# VARIABLES

height = 1.79

## TYPE

### functionality has diferent behavior on diferent types of variables

type(height) output:float

* int
* print(4+5)  
  output: 9
* str represents string
* print('ab'+'cd')  
  output: 'abcd'
* boo represents booleans
* True or False don't need parentheses
* z = True   
  print(type (z))  
  output: bool
* List

# PYTHON LIST

Is a type of variables. List store variables, list may contain variables and strings.

## list type

* fam = [1.79, 1.66, 1.68, 1.65]
* List can contain any type and diferent type in one list fam = ['dad', 1.79, 'mom',1.66,'emma', 1.68,'elsa', 1.65]

### list of list

* list can contain list in itself witch is subsetting list
* fam2 = [  
  ['dad',1.79 ],  
  ['mom',1.66],  
  ['emma', 1.68],  
  ['elsa', 1.65]  
  ]  
  print(type(fam2))  
  output : list

hall = 11.25  
kit = 18.0  
liv = 20.0  
bed = 10.75  
bath = 9.50  
  
# Adapt list areas  
areas = ["hallway", hall, "kitchen", kit, "living room", liv, "bedroom", bed, "bathroom", bath]

"hallway" is string need quotation marks, hall is a variable, no quotation is needed

## Access information of list

### index

for select single element from a list

print(fam[6])

print(fam[-1])

#### slicing

for select multiple elements from a list

空值取到表头或者表尾，数字不包括

### access information of list of list

house = [["hallway", 11.25],  
 ["kitchen", 18.0],  
 ["living room", 20.0],  
 ["bedroom", 10.75],  
 ["bathroom", 9.50]]  
  
# Subset the house list  
house[-1][1]

house = ["cat", "dog", "bird"]  
result = house[-1][1] # 取最后一个字符串 "bird" 的第二个字符  
print(result) # 输出: "i"

areas\_copy = areas[:] = areas\_copy = list(areas)

## manipulate list or list of list

### 1. Modify a Single Element

my\_list = [1, 2, 3, 4, 5]  
my\_list[2] = 10 # Change the element at index 2  
print(my\_list) # Output: [1, 2, 10, 4, 5]  
  
nested\_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
nested\_list[1][2] = 100 # Change the third element of the second inner list  
print(nested\_list) # Output: [[1, 2, 3], [4, 5, 100], [7, 8, 9]]

### 2. Modify Multiple Elements with Slicing

my\_list = [1, 2, 3, 4, 5]  
my\_list[1:4] = [20, 30, 40] # Replace elements at indices 1, 2, and 3  
print(my\_list) # Output: [1, 20, 30, 40, 5]

### 3. Replace a Range with a Different-Sized List

# Replace with a larger list  
my\_list = [1, 2, 3, 4, 5]  
my\_list[1:3] = [20, 30, 40, 50]  
print(my\_list) # Output: [1, 20, 30, 40, 50, 4, 5]  
  
# Replace with a smaller list  
my\_list = [1, 2, 3, 4, 5]  
my\_list[1:4] = [100]  
print(my\_list) # Output: [1, 100, 5]

### 4. Add New Elements to a List

1. my\_list = [1, 2, 3]  
my\_list.append(4) # Add an element at the end  
print(my\_list) # Output: [1, 2, 3, 4]  
  
2. my\_list.extend([5, 6]) # Add multiple elements at the end  
print(my\_list) # Output: [1, 2, 3, 4, 5, 6]  
  
3. new\_list = my\_list + [7] # create a new variable  
print(new\_list) # output:[1, 2, 3, 4, 5, 6,7]

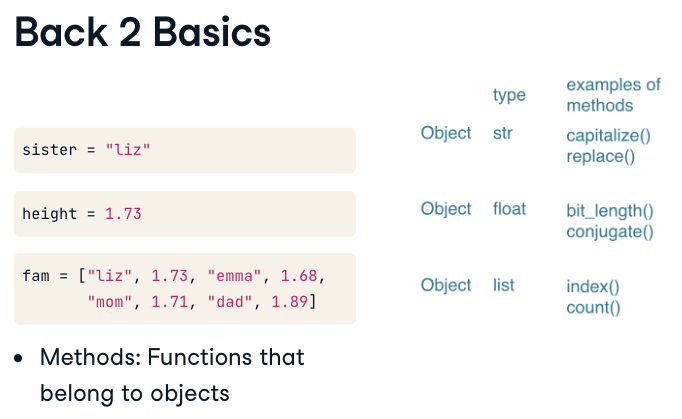
### 6. Remove Elements

# Remove by index  
#1.   
my\_list = [1, 2, 3, 4]  
my\_list.pop(2) # Remove the element at index 2  
print(my\_list) # Output: [1, 2, 4]  
  
# 2.   
my\_list = [1, 2, 3, 4]  
del my\_list[1] # Remove the element at index 1  
print(my\_list) # Output: [1, 3, 4]  
  
areas = ["hallway", 11.25, "kitchen", 18.0,  
 "chill zone", 20.0, "bedroom", 10.75,  
 "bathroom", 10.50, "poolhouse", 24.5,  
 "garage", 15.45]  
  
# Delete the poolhouse items from the list  
del areas[10:12]  
  
# Remove by value  
my\_list = [1, 2, 3, 4]  
my\_list.remove(3) # Remove the first occurrence of 3  
print(my\_list) # Output: [1, 2, 4]

# FUNCTION

type()  
max(int1,int2) 取int1,和int2的最大值  
round( number,decimals) round() to output integer四舍五入  
len()  
sort(iterable, reverse = False)排序功能， false asc， True desc  
str(), int(), bool() , float(),list() switch between data types.

# METHOD



string\_name.capitalize()  
string.replace(original content ,new content )  
string\_name.upper()

list\_name.index() find the index of a particular value  
list\_name.count()计算某个值出现了多少次  
list.append()  
list.pop() pop()内是index值，不是list表里面的值  
list.remove() 移除object\_name第一个 ()里面的值  
list.reverse()reverse the order of the list

## methods change the object

everything = object  
fam.append('me')  
fam.append(1,79)

## upper

.upper() used for individual strings  
The .str accessor is a special bridge that allows you to apply string methods like .upper() to each element in the Series automatically.

Series.str.upper()

my\_string = 'hello'  
print(my\_string.upper()) # Output: 'HELLO'

# Numpy

## 1D lumpy array

### np.array()

np.array() only contain one datatype  
np.array()is a new data type which means it has itself methods or function

Advantage compare to the list: calculations over entire array.两个array间的运算

### selection of data

#### Index and slicing

bmi[2]  
bmi[2:]

#### Boolean

bmi[bmi >23] 里面的bmi>23 输出boolean值，外面的bmi[]只筛选true

and or not

* and: ture and true = true; false and false = false
* or: true or false = true;
* not: not true = false, not false = true
* np.logical\_and( , ), np.logical\_or( , ), np.logical\_not( ) 括号内只有一个条件

if, else, elif  
值输出一个结果，如果两个条件都满足，只返回第一个条件对应的输出内容  
if condition:  
 expression  
elif condition:  
 expression  
else:  
 expression  
expression一定要空两格才能生效， if condition后一定要加：

## 2D Numpy Arrays

### from 1d numpy array to 2d numpy array

import numpy as np  
  
# Create two 1D NumPy arrays  
arr1 = np.array([1, 2, 3])  
arr2 = np.array([4, 5, 6])  
2d\_array = np.array([arr1,arr2])

### shape

shape is an attribute not a method, list of list could be transferred to the 2d dumpy array, each list is one row.

2d np.array has rectangular data structure

print(np\_array1.shape)  
shape means the number of rows and the number of elements/columns in an array, for example :

import numpy as np  
  
# Create a 2D NumPy array  
np\_array1 = np.array([[1, 2, 3],   
 [4, 5, 6],   
 [7, 8, 9],   
 [10, 11, 12]])  
  
# Get the shape  
print(np\_array1.shape)

(4, 3)# 4 rows 3 columns

### subsetting of data 2 methods

# Create a 2D NumPy array  
np\_2d = np.array([[1, 2, 3],   
 [4, 5, 6],   
 [7, 8, 9]])  
  
# Access element in the 0th row, 2nd column  
print(np\_2d[0][2]) # Step-by-step method  
print(np\_2d[0, 2]) # NumPy-style direct access  
print(np\_2d[:,:1])

3  
3  
[1,2]  
[4,5]  
[7,8]

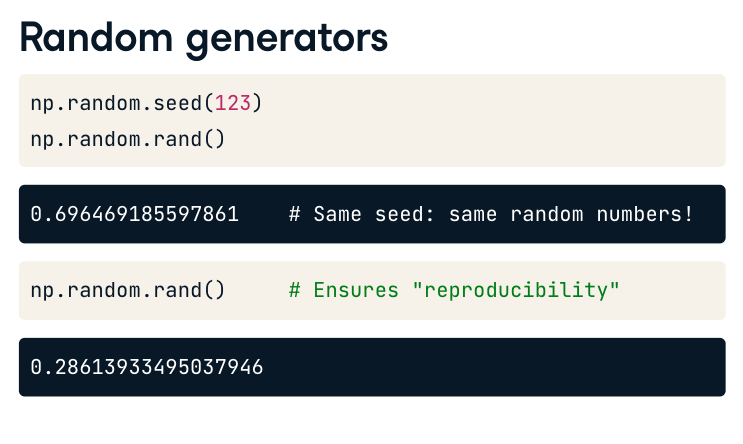
[ , ]不能只有一个数字，要包括行和列。

## Statistics of numpy array

print(np\_baseball+updated)  
# calculation of 2 numpy arrays  
# element-wise calculation

np.mean(variable1[ , ]) get the average number, if the variable 1 is a 2d numpy array  
np.median(variable1[ , ])get the median number  
np.corrcoef(variable1[ :, 0], variable1[:, 1])  
np.std(variable1[ , ])  
np.random.normal() simulate data   
np.column\_stack(variable1, variable2)把横向的list变成纵向的列

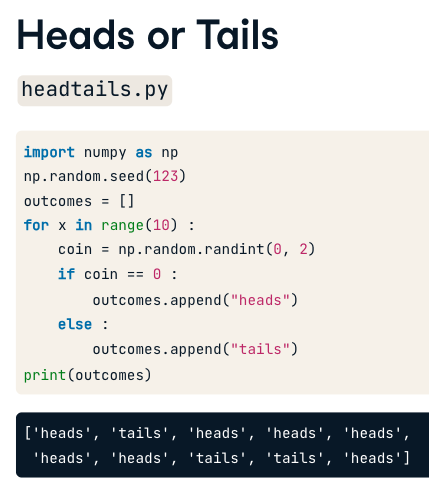
## Simulate data





# Import numpy and set seed  
import numpy as np  
np.random.seed(123)  
# to make sure everytime the output random number is the same  
  
# Use randint() to simulate a dice  
print(np.random.randint(1,7))  
# Use randint() again  
print(np.random.randint(1,7))

import numpy as np  
  
print(np.random.randint(1, 7)) # Different result each time  
print(np.random.randint(1, 7)) # Different result each time

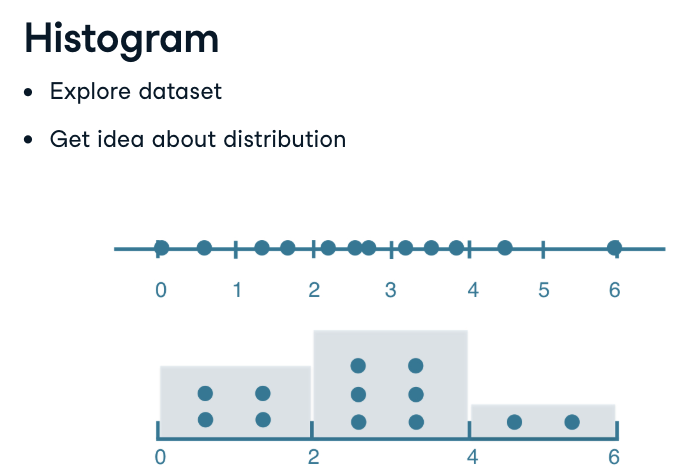


import numpy as np   
np.random.seed(123)  
tails = [0]  
for x in range (10):  
 coin = np.random.randint(0,2)  
 tails.append(tails[x] + coin )

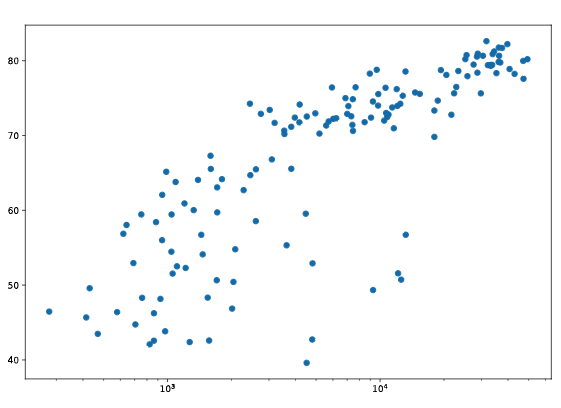
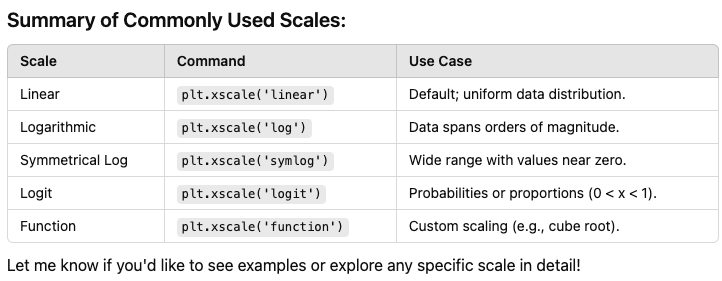
# VISULIZATION - Matplotlib

import matplotlib.pyplot as plt

## Plot type

plt.plot(year, pop) plt.show() 折线图   
plt.scatter(life\_exp, gdp, s= pop) plt.show() 点图，相比折线图没有连线，不会给观众造成误导  
s stands for size, the dot size will be changed adopted to the size of the population  
plt.hist(x, bins=3) plt.show()数据分布图。用于查看数据(x )集中在哪个区间 check the distribution of the values. plt.hist(x) bins = 10 by default  
  
plt.clf() cleans it up again so you can start afresh.

## Customize plot

plt.xscale('log')把x轴的值承指数性分布  
  
plt.title('World Population Projections')  
plt.xlabel('year')  
plt.ylabel('population')  
plt.yticks([0,2,4,6,8,10]，['0','2B','4B','6B','8B','10B'])  
The plt.yticks() function requires== two arguments== to set custom labels:  
Tick positions: A list of numeric positions where the labels should be placed.  
Tick labels: A list of strings corresponding to the tick positions.

plt.text(1550, 71, 'India')  
plt.text(5700, 80, 'China')  
plt.grid(True) 有没有网格线

# DICTIONARIES

A new type of data in Python   
key value pairs are in{}  
keys should be unique and immutable objects(不可变对象)float, string, int.   
list could be updated (reference: manipulate list) which is cannot be a key  
europe = {'spain':'madrid', 'france':'paris', 'germany':'berlin', 'norway':'oslo' }

## Methods

europe.keys()retrieve the keys   
europe['norway'] retrieve the value from a key variable1['key\_name']because here the key is just an element of the dictionary so {} is not suitable here.

## manipulate dictionary

1. Create a Dictionary

* # Creating a dictionary  
  my\_dict = {"name": "Alice", "age": 25, "city": "New York"}

1. Access Dictionary Values

* # Access a value by its key  
  print(my\_dict["name"]) # Output: Alice

1. Add or Update

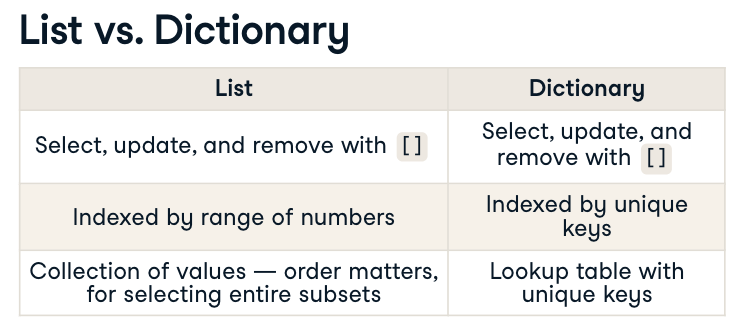
# Add a new pair  
my\_dict["country"] = "USA"  
print(my\_dict) # Output: {'name': 'Alice', 'age': 25, 'city': 'New York', 'country': 'USA'}  
  
# Update an existing value  
my\_dict["age"] = 30  
print(my\_dict) # Output: {'name': 'Alice', 'age': 30, 'city': 'New York', 'country': 'USA'}

1. Delete

# Remove a key-value pair using del  
del(my\_dict["city"]) or del my\_dict["city"]  
print(my\_dict) # Output: {'name': 'Alice', 'age': 30, 'country': 'USA'}  
  
# Remove and return an item using pop()  
age = my\_dict.pop("age")  
print(age) # Output: 30  
# It returns only the value of the removed key, not the entire dictionary.  
print(my\_dict) # Output: {'name': 'Alice', 'country': 'USA'}  
  
# Clear all items  
my\_dict.clear()  
print(my\_dict) # Output: {}

1. check if the key is true

# Definition of dictionary  
europe = {'spain':'madrid', 'france':'paris', 'germany':'berlin', 'norway':'oslo' }  
  
# Add italy to europe  
europe['italy'] = 'rome'  
  
# Print out italy in europe  
print('italy'in europe)

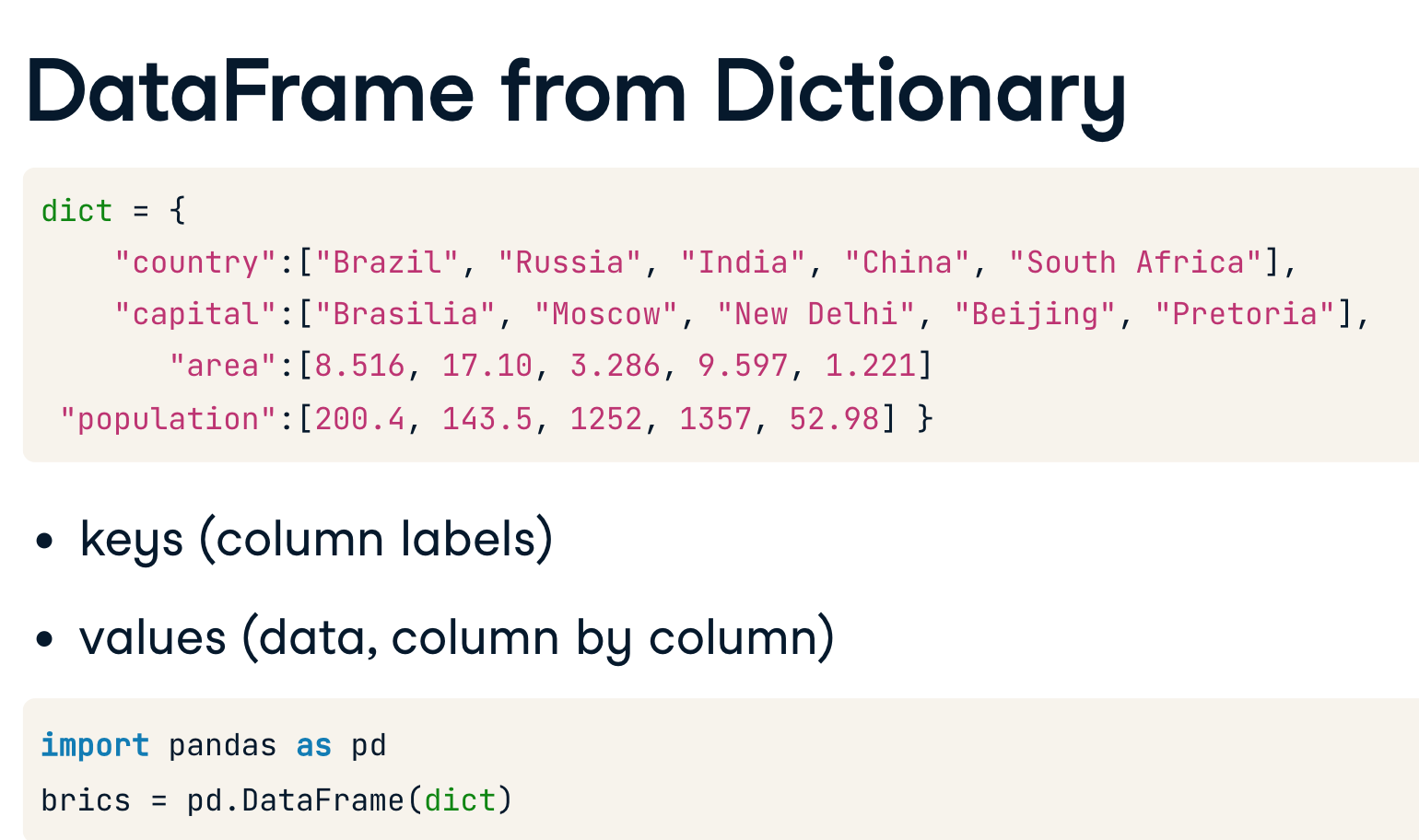


## dictionaries of dictionaries

# Dictionary of dictionaries  
europe = { 'spain': { 'capital':'madrid', 'population':46.77 },  
 'france': { 'capital':'paris', 'population':66.03 },  
 'germany': { 'capital':'berlin', 'population':80.62 },  
 'norway': { 'capital':'oslo', 'population':5.084 } }  
  
# Print out the capital of France  
print(europe['france']['capital'])  
  
# Create sub-dictionary data  
data ={'capital': 'rome', 'population': 59.83}  
  
# Add data to europe under key 'italy'  
europe['italy'] = data  
  
# Print europe  
print(europe)

# PANDAS

Revolution from dictionaries to dataframe  
key in the dict is the labels of the column, **values** can be any type



pandas is much more advanced than 2d numpy array cause there could be variable datatype in pandas.

## Create dataframe

### By rows

# Create a list of dictionaries with new data  
avocados\_list = [  
 {'date': '2019-11-03', 'small\_sold': 10376832, 'large\_sold': 7835071},  
 {'date': '2019-11-10', 'small\_sold': 10717154, 'large\_sold': 8561348},  
]  
  
# Convert list into DataFrame  
avocados\_2019 = pd.DataFrame(avocados\_list)  
  
# Print the new DataFrame  
print(avocados\_2019)

### By columns

# Create a dictionary of lists with new data  
avocados\_dict = {  
 "date": ['2019-11-17', '2019-12-01'],  
 "small\_sold": [10859987, 9291631],  
 "large\_sold": [7674135, 6238096]  
}  
  
# Convert dictionary into DataFrame  
avocados\_2019 = pd.DataFrame(avocados\_dict)  
  
# Print the new DataFrame  
print(avocados\_2019)

## Read csv and save

pd.DataFrame(dict) create a dataframe  
pd.read\_csv('path/to/brics.csv', index\_col=0)  
variable.index更改row lable

new\_dogs.to\_csv('new\_dogs\_with\_bmi.csv')

row\_labels = ['US', 'AUS', 'JPN', 'IN', 'RU', 'MOR', 'EG']  
# Specify row labels of cars  
cars.index = row\_labels

## Info of dataframe

print(dog.head())# display the first rows of the data  
print(dog.info())# display the column name, data type, missing data  
print(dog.shape)# diplay the number of rows and columns  
print(dog.describe())# display the summary statistic data  
# 3 component of dataframe  
print(dog.values)# display values in 2d numpy array  
print(dog.columns)# display the lable of columns  
print(dog.index)# display row number or row name

### index

Transform value to index make the filter much easier

dog\_ind = dog.set\_index('name')# while the index is not settled  
dog\_ind.reset\_index(drop = True) #Use drop=False if you want to keep the original index as a column (e.g., for further analysis or exporting).  
# Use drop=True if you want to completely discard the index and just work with the data.

Double index lable

dog\_ind2 = dog.set\_index(['breed', 'color'])# color是breed子集 nested array  
dog\_ind2.loc[[('Laborador', 'brown'), ('chiwawa','white')]]

## Sort and Selection of data

### Sort

#### sort by values

dog.sort\_values('weight\_kg', ascending = False)  
dog.sort\_values(['weight\_kg', 'height\_com'], ascending = [False,True])

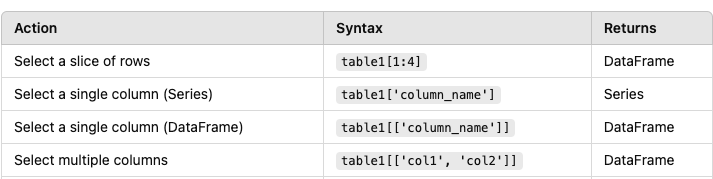
#### sort by index

dog.sort\_index(level = ['breed','color'], ascending = True # or False)

### selection of data

#### square brackets

Limitation: cannot select specific rows and columns simultaneously.   
traits:   
 rows: selection for rows is always dataframe.   
 columns: single brackets is series, double brackets is dataframe

For rows: only support slicing don't support only one index  
For columns: [['column\_name1', 'column\_name2']] to select data in dataframe  


#### loc & iloc

Limitation:   
 1. 选column的时候，必须写rows。可以单独选rows，不选column

Traits:   
 1. single brackets is series, double brackets is dataframe  
 2.loc including the last elements of slicing, iloc not just like slicing of list

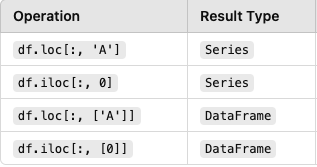
##### how to select double index

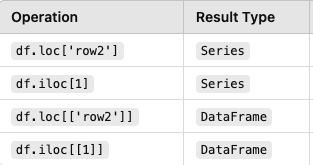
dog.loc[[('Laborador', 'brown'):('chihuahua','white')], 'height\_cm':'weight\_kg']

##### slicing the parcial data for Date

dog=dog.set\_index('birth\_of\_date').sort\_index()  
dog.loc['2012':'2014']

* loc(label-based)
* iloc(index-based)





### Filter pandas DataFrames

only filter the true observations  
sel = cars[cars['drives\_right']]  
car\_maniac = cars[cars['cars\_per\_cap']>500]  
最里面的[]用于筛选列，在两个[]筛选true值  
medium = cars[np.logical\_and(cars['cars\_per\_cap']<=500, cars['cars\_per\_cap']>=300)]

#### filter by strings

# Filter for rows where region is Mountain  
mountain\_reg = homelessness[homelessness['region']=='Mountain']  
  
# See the result  
print(mountain\_reg)

#### filter by numeric

# Filter for rows where family\_members is less than 1000   
# and region is Pacific  
fam\_lt\_1k\_pac = homelessness[homelessness['family\_members']<1000]  
  
# See the result  
print(fam\_lt\_1k\_pac)

# Get the worldwide mean temp by year  
mean\_temp\_by\_year = temp\_by\_country\_city\_vs\_year.mean(axis = 'index')  
print(mean\_temp\_by\_year)  
  
# Filter for the year that had the highest mean temp  
print(mean\_temp\_by\_year[mean\_temp\_by\_year == mean\_temp\_by\_year.max()])  
  
# Get the mean temp by city  
mean\_temp\_by\_city = temp\_by\_country\_city\_vs\_year.mean(axis = 'columns')  
print(mean\_temp\_by\_city)  
  
# Filter for the city that had the lowest mean temp  
print(mean\_temp\_by\_city[mean\_temp\_by\_city == mean\_temp\_by\_city.min()])

#### filter by multiple atrributes

# The Mojave Desert states  
canu = ["California", "Arizona", "Nevada", "Utah"]  
  
# Filter for rows in the Mojave Desert states  
mojave\_homelessness = homelessness[homelessness['state'].isin(canu)]  
  
# See the result  
print(mojave\_homelessness)

#### filter by multiple conditions

# Filter for rows where family\_members is less than 1000   
# and region is Pacific  
fam\_lt\_1k\_pac = homelessness[(homelessness['family\_members']<1000)& (homelessness['region']=='Pacific')]  
  
or # by using np.logical\_and()  
import numpy as np  
fam\_lt\_1k\_pac = homelessness[np.logical\_and(homelessness['region']=='Pacific',homelessness['family\_members']<1000)]  
  
# See the result  
print(fam\_lt\_1k\_pac)

#### filter by query.()

##### by values

similar to where in sql

stocks.query('nike >90 and|or disney is <140')

stocks\_long.query('stock =="disney" or (stock =="nike" and close <90)')

##### by index

# Pivot data so gdp\_per\_capita, where index is date and columns is country  
gdp\_pivot = gdp\_pop.pivot\_table('gdp\_per\_capita', 'date', 'country')  
  
# Select dates equal to or greater than 1991-01-01  
recent\_gdp\_pop = gdp\_pivot.query('date >="1991-01-01"')

## Generate new columns

# Add a year column to temperatures  
temperatures['year'] =temperatures['date'].dt.year

# Add total col as sum of individuals and family\_members  
homelessness['total']=homelessness['individuals']+homelessness['family\_members']  
  
# Add p\_homeless col as proportion of total homeless population to the state population  
homelessness['p\_homeless']=homelessness['total']/homelessness['state\_pop']  
  
# See the result  
print(homelessness)

# Import cars data  
import pandas as pd  
cars = pd.read\_csv('cars.csv', index\_col = 0)  
  
# Use .apply(str.upper)  
cars['COUNTRY']=cars['country'].apply(str.upper)  
print(cars)

## Aggregation data

dogs['height\_cm'].mean(axis = 'column')# calculate statistics across the rows   
dogs['height\_cm'].median()  
dogs['height\_cm'].mode()  
dogs['height\_cm'].min()  
dogs['height\_cm'].max()  
dogs['height\_cm'].sum()  
dogs['height\_cm'].std()  
dogs['height\_cm'].var()  
dogs['height\_cm'].quantile()

### .agg()

DataFrame.groupby(columns).agg(aggregation\_rules)

aggregation\_rules 可以是：  
一个聚合函数（例如：'sum', 'count', 'mean'）。  
一个函数名字符串列表（例如：['mean', 'std']）。  
一个字典，指定每列使用的聚合方式（例如：{'column1': 'mean', 'column2': 'sum'}）。

# Merge licenses and zip\_demo, on zip; and merge the wards on ward  
licenses\_zip\_ward = licenses.merge(zip\_demo, on = 'zip')\  
 .merge(wards, on = 'ward')  
  
# Print the results by alderman and show median income  
print(licenses\_zip\_ward.groupby('alderman').agg({'income':'median'}))

# Import NumPy and create custom IQR function  
import numpy as np  
def iqr(column):  
 return column.quantile(0.75) - column.quantile(0.25)  
  
# Update to print IQR and median of temperature\_c, fuel\_price\_usd\_per\_l, & unemployment  
print(sales[["temperature\_c", "fuel\_price\_usd\_per\_l", "unemployment"]].agg([iqr, np.median]))

### .cumsum(), cummin(), .cummax(), .cumprod()

# Sort sales\_1\_1 by date  
sales\_1\_1 = sales\_1\_1.sort\_values('date')  
  
# Get the cumulative sum of weekly\_sales, add as cum\_weekly\_sales col  
sales\_1\_1['cum\_weekly\_sales'] = sales\_1\_1['weekly\_sales'].cumsum()  
  
# Get the cumulative max of weekly\_sales, add as cum\_max\_sales col  
sales\_1\_1['cum\_max\_sales']=sales\_1\_1['weekly\_sales'].cummax()# 距某日为止，最高值  
  
# See the columns you calculated  
print(sales\_1\_1[["date", "weekly\_sales", "cum\_weekly\_sales", "cum\_max\_sales"]])

output:  
 date weekly\_sales cum\_weekly\_sales cum\_max\_sales  
 0 2010-02-05 24924.50 24924.50 24924.50  
 1 2010-03-05 21827.90 46752.40 24924.50  
 2 2010-04-02 57258.43 104010.83 57258.43  
 3 2010-05-07 17413.94 121424.77 57258.43  
 4 2010-06-04 17558.09 138982.86 57258.43  
 5 2010-07-02 16333.14 155316.00 57258.43  
 6 2010-08-06 17508.41 172824.41 57258.43  
 7 2010-09-03 16241.78 189066.19 57258.43  
 8 2010-10-01 20094.19 209160.38 57258.43  
 9 2010-11-05 34238.88 243399.26 57258.43  
 10 2010-12-03 22517.56 265916.82 57258.43  
 11 2011-01-07 15984.24 281901.06 57258.43

## Distinct

unique\_dog = dog\_visits.drop\_duplicates(subset =['name','breed'])

## Count

# Count the number of stores of each type  
store\_counts = store\_types['type'].value\_counts()  
print(store\_counts)

### Count percentage

# Get the proportion of stores of each type  
store\_props = store\_types['type'].value\_counts(normalize= True)  
print(store\_props)

### Count & sort

# Get the proportion of stores in each department and sort  
dept\_props\_sorted = store\_depts['department'].value\_counts(sort=True, normalize=True) # default sort = True descending order. sort = False, no sorting is performed  
print(dept\_props\_sorted)

## Groupby

### groupby index keys

# Concatenate the tables and add keys  
inv\_jul\_thr\_sep = pd.concat([inv\_jul, inv\_aug, inv\_sep],   
 keys=['7Jul', '8Aug', '9Sep'])  
print(inv\_jul\_thr\_sep.head())   
  
# Group the invoices by the index keys and find avg of the total column  
avg\_inv\_by\_month = inv\_jul\_thr\_sep.groupby(level=0).agg({'total':'mean'})

### groupby columns

dogs.groupby('color')['weight\_kg'].mean()  
dogs.groupby('color')['weight\_kg'].agg([min, max, sum])  
dogs.groupby(['color','breed'])['weight\_kg'].agg([min, max, np.mean, np.median])  
dogs.groupby(['color','breed'])[['weight\_kg','height\_cm']].agg([min, max, sum])

## Pivot table

dogs.pivot\_table(values = 'weight\_kg', index = 'color', columns = 'bread', aggfunc = [np.median, np.mean], fill\_value = 0, margins = True)# default aggfuncn is output mean value, margins = True to output mean values

## .melt()

this method is opposite to pivot table, reframe the table to long format

social\_fin.melt(id\_vars = ['financial', 'company']，# the columns to be maintained  
 value\_var['2018', '2017']，# column变value想要保留的值  
 var\_name = 'year', value\_name = 'dollars'  
 )

## Visulization

### .plot () for bar, line, scatter

price\_diffs.plot(y=['close\_jpm', 'close\_wells', 'close\_bac'])  
plt.show()

3 lines, x axis is the index, python treat 3 y axis as 3 separate lists, so there is 3 lines

avg\_weight\_by\_breed.plot(kind = 'bar', title = 'Mean Weight by Dog Breed', x = 'breed', y = 'mean\_weight', rot = 45, alpha = 0.7)  
plt.legend =()  
# rot = 45 rotate x label 45degrees   
plt.show

### .hist()

# Modify bins to 20  
avocados[avocados["type"] == "conventional"]["avg\_price"].hist(alpha=0.5, bins =20)  
  
# Modify bins to 20  
avocados[avocados["type"] == "organic"]["avg\_price"].hist(alpha=0.5, bins = 20)  
  
# Add a legend  
plt.legend(["conventional", "organic"])  
  
# Show the plot  
plt.show()

#### Customize bins

# Create a histogram of restaurant\_groups and show plot  
restaurant\_groups['group\_size'].hist(bins =[2, 3, 4, 5, 6])  
  
plt.show()

## Missing Values

### Check the missing values

dogs['weight\_kg'].isnull()  
dogs.isna().any()# if there is any nan in each column  
dogs.isna().sum()# count the total number of nan values  
dogs.isna().sum().plot(kind ='bar')  
plt.show()

### Remove missing values

dogs.dropna()

### Replace NAN with 0

dogs.fillna(0)

### Fill with the proceeding values

pd.merge\_ordered(aapl, mcd,   
 on = 'date',  
 suffixes = ('\_aapl', 'mcd'),  
 fill\_method = 'ffill')

## Merge tables horizontally

### Merge on columns

#### Inner Join

##### Merge 1 table

wards\_census=wards.merge(census, on = 'ward', suffixes=('\_ward','\_cen'))  
# the order is from left table to the right table

##### Merge multiple tables

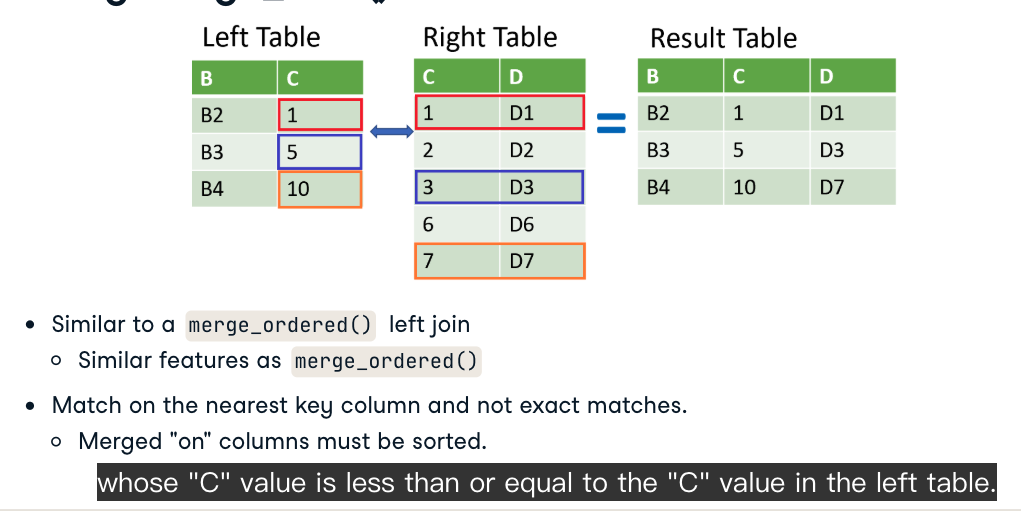
# Merge the ridership, cal, and stations tables  
ridership\_cal\_stations = ridership.merge(cal, on=['year','month','day'])\  
 .merge(stations, on = 'station\_id')

#### outer join

##### left outer join

wards\_census=wards.merge(census, left\_on = 'ward',right\_on = 'ward\_id', suffixes=('\_ward','\_cen'), how = 'left')

##### merge\_asof

similar to left join, but match on the nearest key column. right table value is smaller ou equal to the left table values  


pd.merge\_asof(visa, ibm, on = 'date\_time', suffixes = ('\_visa', '\_ibm'),  
 direction = 'forward'|'backward'|'nearest')

# Merge gdp and recession on date using merge\_asof()  
gdp\_recession = pd.merge\_asof(gdp, recession, on ='date')  
print(gdp\_recession.head())  
  
# Create a list based on the row value of gdp\_recession['econ\_status']  
is\_recession = []  
for s in gdp\_recession['econ\_status']:  
 if s=='recession':  
 is\_recession.append('r')   
 else:  
 is\_recession.append('g')  
print(is\_recession)  
  
# Plot a bar chart of gdp\_recession  
gdp\_recession.plot(kind='bar', y='gdp', x='date', color=is\_recession, rot=90)  
plt.show()

##### right outer join

wards\_census=wards.merge(census, left\_on = 'ward',right\_on = 'ward\_id', suffixes=('\_ward','\_cen'), how = 'right')

##### Full outer join

返回左表和右表中所有行

# Merge iron\_1\_actors to iron\_2\_actors on id with outer join using suffixes  
iron\_1\_and\_2 = iron\_1\_actors.merge(iron\_2\_actors,  
 how = 'outer',  
 on ='id',  
 suffixes=['\_1', '\_2'])  
print(iron\_1\_and\_2.head())   
  
# Create an index that returns true if name\_1 or name\_2 are null  
m = ((iron\_1\_and\_2['name\_1'].isnull()) |   
 (iron\_1\_and\_2['name\_2'].isnull()))  
  
# Print the first few rows of iron\_1\_and\_2  
print(iron\_1\_and\_2[m].head())

##### merge\_ordered()

like outer join, but the values are sorted

# Use merge\_ordered() to merge gdp and sp500, and forward fill missing values  
gdp\_sp500 = pd.merge\_ordered(gdp, sp500, left\_on='year', right\_on='date',   
 how='left', fill\_method='ffill')  
  
# Subset the gdp and returns columns  
gdp\_returns = gdp\_sp500[['gdp', 'returns']]  
  
# Print gdp\_returns correlation  
print (gdp\_returns.corr())

#### self join

self join could be inner join, only return the observation not null, or left table, with nan value

# Merge sequels and financials on index id  
sequels\_fin = sequels.merge(financials, on='id', how='left')  
  
# Self merge with suffixes as inner join with left on sequel and right on id  
orig\_seq = sequels\_fin.merge(sequels\_fin, how='inner', left\_on='sequel',   
 right\_on='id', right\_index=True,  
 suffixes=('\_org','\_seq'))  
  
# Add calculation to subtract revenue\_org from revenue\_seq   
orig\_seq['diff'] = orig\_seq['revenue\_seq'] - orig\_seq['revenue\_org']  
  
# Select the title\_org, title\_seq, and diff   
titles\_diff = orig\_seq[['title\_org','title\_seq','diff']]  
  
# Print the first rows of the sorted titles\_diff  
print(titles\_diff.sort\_values('diff', ascending = False).head())

### Merge on indexes

#### set an index

movies = pd.read\_csv('tmdb\_movies.csv', index\_col = ['id'])

### Filtering joins

#### semi join

只保留左表，左表primary key在右表出现 inner join

print(non\_mus\_tcks.head())  
print(top\_invoices.head())  
print(genres.head())  
  
# 寻找左右两表都有的行  
tracks\_invoices = non\_mus\_tcks.merge(top\_invoices, on = 'tid')  
  
# 用原表与⬆️表对比，找到，只在右表出现过的左表里的行  
top\_tracks = non\_mus\_tcks[non\_mus\_tcks['tid'].isin(tracks\_invoices['tid'])]  
  
# Group the top\_tracks by gid and count the tid rows  
cnt\_by\_gid = top\_tracks.groupby('gid',as\_index=False).agg({'tid':'count'})  
 # as\_index=False，pandas 会将分组键作为普通列保留在结果中 groupby(['gid'])默认gid变成index  
  
# Merge the genres table to cnt\_by\_gid on gid and print  
print(cnt\_by\_gid.merge(genres, on = 'gid'))

#### Anti join

只保留左表，左表primary key没有在右表出现 left join

indicator = True tell the source of observations from which tables

# Merge employees and top\_cust  
empl\_cust = employees.merge(top\_cust, on='srid',   
 how='left', indicator=True)  
  
# Select the srid column where \_merge is left\_only  
srid\_list = empl\_cust.loc[empl\_cust['\_merge'] == 'left\_only', 'srid']  
print(srid\_list)  
# empl\_cust.loc[<condition>, 'srid']  
# Select rows where the <condition> is True (i.e., empl\_cust['\_merge'] == 'left\_only').  
  
# Get employees not working with top customers  
print(employees[employees['srid'].isin(srid\_list)])

## Concatenate vertically

### multi tables with same columns

pd.concat([tracks\_master, tracks\_ride, tracks\_st],   
 ingore\_index = True)

pd.concat([tracks\_master, tracks\_ride, tracks\_st],   
 ingore\_index = True,  
 keys = ['jan', 'Feb', 'Mar'])  
# if you want to set keys, ignore\_index = True is not allowed

### multi tables with variable columns

#### outer join

concat()includes all columns from diferent tables by default

pd.concat([tracks\_master, tracks\_ride, tracks\_st],   
 sort = True ,# sort the columns' name alphabeticaly  
 ingore\_index = True,  
 keys = ['jan', 'Feb', 'Mar'])

#### iner join

sort = True doesn't work for inner join

pd.concat([tracks\_master, tracks\_ride, tracks\_st],   
 join = 'inner',# default is 'outer'  
 ingore\_index = True,  
 keys = ['jan', 'Feb', 'Mar'])

## Verifying integrity

### For merging

.merge(validate = 'one\_to\_one'|'one\_to\_many')

### For concatate

.concat(verify\_integrity = True)# check if there is duplicate index, the default is False, which return with duplicate values

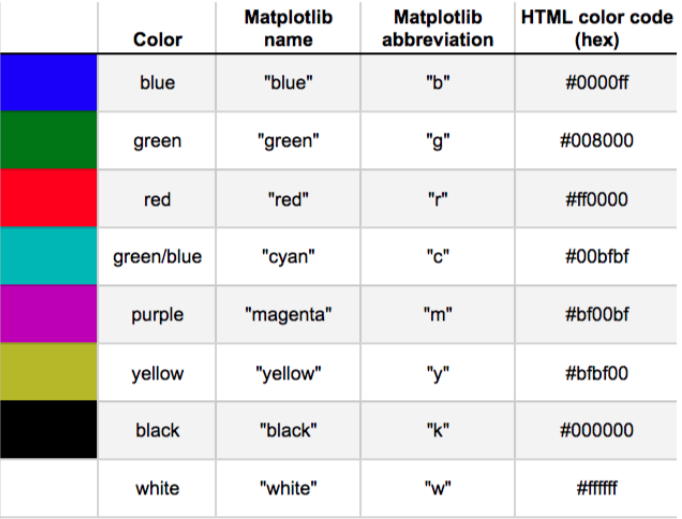
# SEABORN

## Axis subplot

import seaborn as sns  
import matplotlib.pyplot as plt  
import pandas as pd  
df = pd.read\_csv('maculinity.csv')  
sns.countplot( x = 'how\_masculine', data = df)  
sns.scatterplot(x=gdp, y=phones)  
plt.show()

## Adding third variable

import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
hue\_colors = {'yes':'black', 'no': 'red'}  
  
tips = sns.load\_dataset('tips')  
sns.scatterplot(x = 'total\_bill',  
 y = 'tip',  
 hue = 'smoker', # by addinga color to add a new variable  
 paletter = hue\_colors,# dictionary 给变量自定义颜色  
 hue\_order = ['yes', 'no']# modify the legend order, list  
 data = tips)  
plt.show()



## facetGrid plot

### relplot.()

relplot() for scatter or line plot  
relplot() display the result of subset

import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
hue\_colors = {'yes':'black', 'no': 'red'}  
  
tips = sns.load\_dataset('tips')  
sns.relplot(  
 x = 'total\_bill',  
 y = 'tip',  
 kind = 'scatter'  
 col = 'smoker'，  
 col\_wrap= 2， # 一行放几张表  
 col\_order = ['Thur', 'Fri', 'Sat', 'Sun']  
 row = 'time', # col横向，row竖向展示,两个变量互相结合，四种结果  
 paletter = hue\_colors,# dictionary 给变量自定义颜色  
 hue\_order = ['yes', 'no']# modify the legend order, list  
 data = tips)  
plt.show()

### customize scatter plot

style, size, hue, alpha

# Import Matplotlib and Seaborn  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
# Create a scatter plot of acceleration vs. mpg  
sns.relplot(  
 kind = 'scatter',  
 x = 'acceleration',  
 y = 'mpg',  
 data = mpg,  
 style = 'origin',  
 hue = 'origin'  
)  
# Show plot  
plt.show()

### line plot

differenciation with scatter plot, each poin of scatter plot is a unique observation  
line plot display the same thing over time  
kind = 'line' ci = 'sd'

# Make the shaded area show the standard deviation  
sns.relplot(x="model\_year", y="mpg",  
 data=mpg, kind="line", ci = 'sd')# for standard deviation  
  
# Show plot  
plt.show()

# Import Matplotlib and Seaborn  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
# Add markers and make each line have the same style  
sns.relplot(x="model\_year", y="horsepower",   
 data=mpg, kind="line",   
 ci=None, style="origin",   
 hue="origin",  
 markers = True,# point  
 dashes = False)# solid line  
  
# Show plot  
plt.show()

## .catplot()

bar plot shows the confidence intervals by default

# List of categories from lowest to highest  
category\_order = ["<2 hours",   
 "2 to 5 hours",   
 "5 to 10 hours",   
 ">10 hours"]  
  
# Turn off the confidence intervals  
sns.catplot(x="study\_time", y="G3",  
 data=student\_data,  
 kind="bar",  
 order=category\_order,  
 ci = None)  
# Show plot  
plt.show()

### box plot

sym = ' ' omit outliers, whis = 2.0, whis = [5,95]

# Create a box plot with subgroups and omit the outliers  
sns.catplot(  
 x = 'internet',  
 y = 'G3',  
 data = student\_data,  
 kind = 'box',  
 hue = 'location',  
 sym = ''  
)  
  
# Show plot  
plt.show()

### Point plot

join = False   
estimator = median import numpy   
median is useful to deal with dataset where there is lots of outliers

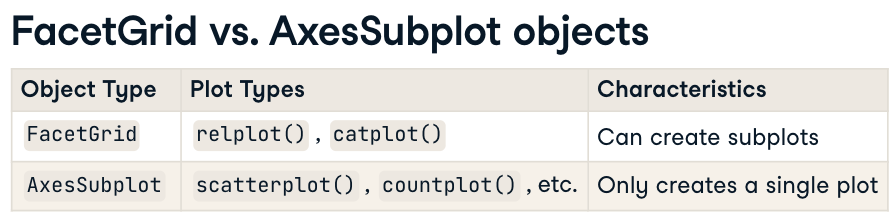
# Import median function from numpy  
from numpy import median  
  
# Plot the median number of absences instead of the mean  
sns.catplot(x="romantic", y="absences",  
 data=student\_data,  
 kind="point",  
 hue="school",  
 ci=None,  
 estimator = median)  
  
# Show plot  
plt.show()

## Customize seaborn plot

### style, palette, context

sns.set\_style(white|dark|whitegrid|darkgrid|ticks)  
sns.set\_palette()# diverging paletter and sequential palette  
sns.set\_context('paper'|'notebook'|'talk'|'poster')# set the size of text from small to big

### title and axis labels



# Create scatter plot  
g = sns.relplot(x="weight",   
 y="horsepower",   
 data=mpg,  
 kind="scatter")  
  
# Add a title "Car Weight vs. Horsepower"  
g.fig.suptitle("Car Weight vs. Horsepower")# For main title, only for fig  
g.set\_title('new\_title', y = 1.03)|  
g.set\_title('This is {col\_name}')# title in each chart, for axes subplot also  
g.set( xlabel = 'New X Label',  
 ylabel = "New Y Label")  
plt.xticks(rotation = 90)# rotate ticks  
  
# Show plot  
plt.show()

# COMPARISON OPERATOR

## Operators: >, <, ==, >=, <=, !=

## Boolean operators and or not

### numpy boolean function

np.logical\_and(), np.logical\_or(), np.logical\_not()

## Conditional statements

if, elif, else  
第一个条件满足，就输出结果，不会进行第二个条件运算

## Pandas comparison

== only works for series==  
pandas works on numpy, numpy boolean operators works for pandas series also

# Import cars data  
import pandas as pd  
cars = pd.read\_csv('cars.csv', index\_col = 0)  
  
# Import numpy, you'll need this  
import numpy as np  
  
# Create medium: observations with cars\_per\_cap between 100 and 500  
medium= cars[np.logical\_and(cars['cars\_per\_cap']>=100,cars['cars\_per\_cap']<=500 )]  
  
# Print medium  
print(medium)

# LOOP

## While loop

while condition:  
 expression

while offset != 0 :  
 print("correcting...")  
 if offset >0 :  
 offset = offset - 1  
 else :   
 offset = offset + 1   
 print(offset)

## for loop

### iterate list

# areas list  
areas = [11.25, 18.0, 20.0, 10.75, 9.50]  
  
# Code the for loop  
for a in areas:  
 print(a)

### iterate 1d numpy array

import numpy as np  
  
# Create a 1D NumPy array  
arr = np.array([10, 20, 30, 40, 50])  
  
# Iterate through the array  
for element in arr:  
 print(f"Element: {element}")

### iterate 2d numpy array

import numpy as np  
  
# Create a 2D NumPy array  
arr\_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
for x in arr\_2d:  
 print(x)  
# output:   
#[1, 2, 3]  
#[4, 5, 6]  
#[7, 8, 9]

### iterate elemens in the list of list

Write a for loop that goes through each sublist of house and prints out the x is y sqm, where x is the name of the room and y is the area of the room.

# house list of lists  
house = [["hallway", 11.25],   
 ["kitchen", 18.0],   
 ["living room", 20.0],   
 ["bedroom", 10.75],   
 ["bathroom", 9.50]]  
   
# Build a for loop from scratch  
for h in house:  
 print('the'+' '+h[0] +' '+'is'+' '+str(h[1])+' '+'sqm')  
 # h[0]is the first element of each sublist, h[1]is the second element of each sublist

### iterate elements of 2d numpy array- nditer(var) function

for var in np.nditer(np\_baseball):  
 print(var)

### iterate list with index

fam = [1.73, 1.68, 1.71, 1.89]

for x, y in enumerate(fam) :# x是index数字，y是value  
 print("person " + str(x) + ": " + str(y))

### iterate numpy array with index

import numpy as np  
  
# Create a 1D NumPy array  
arr = np.array([10, 20, 30, 40, 50])  
for idx, val in enumerate(arr):  
 print(f"Index: {idx}, Value: {val}")

### iterate dictionaries- item()method

# Definition of dictionary  
europe = {'spain':'madrid', 'france':'paris', 'germany':'berlin',  
 'norway':'oslo', 'italy':'rome', 'poland':'warsaw', 'austria':'vienna' }  
   
# Iterate over europe  
for k, v in europe.items():  
 print('the'+' '+'capital'+' '+'of'+' '+k+' '+'is'+' '+v)

### iterate pandas - iterrows method

#### select observations in panda series

# Import cars data  
import pandas as pd  
cars = pd.read\_csv('cars.csv', index\_col = 0)  
  
# Iterate over rows of cars  
for x,y in cars.iterrows():  
 print(x)  
 print(y)

output:  
 US # output x which is lable  
 cars\_per\_cap 809  
 country United States  
 drives\_right True  
 Name: US, dtype: object  
 # output values with column name  
 AUS  
 cars\_per\_cap 731  
 country Australia  
 drives\_right False  
 Name: AUS, dtype: object

#### select variables from panda series

# Import cars data  
import pandas as pd  
cars = pd.read\_csv('cars.csv', index\_col = 0)  
  
# Adapt for loop  
for lab, row in cars.iterrows():  
 print(lab+':'+' '+ str(row['cars\_per\_cap']))

#### create a new column in the panda series

import pandas as pd  
cars = pd.read\_csv('cars.csv', index\_col = 0)  
  
# Code for loop that adds COUNTRY column  
for x in cars.iterrows():  
 cars['COUNTRY']=cars['country'].str.upper()# .str把cars['country']转换成str.接着才能在每一个str上应用upper()  
# Print cars  
print(cars)

# Monte Carlo simulations

## Random generator

np.random.seed(1,2,3)  
np.random.rand()  
np.random.randint(0，2)#只输出整数0 or 1，不包括2  
# 有seed，不管任何时候结果都是和第一次一样的

# NumPy is imported, seed is set  
  
# Starting step  
step = 50  
  
# Roll the dice  
dice = np.random.randint(1,7)  
  
# Finish the control construct  
if dice == 1 or dice== 2:  
 step = step-1  
elif dice >=3 and dice<=5:  
 step = step +1  
else:  
 step = step+np.random.randint(1,7) # 需要重新掷骰子，dice已有是6，这里需要一个新的值，不能用dice直接代替，不然规则就是6号直接走六步  
# Print out dice and step  
print(dice)  
print(step)

## Random path

# NumPy is imported, seed is set  
  
# Initialize random\_walk  
random\_walk = [0]  
  
# Complete the \_\_\_  
for x in range(100) :  
 # Set step: last element in random\_walk  
 step = random\_walk[-1]  
  
 # Roll the dice  
 dice = np.random.randint(1,7)  
  
 # Determine next step  
 if dice == 1 or dice == 2:  
 step = max(0,step -1)  
 elif dice >=3 and dice <=5:  
 step = step +1  
 else:  
 step = step +np.random.randint(1,7)  
 # append next\_step to random\_walk  
 random\_walk.append(step)  
   
# Print random\_walk  
print(random\_walk)  
  
# Import matplotlib.pyplot as plt  
import matplotlib.pyplot as plt  
  
# Plot random\_walk  
plt.plot(random\_walk)  
# Show the plot  
plt.show()

## Distribution

After 1000 times of random walk, we can see the distribution of the last value and do some statistic works.

# numpy and matplotlib imported, seed set  
  
# Simulate random walk 500 times  
all\_walks = []  
for i in range(500) :  
 random\_walk = [0]  
 for x in range(100) :  
 step = random\_walk[-1]  
 dice = np.random.randint(1,7)  
 if dice <= 2:  
 step = max(0, step - 1)  
 elif dice <= 5:  
 step = step + 1  
 else:  
 step = step + np.random.randint(1,7)  
 if np.random.rand() <= 0.001 :  
 step = 0  
 random\_walk.append(step)  
 all\_walks.append(random\_walk)  
  
# Create and plot np\_aw\_t  
np\_aw\_t = np.transpose(np.array(all\_walks))  
  
# Select last row from np\_aw\_t: ends  
ends = np\_aw\_t[-1,:]  
  
# Plot histogram of ends, display plot  
plt.hist(ends)  
plt.show()  
# the ratio of end with steps more than 60  
true\_value = ends[ends>=60]  
print(true\_value.size/500)

# Data Analysis

## Exploratory of data

print(books.head(5))  
print(books.info())   
print(books.dtypes)  
print(books.describe())  
print(books['category'].value\_counts()) # types of categorical values  
sns.histplot( x = '2021', data = books, binswidth = 1)# numerical data

## Data validation

### update datatypes

books['year']= books['years'].astype(int)

Types of data: int, float, str, dict, list, bool

### validate categorical data

books['genre'].isin(['Fiction', 'Not Fiction'])  
# isin( series or data frame)  
# return bool data  
~books['genre'].isin(['Fiction', 'Not Fiction'])  
# reverse the results, if the value is in the series, the result is False  
books[books['genre'].isin(['Fiction', 'Not Fiction'])]  
# return with the dataframe instead of bool

### Validate numerical data

books.select\_dtypes('number')  
books['year'].min()  
books['year'].max()  
sns.boxplot( x = 'year', y = 'genre', data = books)

## Data Summarization

books.groupby('genre').mean()  
books.groupby('genre').min()  
books.groupby('genre').max()  
books.groupby('genre').count()  
books.groupby('genre').std()  
books.groupby('genre').var()  
books.groupby('genre').median()

### .agg()

books.agg(['mean', 'std'])# only applys to numerical columns  
books.agg({'rating':['mean', 'std'], 'year': ['median']})  
  
books.groupby('genre').agg(  
mean\_rating = ('rating', 'mean'),  
std\_rating = ('rating', 'std'))

### Visualize categorical summaries

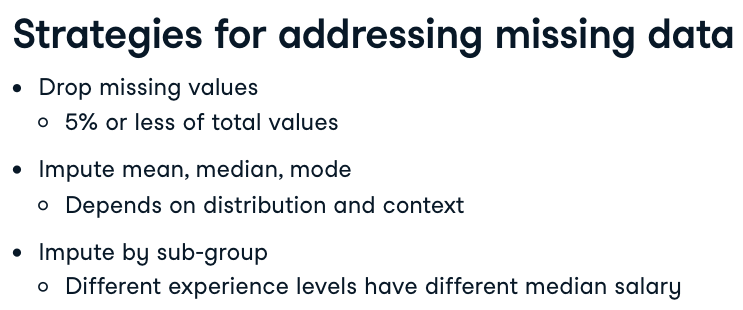
sns.barplot(data = books, x = 'genre', y = 'rating')  
# give the mean value of the quantatative variable auto  
plt.show()

## Addressing missing data

### Checking the missing values

print(salaries.isna().sum())

### Strategies for addressing missing data



#### drop

threshold = len(salaries)\*0.05 # calculate the maximum number for missing number in each columns  
cols\_to\_drop = salaries.columns[salaries.isna().sum()<= threshold]  
salaries.dropna(subset = cols\_to\_drop, inplace = True)# inplace means to update the dataframe

#### Imputing a summary statistic

cols\_with\_missing\_values = salaries.columns[salaries.isna().sum()>0]  
for col in cols\_with\_missing\_values[:-1]:  
 # using silicing to exclue the last col  
 salaries[col].fillna(salaries[col].mode()[0])  
 # there could be multiple mode

##### Imputing by sub-group

salaries.dict = salaries.groupby('experience')['Salary\_USD'].median().to\_dict()  
salaries['Salary\_USD'] = salaries['Salary\_USD'].fillna(salaries['experience'].map(salaries.dict))

## Analyze categorical data

print(salaries.select\_dtypes('object').head())  
# check the frequency of the data   
print(salaries['designation'].value\_counts())  
# count the number of unique job titles  
print(salaries['designation'].nunique())

### Convert categorical data

Find the data with specific strings

pandas.series.str.contains()  
# To search a column for a specific string or multiple string   
salaries['designation'].str.contains('scientist')  
salaries['designation'].str.contains('scientist|AI')# no space  
# to search the strings with the iniciation of data  
salaries['designation'].str.contains('^data')

1. Create job titles that we want to find

job\_categories=['Data Scientice', 'Data Analyst']

1. Create variable for filter the string

data\_science = 'Data Scientist|NLP'  
data\_analyst = 'Analyst|Analytics'

1. Create a list for str.contain()

conditions = [  
(salaries['Designation'.str.contains(data\_science)]),  
(salaries['Designation'.str.contains(data\_analyst)])  
]

1. Create new job\_category column

salaries['job\_category'] = np.select (conditions, job\_categories, default = 'Other')  
# 只要condition为 True，就会自动选择对应索引的 choice 值。这是 np.select() 的核心逻辑之一：条件和选择值的索引是一一对应的。  
# If neither condition is satisfied, the default value 'Other' will be assigned.

# Create conditions for values in flight\_categories to be created  
conditions = [  
 (planes["Duration"].str.contains(short\_flights)),  
 (planes["Duration"].str.contains(medium\_flights)),  
 (planes["Duration"].str.contains(long\_flights))  
]  
  
# Apply the conditions list to the flight\_categories  
planes["Duration\_Category"] = np.select(conditions,   
 flight\_categories,  
 default="Extreme duration")  
  
# Plot the counts of each category  
sns.countplot(data=planes, x="Duration\_Category")  
plt.show()

iterate dataframe

# Filter the DataFrame for object columns  
non\_numeric = planes.select\_dtypes("object")  
  
# Loop through columns  
for x in non\_numeric.columns:  
   
 # Print the number of unique values  
 print(f"Number of unique values in {x} column: ", non\_numeric[x].nunique())  
# The f before the string allows you to embed variables directly inside {}  
# [x] accesses the column named x in the DataFrame

## Analyze numerical data

### Convert strings to numbers

1. remove comma pd.series.str.replace(',', '')

salaries['salary\_in\_rupee'] = salaries['salary\_in\_rupee'].str.replace(',','')

1. convert the column to float

salaries['salary\_in\_rupee'] = salaries['salary\_in\_rupee'].astype(float)

1. create a new column with the updated currency

salaries['salary\_in\_usd']= salaries['salary\_in\_rupee']\*0.012

# Preview the column  
print(planes["Duration"].head())  
  
# Remove the string character  
planes["Duration"] = planes["Duration"].str.replace("h", "")  
  
# Convert to float data type  
planes["Duration"] = planes["Duration"].astype(float)  
  
# Plot a histogram  
sns.histplot(data = planes, x = 'Duration')  
plt.show()

### Adding summary statistics

salaries['std\_dev'] = salaries.groupby('experience')['Salary\_USD'].transform(lambda x:x.std())  
# x stands for ['Salary\_USD']  
# transform method is a row\_wise application, just like windows function in sql

# Price standard deviation by Airline  
planes["airline\_price\_st\_dev"] = planes.groupby("Airline")["Price"].transform(lambda x: x.std())  
print(planes[["Airline", "airline\_price\_st\_dev"]].value\_counts())  
  
# Mean Price by Destination  
planes["price\_destination\_mean"] = planes.groupby("Destination")["Price"].\_\_\_\_(\_\_\_\_ \_\_\_\_: \_\_\_\_.\_\_\_\_())  
print(planes[["Destination","price\_destination\_mean"]].value\_counts())  
  
# Mean Price by Destination  
planes["price\_destination\_mean"] = planes.groupby("Destination")["Price"].transform(lambda x: x.mean())  
print(planes[["Destination","price\_destination\_mean"]].value\_counts())

## Outliers

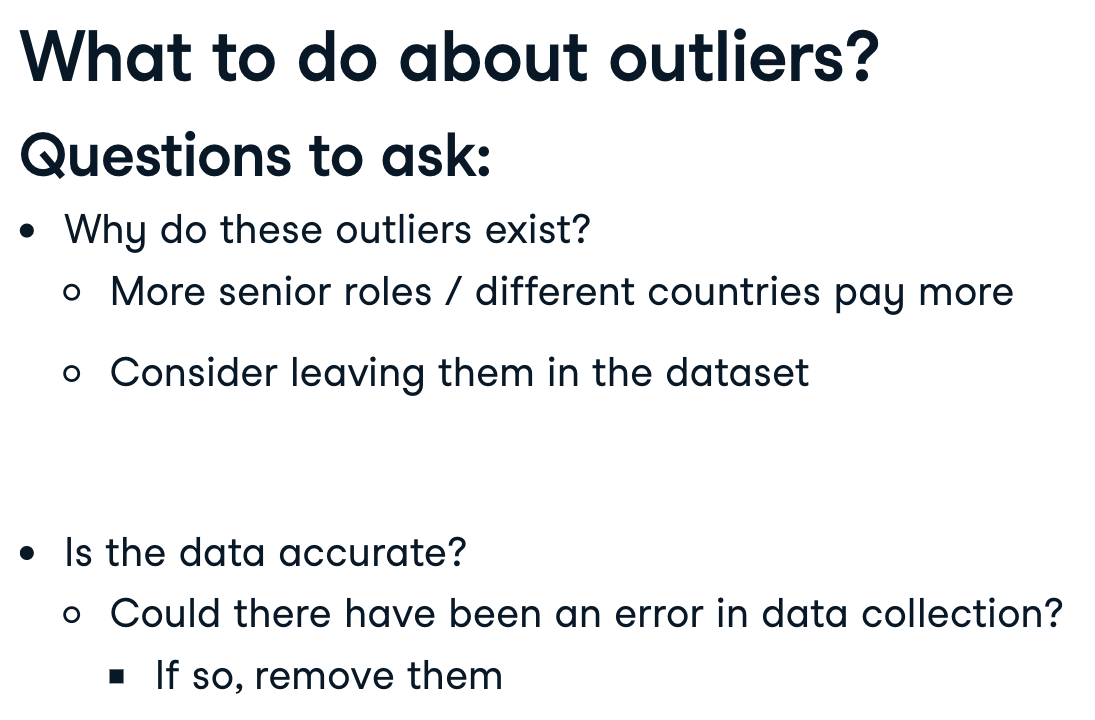
Values > Q3 + 1.5IQR  
Values < Q1 - 1.6 IQR  
Methods to get iqr

1. scipy.stats iqr()

from scipy.stats import iqr  
iqr = iqr(mysleep['total\_sleep'])

1. quantile()

iqr = salaries['salary\_usd'].quantile(0.75)-  
salaries['salary\_usd'].quantile(0.25)



# Find the 75th and 25th percentiles  
price\_seventy\_fifth = planes["Price"].quantile(0.75)  
price\_twenty\_fifth = planes["Price"].quantile(0.25)  
  
# Calculate iqr  
prices\_iqr = price\_seventy\_fifth - price\_twenty\_fifth  
  
# Calculate the thresholds  
upper = price\_seventy\_fifth + (1.5 \* prices\_iqr)  
lower = price\_twenty\_fifth - (1.5 \* prices\_iqr)  
  
# Subset the data  
planes = planes[(planes["Price"] > lower) & (planes["Price"] < upper)]  
  
print(planes["Price"].describe())

## Relationships in data

### Datetime Data

#### Convert to timedata

# Convert when import data  
divorce = pd.read\_csv('divorce.csv', parse\_dates = ['marriage\_date'])  
# convert after import  
divorce['marriage\_date'] = pd.to\_datetime(divorce['marriage\_date'])  
# 组装日期  
divorce['marriage\_date'] = pd.to\_datetime(divorce[  
 'month', 'day', 'year'# 顺序不可更改  
])

#### Extract parts of date

divorce['marriage\_month'] = divorce['marriage\_date'].dt.month  
# dt.day, dt.year, dt.weekday, d t.hour

### Correlation with numerical data

Describe the direction and strength of relationship between two variables  
can help us predict variables to predict future outcomes  
minus results mean one variable increase, the other decrease  
positive results mean one increase, the other increase too  
It is to measure the linear relationship

value close to 0 means weak relationship, value close to 1 ou -1 means strong relationship

divorce.corr()

Visulize the correlation

sns.heatmap(divorce.corr(), annot = True)  
# annot lable the corr in each cell

Pairplots

sns.pairplot( data = divorce, vars = ['income\_man', 'income\_woman','marriage\_duration'])  
plt.show()

### Correlation with categorical data

sns.histplot( data = divorce, x = 'marriage\_duration', hue = 'education\_man', binwidth = 1)  
plt.show()

sns.kdeplot( data = divorce, x = 'marriage\_duration', hue = 'education\_man', cut = 0, cumulative = True)  
# cut 设定x轴的开始和结束值  
plt.show()

## Action

### Class frequency

people in each class may have diferent preference

pd.crosstab(planes['source'], planes['destination'], values = planes['price'], aggfunc = 'median')

### Assign a new category

labels = ['economy', 'premium economy', 'business class', 'first class']  
bins = [0, twenty\_fifth, median, seventy\_fifth, maximum]  
planes['price\_category'] = pd.cut(planes['price'], labels = labels, bins = bins

# Create salary labels  
salary\_labels = ["entry", "mid", "senior", "exec"]  
  
# Create the salary ranges list  
salary\_ranges = [0, twenty\_fifth, salaries\_median, seventy\_fifth, salaries["Salary\_USD"].max()]  
  
# Create salary\_level  
salaries["salary\_level"] = pd.cut(salaries["Salary\_USD"],  
 bins=salary\_ranges,  
 labels=salary\_labels)  
  
# Plot the count of salary levels at companies of different sizes  
sns.countplot(data=salaries, x="Company\_Size", hue="salary\_level")  
plt.show()

### Generate hipoheses

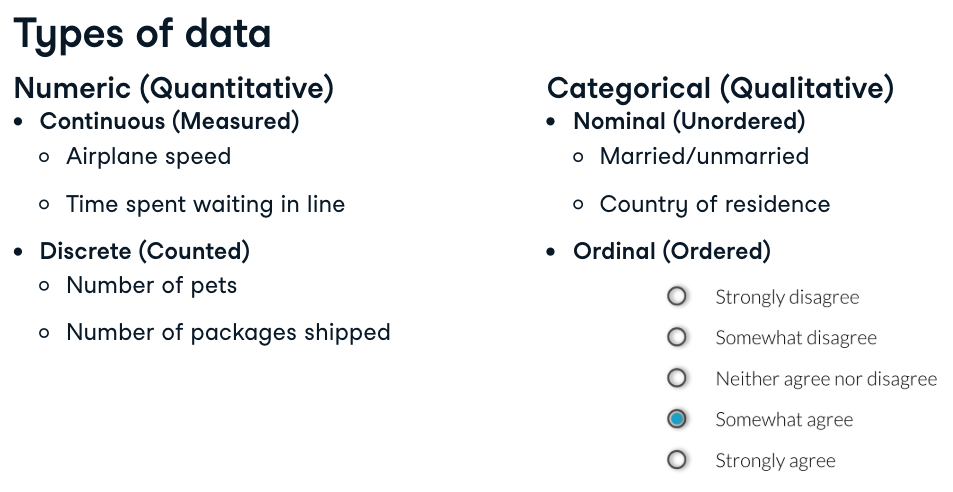
1. generate hypothesis or question
2. decise on what statistical test to use

# Statistic

## types of statistics

Descriptive and inferential statistics  
Descriptive: 已有数据的汇总  
Inferential：通过当前数据，推测出更大群体的数据

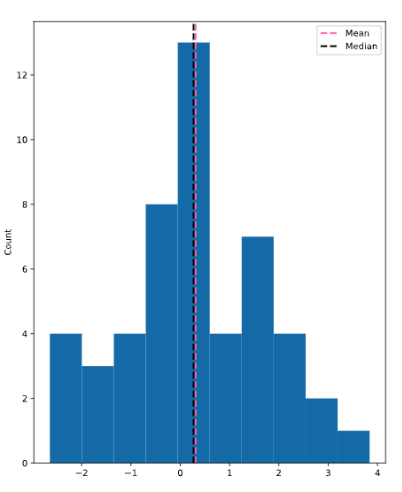
## types of data

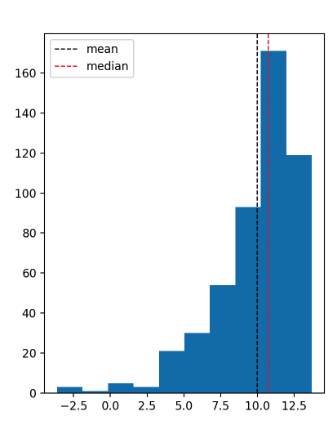


## Summary statistics

### Measures of center

mean:   
 np.mean(msleep['sleep\_total'])  
median:   
 np.median(msleep['sleep\_total'])  
mode:   
 most frequent value in the data   
 statistics.mode(msleep['sleep\_total'])   
 for categorical data  
Mean is much more sensitve to the outliers than median  
median是中间数，不论异常值偏离多大，median都是中间的数字；mean需要所有数相加取平均值，异常值的影响大。  
mean适合分析对称性数据，歪斜数据适合使用median，两张图通过histogram展示center位置，及mean， median所处的位置





### Measures of spread

#### Variance

1. subtract the point from mean value

dists = mysleep['sleep\_total']-np.mean(mysleep['sleep\_total'])# which is a series

1. square each distance

sq\_dists = dists \*\*2

1. sum squared distance

sum\_sq\_dists = np.sum(sq\_dists)

1. divid. by number of data points-1

variance = sum\_sq\_dists/(83-1)

np.var

np.var(mysleep['sleep\_total'],ddof =1)  
# no ddof =1 means you use full population instead of a sample  
# Delta Degrees of Freedom

#### standard deviation

是variance的开平方

np.sqrt(np.var(mysleep['sleep\_total'],ddof =1))  
# sqrt means square root开平方

or

np.std(mysleep['sleep\_total'],ddof =1)

#### Mean absolute deviation

it uses absolute values to measure dispersion离散, making it more robust to outliers

1. find the mean
2. Calculate Absolute Deviations 绝对值
3. Find the Mean of Absolute Deviations

dists = msleep['sleep\_total']-mean(msleep['msleep\_total'])  
np.mean(np.abs(dists))  
# 1. mean(msleep$msleep\_total) find the mean  
# 2. np.abs(msleep['sleep\_total']-mean(msleep$msleep\_total)) absolute deviations  
# 3. np.mean() Mean of Absolute Deviations

## Quantile

np.quantile(msleep['sleep\_total'],0.5)

0.5 quantile = median

### Quartiles

np.quantile(msleep['sleep\_total'], [0, 0.25, 0.5, 0.75, 1])

np.linspace(start, stop, num)

np.quantile(msleep['sleep\_total'], np.linspace(0,1,5))# 把0-1分成几个数字，5-1

### interquartile range IQR

25% - 75% , height of the box

np.quantile(msleep['sleep\_total'], 0.75)-np.quantile(msleep['sleep\_total'], 0.25)

from scipy.stats import iqr  
iqr(msleep['sleep\_total'])

### outliers

outliers > Q1 - 1.5*IQR or Q3 +1.5*IQR

msleep['sleep\_total'].describe()

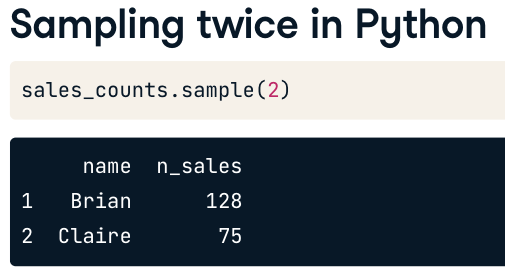
在 Python 中，逻辑运算符 | 的优先级 **低于比较运算符**（如 < 和 >）

lower = q1 - 1.5 \* iqr  
upper = q3 + 1.5 \* iqr  
  
# Subset emissions\_by\_country to find outliers  
outliers = emissions\_by\_country[(emissions\_by\_country<lower) | (emissions\_by\_country> upper)]  
print(outliers)

## Probability

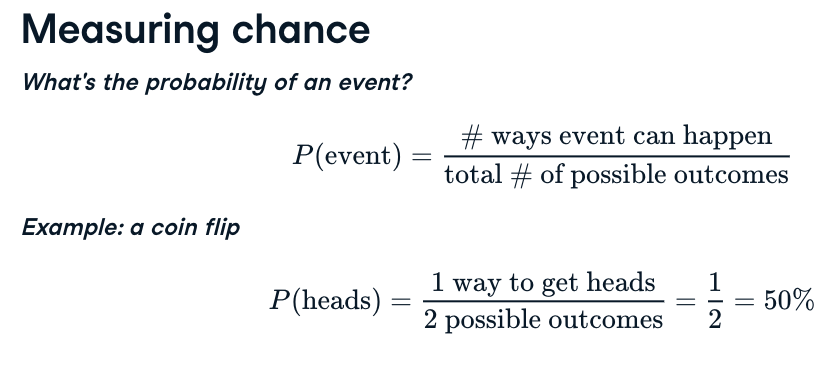
### Dependent chances

np.random.seed(10)# could be a random number   
sales\_count.sample(2)  
# 数字代表选了几次的结果，选了两次第一次是brian第二次是claire



### Independent chances

np.random.seed(10)# could be a random number   
sales\_count.sample(5, replace = True)  
# 5代表选了5次的结果  
# replace means已选的还能继续被选



## Discrete distributions

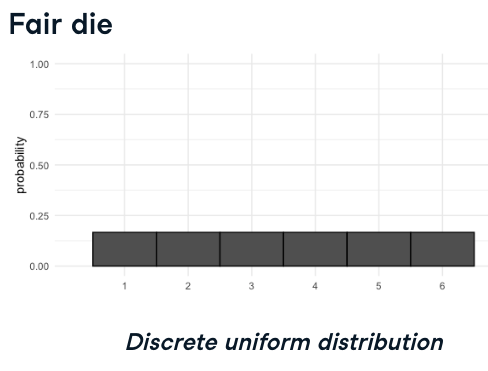
Expected value = mean of probability distribution = sum(discrete \* prob)

### Uniform distribution

任何事件发生的概率都是一样的，y轴值统一

#### discrete uniform distribution

时间可列举

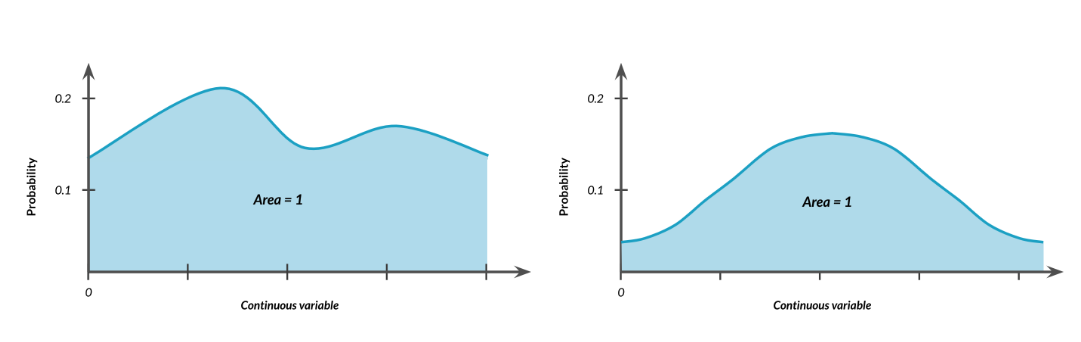


#### Continuous distribution

from scipy.stats import uniform  
# The uniform.cdf refers to the  
uinform.cdf(7,0,12) value <=7# value, start of the spread, end of the spread  
1-uniform.cdf(7,0,12)value >=7  
uinform.cdf(7,0,12) value <=7 - uniform.cdf(4,0,12)<=4 value<=7

from scipy.stats import uniform  
uniform.rvs(0,5,size =10)# minimum value, max value, number of values

Continuous distribution could have other format, instead of uniform:



### Binomial distribution

Probability distribution of the number of success in a sequence of independent trials

Expected value = n \* p number of trials, probability

根据True概率，制作随机数

Binary outcomes   
0 False 1 True

from scipy.stats import binom  
binom.rvs(1,0.5,size = 8)# number of coins, probability of True, times of try, every time is a index  
binom.rvs(8,0.5,size = 1) #outcome array([5])total number of heads\  
# To calculate the possibility of getting less than or equal to 7 heads  
binom.cdf(7,10,0.5)  
# To calculate the possibility of getting 7 heads  
binom.pmf(7,10,0.5)# number of heads,num trials, prob

binom.pmf(k, n, p) : Calculates the probability of exactly k successes.  
binom.cdf(k, n, p): Calculates the probability of up to k successes

### normal distribution

Describe by mean and std, 数据以平均数为中心对称分布

from scipy.stats import norm  
# To get the percent of women shorter than 154  
norm.cdf(154,161, 7)# value, mean, std, get the prob  
# To get the percent of women taller than 154  
1- norm.cdf(154,11,7)  
# To get what height 90% women are shorter than   
norm.ppf(0.9, 160, 7)# prob, mean, std to get the value  
# To get what height 90% women are taller than   
norm.ppf((1-0.9), 160, 7)# prob, mean, std, to get the value  
# To generate random number  
norm.rvs(160, 7, size = 10)# mean, std, values

#### The central limit theorem

the more times you sample, the distribution looks more similar to normal distribution paradigm

This theory apply to sample distribution, standard deviation distribution and also the cat

##### Sample distribution

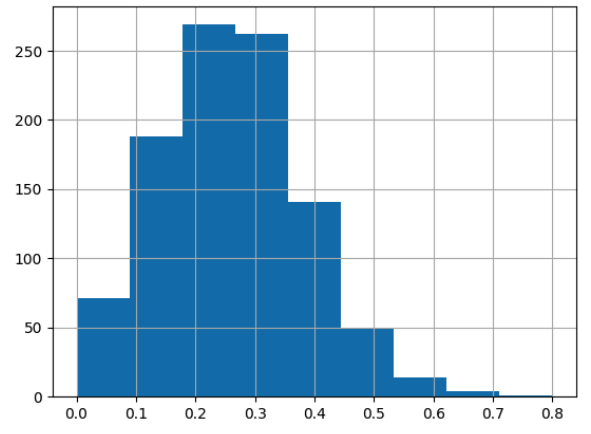
is the distribution of the sample mean after 100/1000/10000 trials

Samples should be random and independent

# Set seed to 104  
np.random.seed(104)  
  
sample\_means = []  
# Loop 100 times  
for i in range(100):  
 # Take sample of 20 num\_users  
 samp\_20 = amir\_deals['num\_users'].sample(20, replace=True)  
 # Calculate mean of samp\_20  
 samp\_20\_mean = np.mean(samp\_20)  
 # Append samp\_20\_mean to sample\_means  
 sample\_means.append(samp\_20\_mean)  
   
# Convert to Series and plot histogram  
sample\_means\_series = pd.Series(sample\_means)  
sample\_means\_series.hist()  
# Show plot  
plt.show()

# Set seed to 321  
np.random.seed(321)  
  
sample\_means = []  
# Loop 30 times to take 30 means   
# 根据已有数据，预估整个公司所有deals 的平均使用用户  
for i in range(30):  
 # Take sample of size 20 from num\_users col of all\_deals with replacement  
 cur\_sample = all\_deals['num\_users'].sample(20, replace = True)  
 # Take mean of cur\_sample  
 cur\_mean = np.mean(cur\_sample)  
 # Append cur\_mean to sample\_means  
 sample\_means.append(cur\_mean)  
  
# Print mean of sample\_means  
print(np.mean(sample\_means))  
  
# Print mean of num\_users in amir\_deals  
# amir订单的平均使用用户与整体公司订单平均使用用户的对比  
print(np.mean(amir\_deals['num\_users']))

##### std and sample distribution of proportion



该模型可以应用于数据不足的情况，通过sample生成值，根据模型，进行其他值的计算

#### Poisson distribution

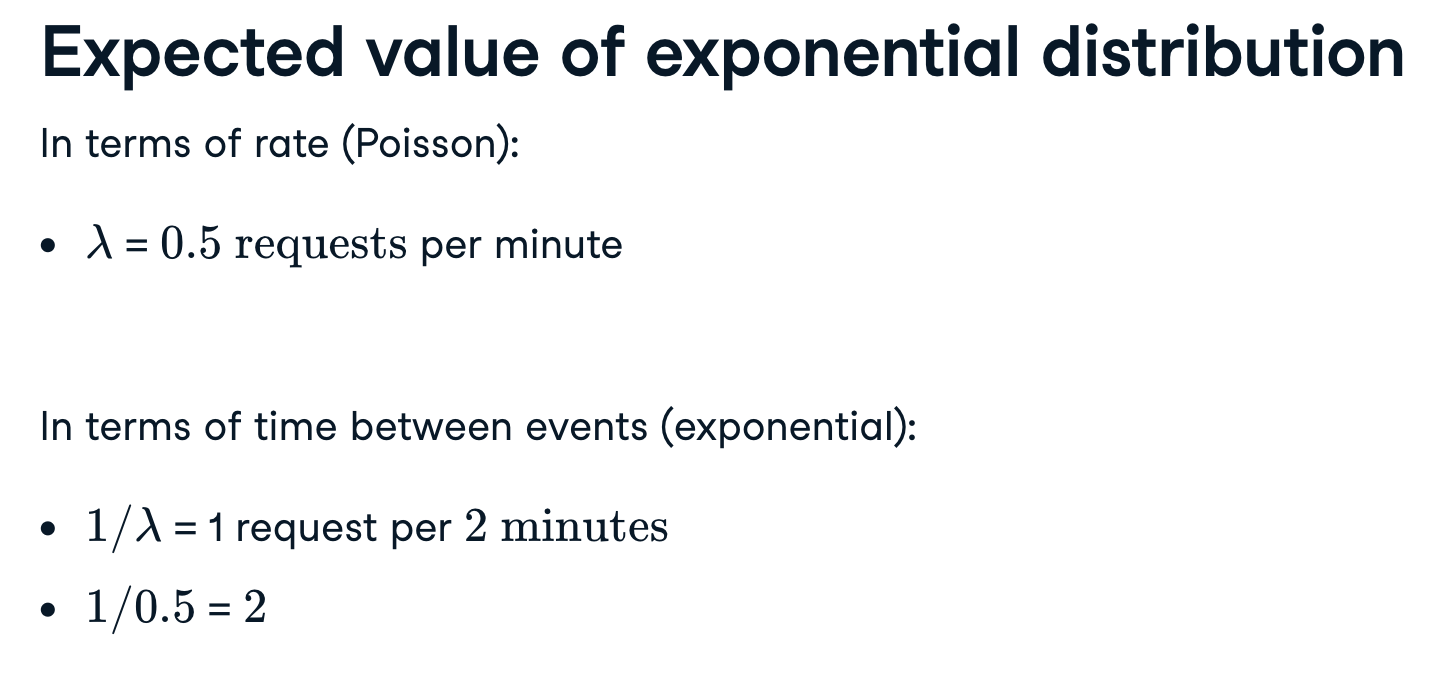
Probability of some # of events occurring over a fixed period of time

##### Lambda

Average number of events per time interval

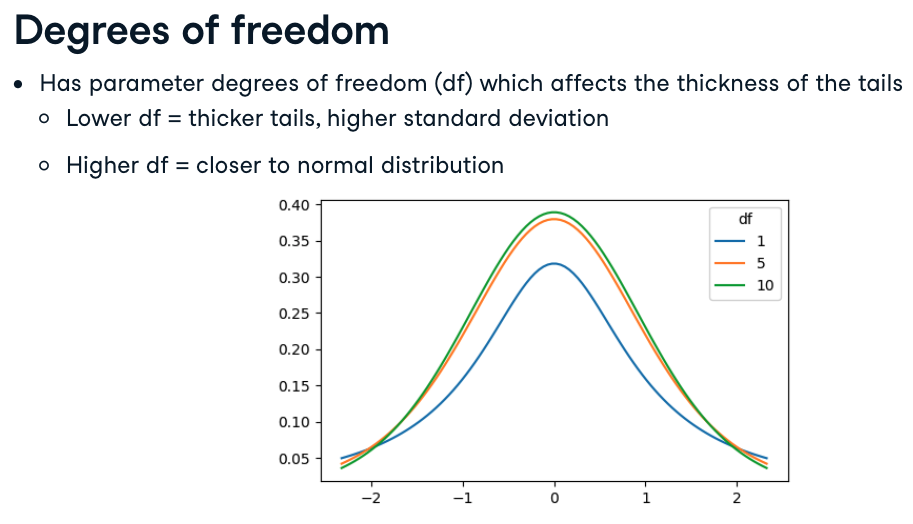
from scipy.stats import poisson  
# To calculate the possibility of a single value   
poisson.pmf(5,8)# values, average number   
# To To calculate the possibility less than or equal to a single value   
poisson.cdf(5,8)  
# To To calculate the possibility higher than a single value  
1-poisson.cdf(5,8)  
# sample from a poisson distribution  
possion.rvs(8, size = 10)# mean

#### Exponential distribution

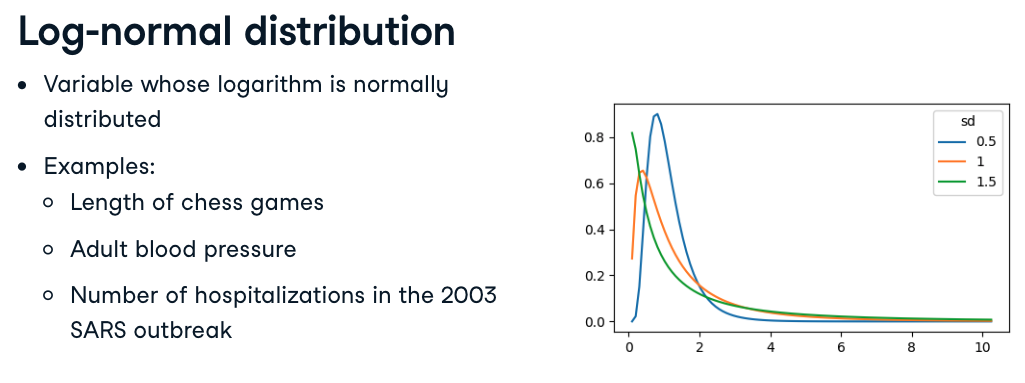


from scipy.stats import expo  
# calculate the possibility of waiting less than 1 min  
expon.cdf(1,scale = 2)# period, time interval  
# calculate the possibility of waiting more than 1 min  
1-expon.cdf(1,scale = 2)

## Student's t-distribution



## Log-normal distribution

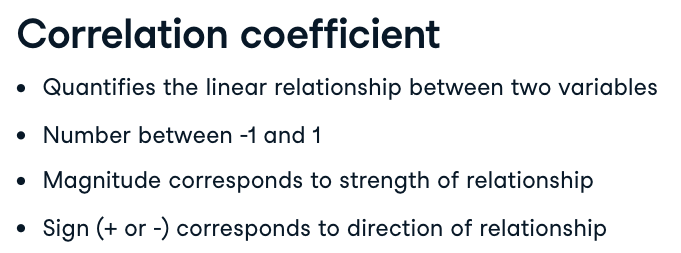


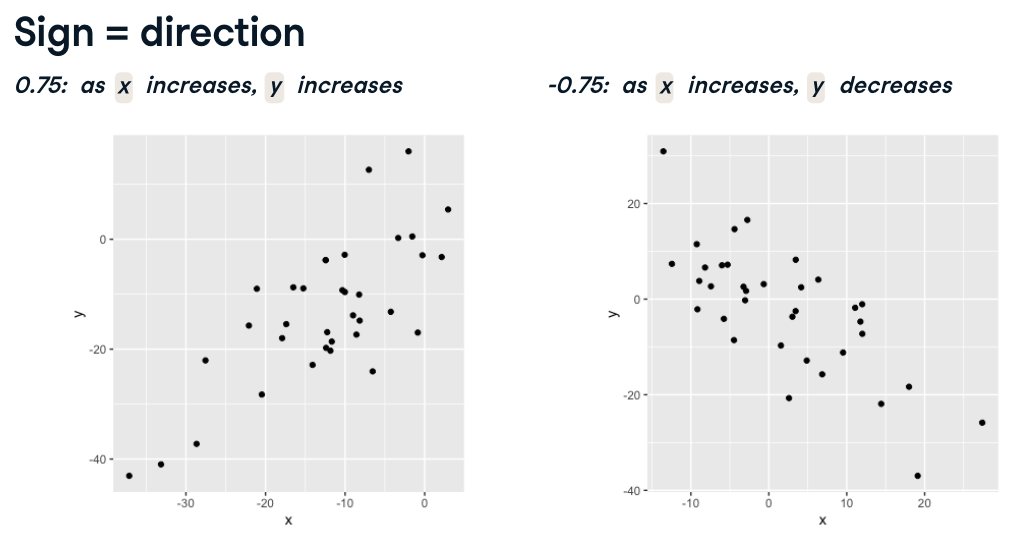
# Correlation

X = independent variable

Y = dependent variable

## Correlation coefficient





import seaborn as sns  
sns.lmplot( x = 'sleep\_total', y = 'sleep\_rem', data = msleep, ci = None)

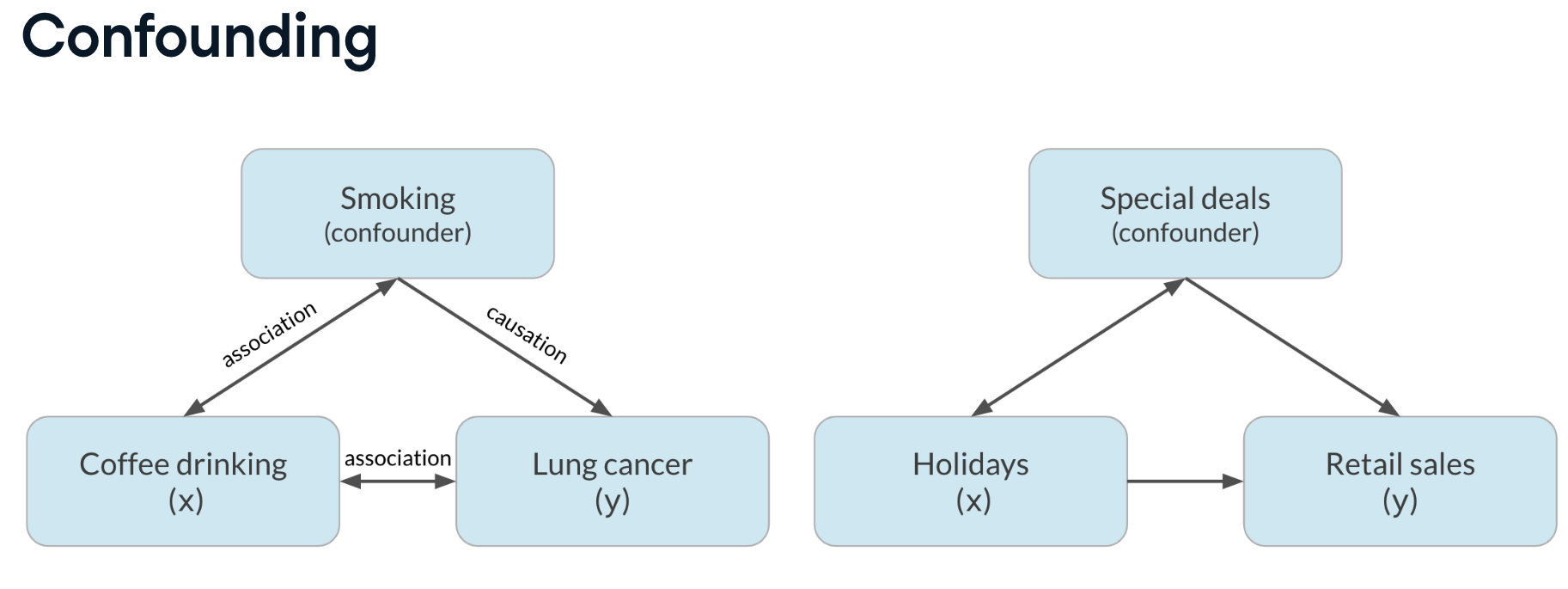
# To calculate the correlation coefficient  
msleep['sleep\_total'].corr(msleep['rem\_sleep'])  
# output 0.75  
   
msleep['rem\_sleep'].corr(msleep['sleep\_total'])  
# output 0.75  
# the result is the same, change the position of x and y

## Correlation caveats

Log transformation

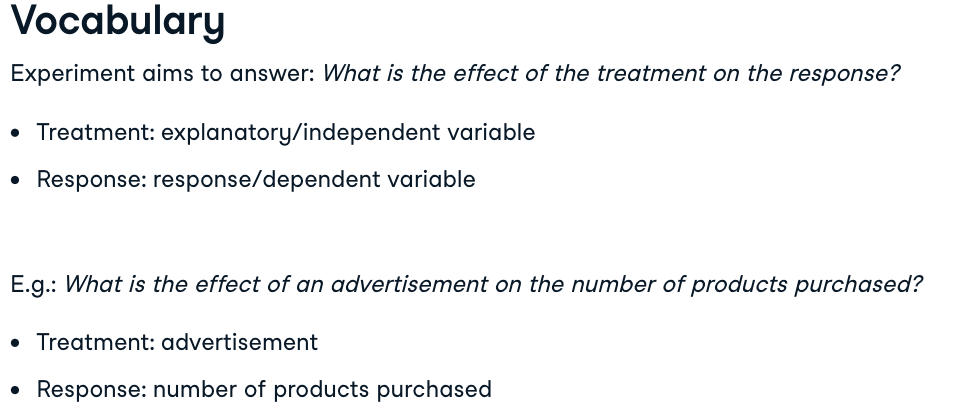
msleep['log\_bodywt']= np.log(mysleep['bodywt'])

sqare root sqrt()  
reciprocal transformation 1/x

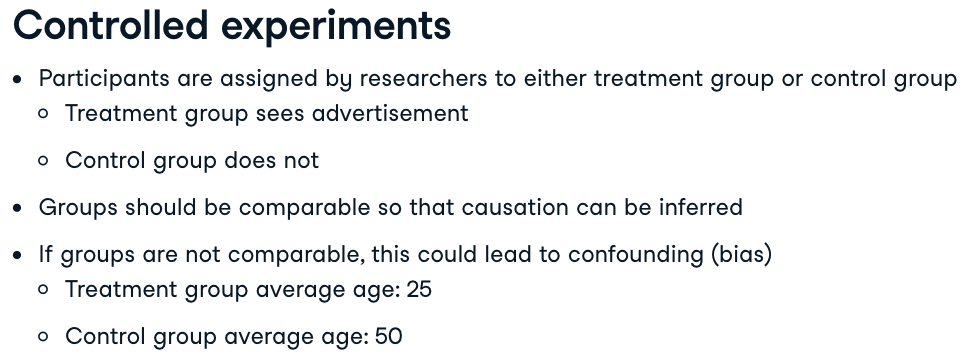


# Design of experiments

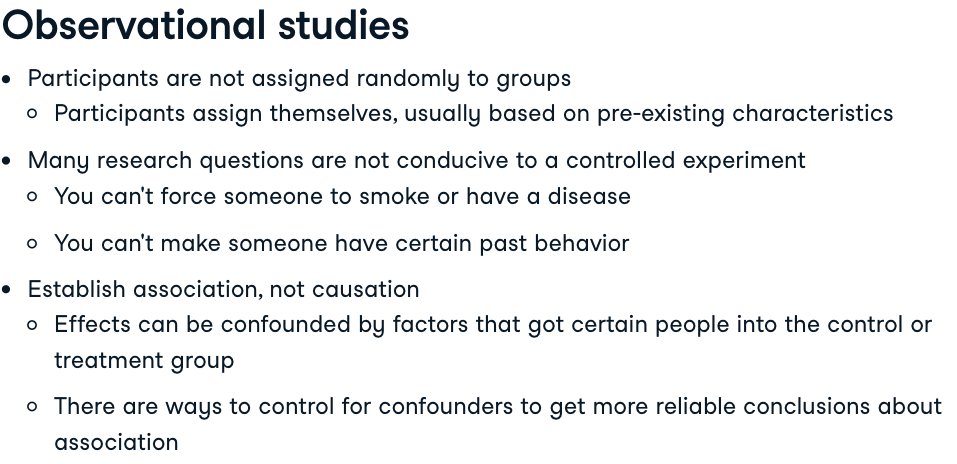
X对Y的影响有多大



## Controlled experiments



## Observational studies



## Longitudinal vs.Cross-sectional studies

