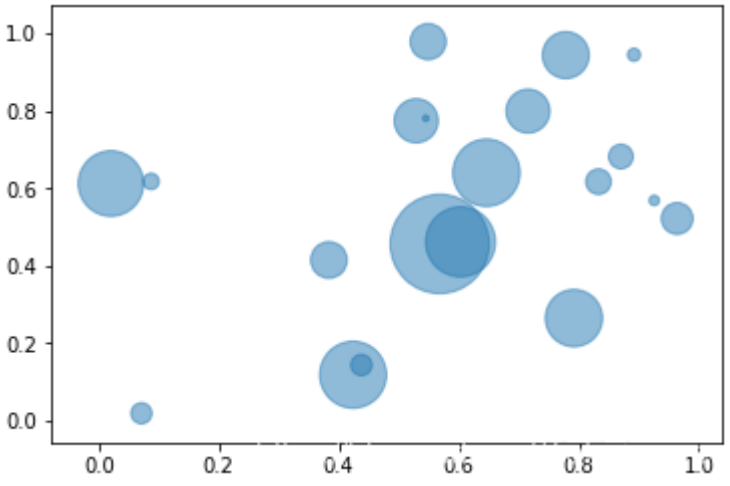
- We can also control the areas and their colors of each dot
- In this page, area is a random number array

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

np.random.seed(0)
x=np.random.rand(20)
y=np.random.rand(20)

area=(50*np.random.rand(20))**2

plt.scatter(x,y,s=area,alpha=0.5)
plt.show()
```



PyPlot: Scatter

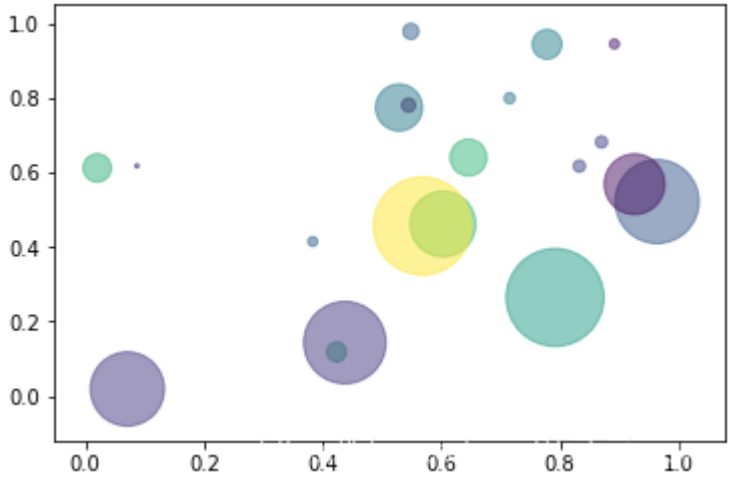
- In this page, color also becomes a random number array.

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

np.random.seed(0)
x=np.random.rand(20)
y=np.random.rand(20)

colors=np.random.rand(20)
area=(50*np.random.rand(20))**2

plt.scatter(x,y,s=area,c=colors,alpha=0.5)
plt.show()
```



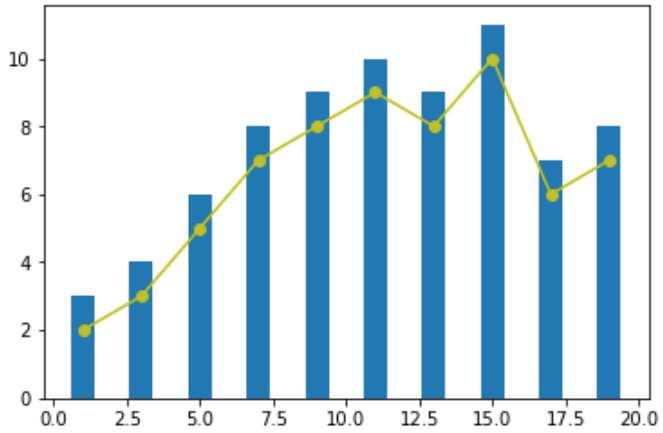
What is seaborn?

- You can simply declare seaborn style in your code

```
import matplotlib.pyplot as plt

x = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
y_bar = [3, 4, 6, 8, 9, 10, 9, 11, 7, 8]
y_line = [2, 3, 5, 7, 8, 9, 8, 10, 6, 7]

plt.bar(x, y_bar)
plt.plot(x, y_line, '-o', color='y')
```

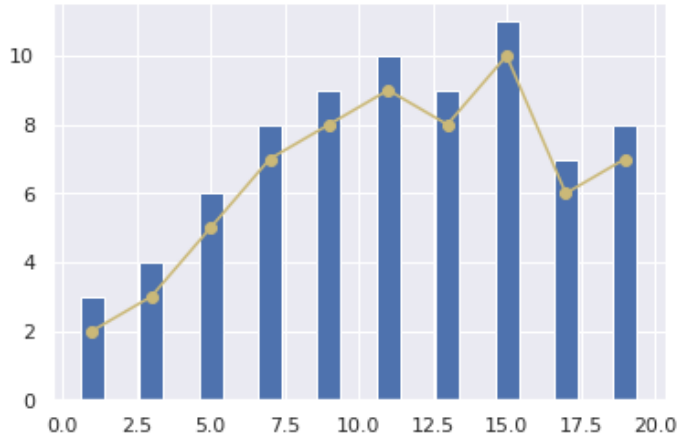



```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set()  # Declare seborn style

x = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
y_bar = [3, 4, 6, 8, 9, 10, 9, 11, 7, 8]
y_line = [2, 3, 5, 7, 8, 9, 8, 10, 6, 7]

plt.bar(x, y_bar)
plt.plot(x, y_line, '-o', color='y')
```



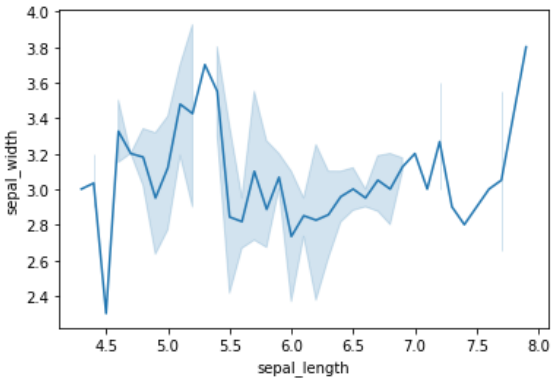
What is seaborn?

- Seaborn is a library for making statistical graphics in Python.
- Seaborn comes with many built-in datasets.
- You can find more in this website:
  - https://github.com/mwaskom/seaborn-data
- That means you don't have to spend a whole lot of your time finding the right dataset and cleaning it up to make Seaborn-ready

```
import seaborn as sns

# loading dataset
data = sns.load_dataset("iris")

# draw lineplot
sns.lineplot(x="sepal_length", y="sepal_width", data=data)
```



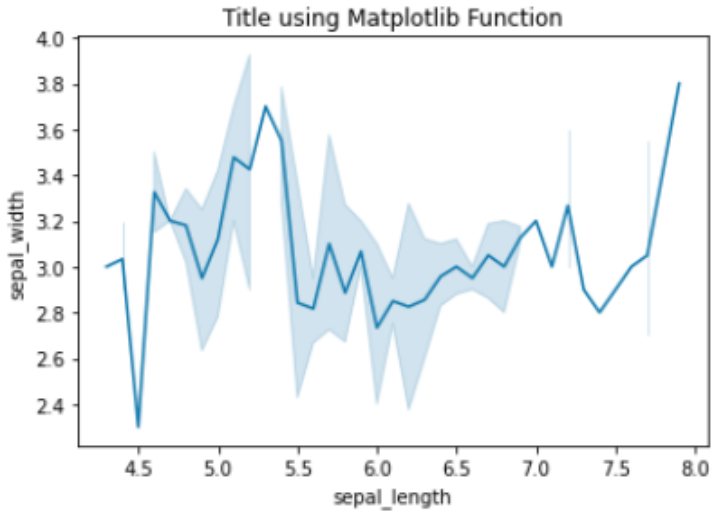
sepal_length	sepal_width	petal_length	petal_width	species

What is seaborn?

Iris flower data set

From Wikipedia, the free encyclopedia

The ***Iris* flower data set** or **Fisher's *Iris* data set** is a multivariate data set used and made famous by the British statistician and biologist Ronald Fisher in his 1936 paper *The use of multiple measurements in taxonomic problems* as an example of linear discriminant analysis.[1] It is sometimes called **Anderson's *Iris* data set** because Edgar Anderson collected the data to quantify the morphologic variation of *Iris* flowers of three related species.[2] Two of the three species were collected in the Gaspé Peninsula "all from the same pasture, and picked on the same day and measured at the same time by the same person with the same apparatus".[3]




```
# loading dataset
data = sns.load_dataset("iris")

# draw lineplot
sns.lineplot(x="sepal_length", y="sepal_width", data=data)

# setting the title using Matplotlib
plt.title('Title using Matplotlib Function')

plt.show()
```

Working with the seaborn dataset

```
import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset("iris")
iris.head()
```

Check the details of initial data

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
iris = sns.load_dataset("iris")
iris.head(3)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa

Working with the seaborn dataset

```
iris = sns.load_dataset("iris")
iris['sepal_width']
```

Check the details of each data type

```
0      3.5
1      3.0
2      3.2
3      3.1
4      3.6
      ...
145    3.0
146    2.5
147    3.0
148    3.4
149    3.0
Name: sepal_width, Length: 150, dtype: float64
```

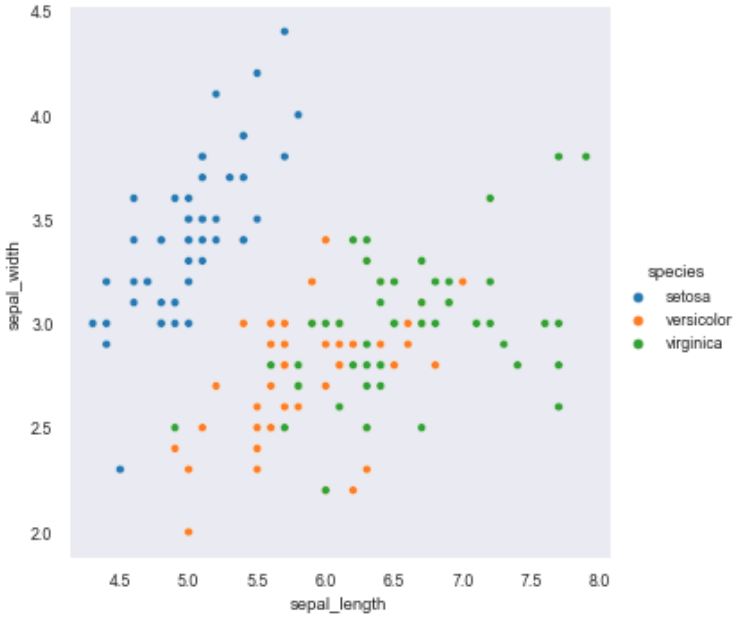
```
iris = sns.load_dataset("iris")
iris['petal_width']
```

```
0      0.2
1      0.2
2      0.2
3      0.2
4      0.2
      ...
145    2.3
146    1.9
147    2.0
148    2.3
149    1.8
Name: petal_width, Length: 150, dtype: float64
```

Working with the seaborn dataset

relplot provides access to several different axes-level functions that show the relationship between two variables with semantic mappings of subsets.
The kind parameter selects the underlying axes-level function to use:

seaborn.**relplot**(*data=None*, *, *x=None*, *y=None*, *hue=None*)
x,y: Variables that specify positions on the x and y axes.
hue*:* Grouping variable that will produce elements with different colors.



Working with the seaborn dataset

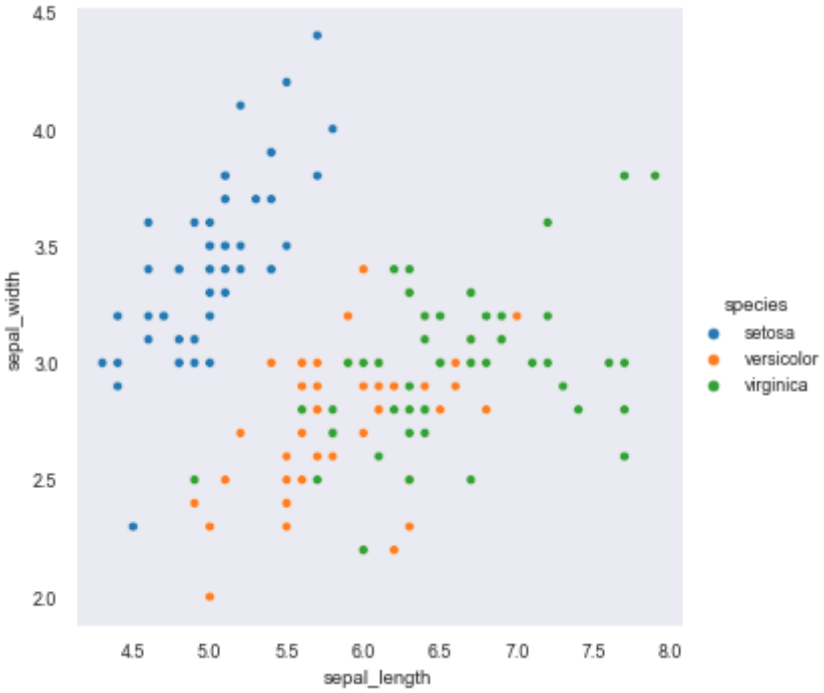
relplot provides access to several different axes-level functions that show the relationship between two variables with semantic mappings of subsets.
The kind parameter selects the underlying axes-level function to use:

```
import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset("iris")

sns.relplot(
    data=iris,
    x="sepal_length", y="sepal_width",
kind='scatter', hue = 'species')
```

line or scatter

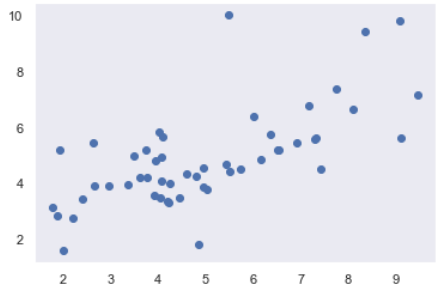


A small exercise for you

```
import pandas
import matplotlib
import scipy
import seaborn as sns
print(sns.get_dataset_names())
from matplotlib import pyplot as plt
import seaborn as sns
df=sns.load_dataset('car_crashes')
print(df.head())
plt.scatter(df.speeding,df.alcohol)
plt.show()
```

```
['anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 'diamonds',
'dowjones', 'exercise', 'flights', 'fmri', 'geyser', 'glue', 'healthexp', 'iris', '
'penguins', 'planets', 'seaice', 'taxis', 'tips', 'titanic']
   total  speeding  alcohol  not_distracted  no_previous  ins_premium  \
0   18.8     7.332    5.640          18.048       15.040       784.55
1   18.1     7.421    4.525          16.290       17.014      1053.48
2   18.6     6.510    5.208          15.624       17.856       899.47
3   22.4     4.032    5.824          21.056       21.280       827.34
4   12.0     4.200    3.360          10.920       10.680       878.41

   ins_losses abbrev
0      145.08     AL
1      133.93     AK
2      110.35     AZ
3      142.39     AR
4      165.63     CA
```

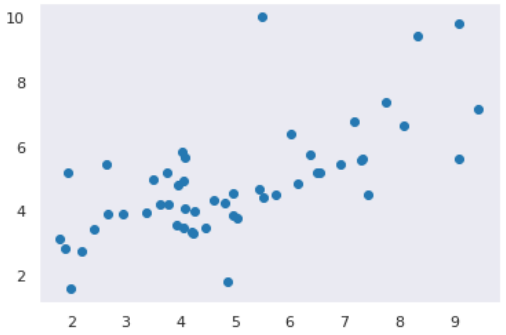


Use replot to show the correlation between speeding and alcohol

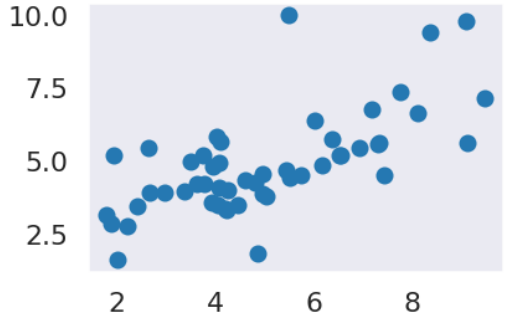
Control individual elements

- Seaborn also allows us to control individual elements of our graphs and thus we can control the scale of these elements or the plot by using the set_context() function. We have four preset templates for contexts, based on relative size, the contexts are named as follows
  - paper
  - notebook
  - talk
  - poster

```
from matplotlib import pyplot as plt
import seaborn as sns
plt.scatter(df.speeding,df.alcohol)
sns.set_style("dark")
sns.set_context("notebook")
plt.show()
```



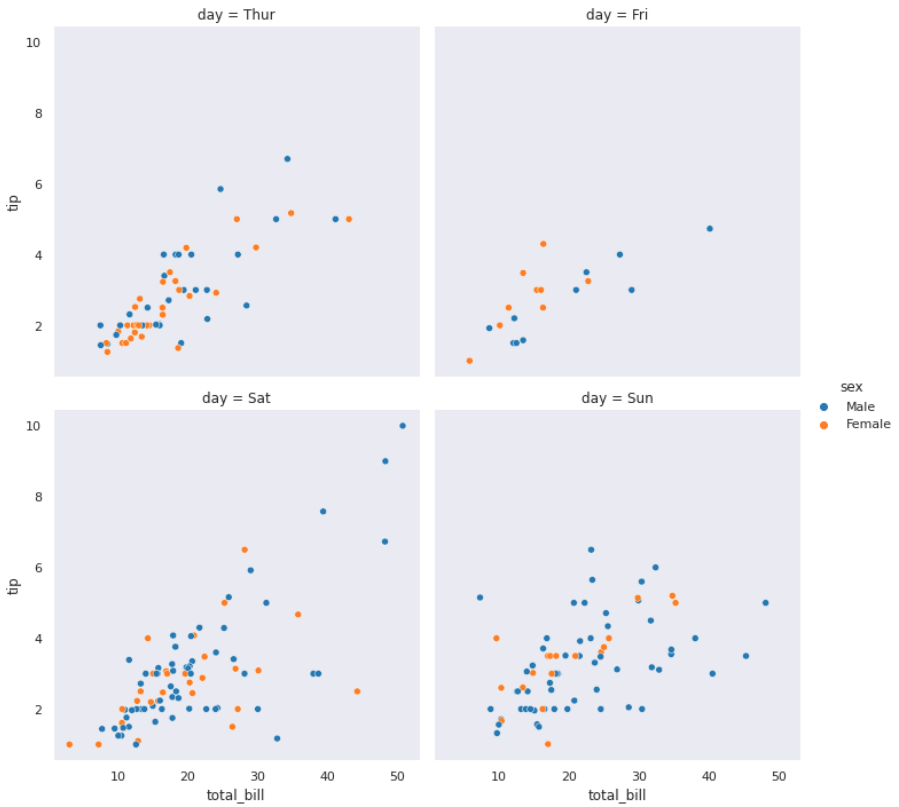

```
from matplotlib import pyplot as plt
import seaborn as sns
plt.scatter(df.speeding,df.alcohol)
sns.set_style("dark")
sns.set_context("poster")
plt.show()
```



Seaborn relplot

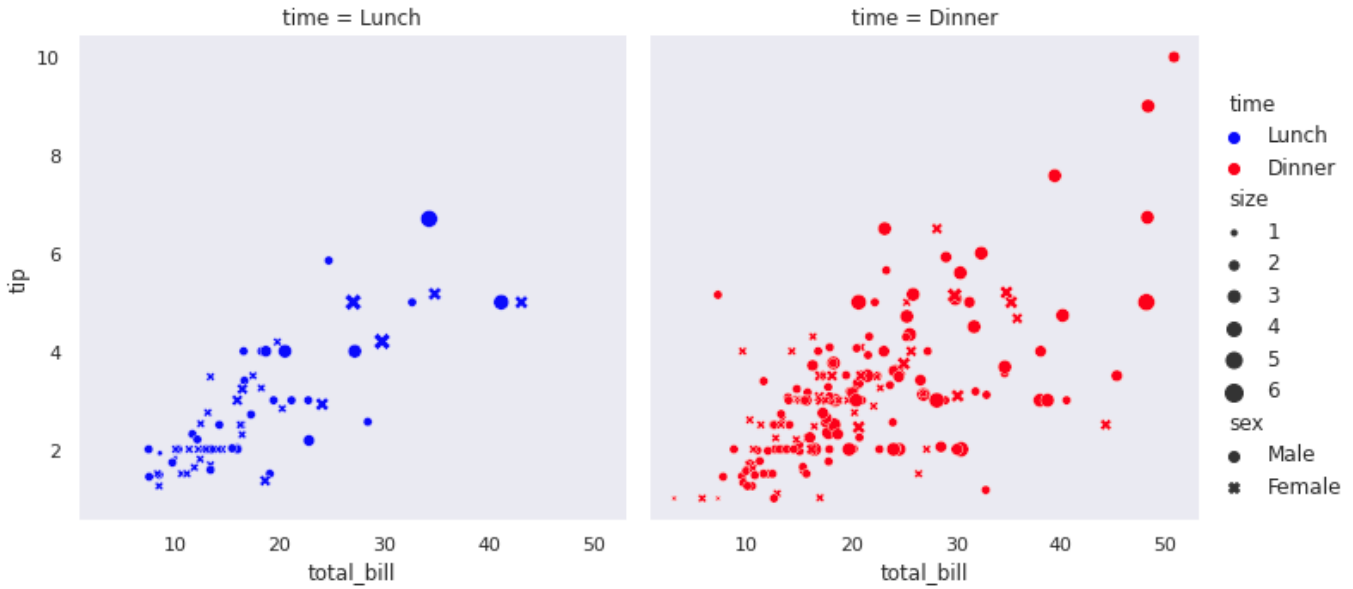
About tip dataset: One waiter recorded information about each tip he received over a period of a few months working in one restaurant. He collected several variables:

```
tips = sns.load_dataset("tips")
sns.relplot(data=tips,
x="total_bill", y="tip",
hue="sex", col="day", col_wrap=2)
```



More power functions of relplot

```
tips = sns.load_dataset("tips")
sns.relplot(data=tips, x="total_bill", y="tip", col="time", hue="time",
size="size", style="sex", palette=["b", "r"], sizes=(10, 100))
```



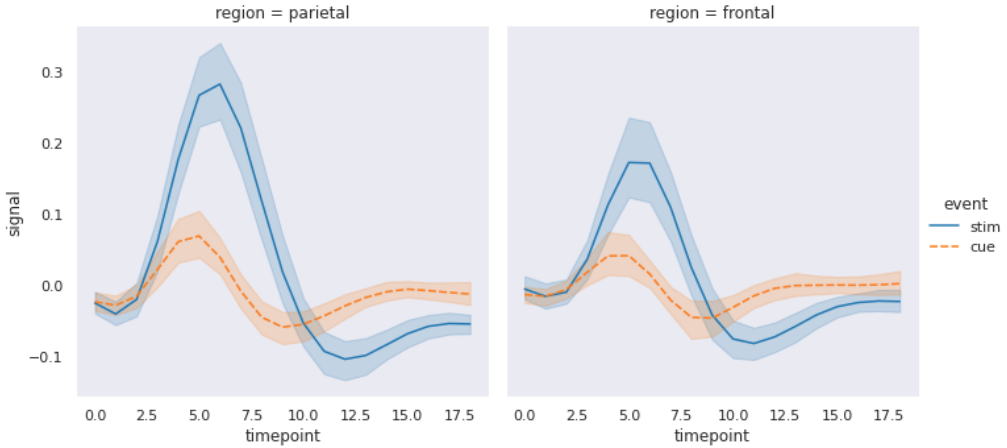
Seaborn relplot

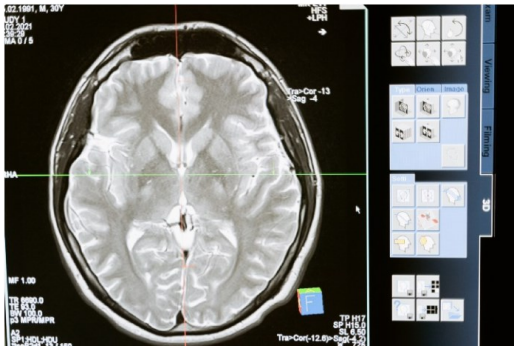
```
fmri = sns.load_dataset("fmri")
fmri.head()
```

fmri is a popular dataset in neuroscience

	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033
3	s11	18	stim	parietal	-0.046134
4	s10	18	stim	parietal	-0.037970

```
sns.relplot(
    data=fmri, x="timepoint", y="signal", col="region",
    hue="event", style="event", kind="line",
)
```



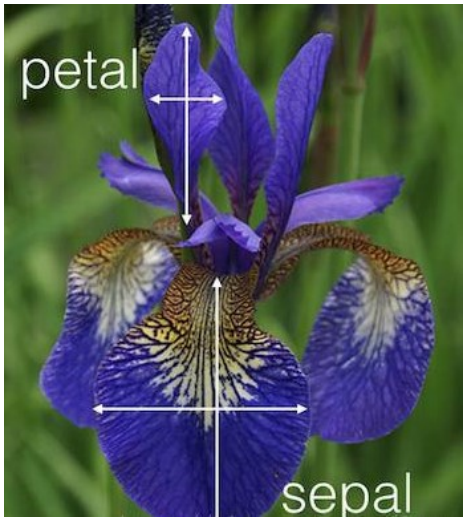


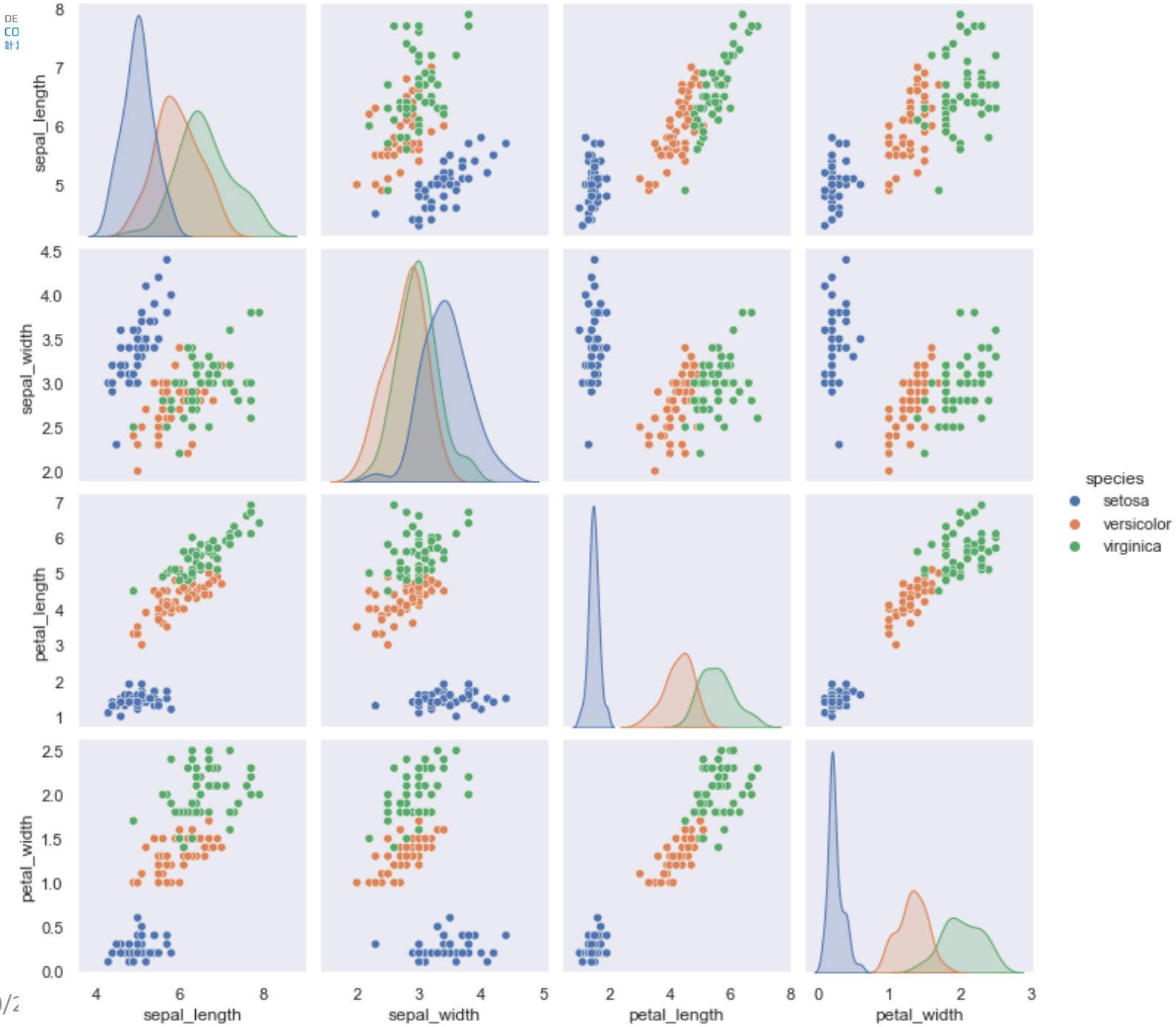
Seaborn pairplot

- Plot pairwise relationships in a dataset.
- How can we find the correlation between petal and sepal?

```
import seaborn as sns
iris = sns.load_dataset("iris")
sns.pairplot(iris, hue = 'species')
```

Different species will
be shown with different color

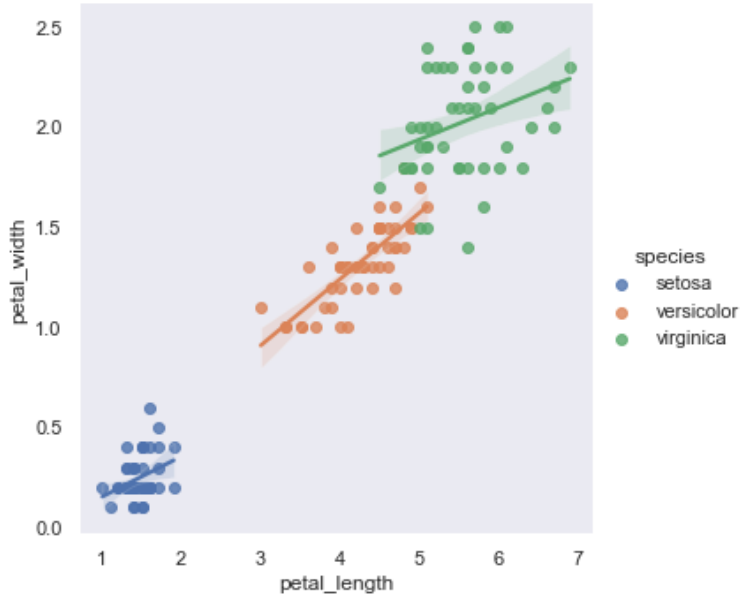



sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica

Seaborn lmplot

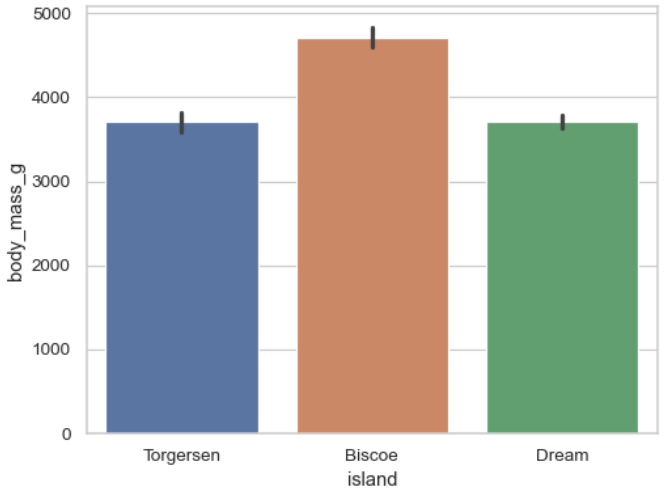
- Show the data regression
- Show the trend across a number of data samples

```
import seaborn as sns
iris = sns.load_dataset("iris")
sns.lmplot(data=iris, x="petal_length", y="petal_width", hue="species");
```



More functions about seaborn

```
df = sns.load_dataset("penguins")
sns.barplot(data=df, x="island",
y="body_mass_g")
```



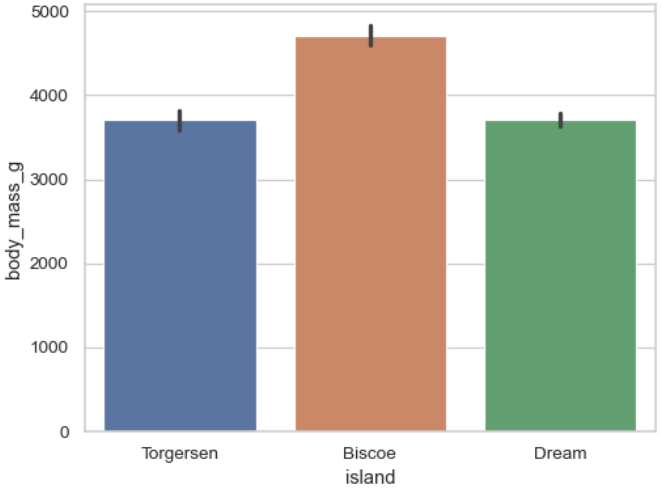

```
df = sns.load_dataset("titanic")
sns.violinplot(x=df["age"])
```

titanic dataset: about the passenges on titantic

	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

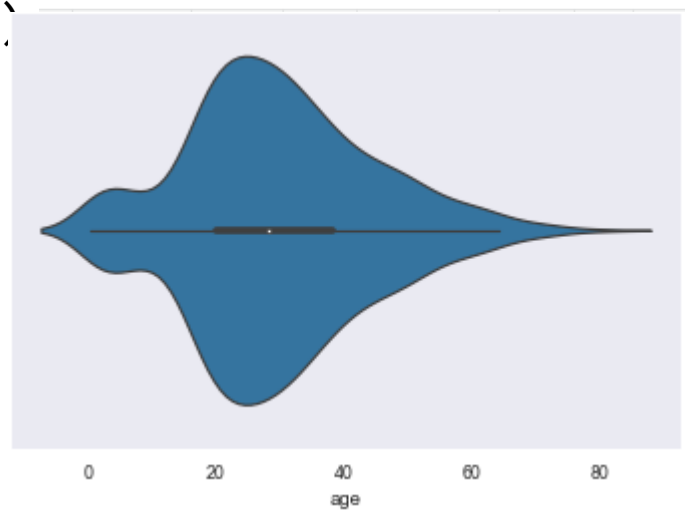
More functions about seaborn

```
df = sns.load_dataset("penguins")
sns.barplot(data=df, x="island",
y="body_mass_g")
```




```
df = sns.load_dataset("titanic")
sns.violinplot(x=df["age"])
```

titanic dataset: about the passenges on titantic



Parch	Ticket	Fare	Cabin	Embarked
0	A/5 21171	7.2500	NaN	S
0	PC 17599	71.2833	C85	C
0	STON/O2. 3101282	7.9250	NaN	S
0	113803	53.1000	C123	S
0	373450	8.0500	NaN	S