รู้จักกับ pandas

author: Wes McKinney http://wesmckinney.com/

Pandas คืออะไร เป็นไลบราลี่ในภาษา Python สำหรับจัดการและวิเคราะห์ข้อมูลที่เป็นแบบโครงสร้างทั้งรูปแบบมิติเดียวและหลายมิติ

pandas ย่อมาจาก "panel data" longitudinal data เป็นการเก็บข้อมูลที่เราสนใจต่อเนื่องหลาย ๆ ช่วงเวลา เช่น

- ยอดขายรถแยกตามยี่ห้อในแต่ละเดือน
- ราคาหุ้นปิดรายวันของหุ้นแต่ละตัว

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

ยอดจดทะเบียนรถยนต์นั่งส่วนบุคคลไม่เกิน 7 คน ปี 2559

http://apps.dlt.go.th/statistics_web/statistics.html

In [2]:
```
df = pd.read_csv('https://github.com/prasertcbs/basic-dataset/raw/master/par
df.head()
```

Out[2]:

	month	brand	1300cc	1600cc	1800cc	2000cc	gt2000cc	elec	total
0	1	ALFA ROMEO	0	0	0	0	1	0	1
1	1	AUDI	0	2	0	14	0	0	16
2	1	BENTLEY	0	0	0	0	2	0	2
3	1	BMW	1	45	0	586	64	0	696
4	1	CADILLAC	0	0	0	0	1	0	1

In [3]:
```
df[df.brand.str.contains('AUDI|BMW|BENZ')]
```

Out[3]:

	month	brand	1300cc	1600cc	1800cc	2000cc	gt2000cc	elec	total
1	1	AUDI	0	2	0	14	0	0	16
3	1	BMW	1	45	0	586	64	0	696
20	1	MERCEDES BENZ	0	161	46	372	605	0	1184
38	2	AUDI	0	0	0	10	0	0	10
39	2	BMW	0	60	0	689	52	0	801
55	2	MERCEDES BENZ	0	193	42	475	672	0	1382
77	3	AUDI	0	0	0	15	1	0	16
79	3	BMW	0	55	0	658	55	0	768
92	3	MERCEDES BENZ	0	174	41	554	650	0	1419
110	4	AUDI	0	0	0	7	0	0	7
112	4	BMW	1	39	0	443	44	0	527
127	4	MERCEDES BENZ	0	133	29	311	427	0	900
144	5	AUDI	0	1	0	13	0	0	14
146	5	BMW	0	42	0	522	35	0	599
162	5	MERCEDES BENZ	0	156	24	482	516	0	1178
181	6	AUDI	0	0	0	11	0	0	11
183	6	BMW	0	62	0	568	56	0	686
200	6	MERCEDES BENZ	0	151	36	509	503	0	1199
220	7	AUDI	0	1	0	5	1	0	7
222	7	BMW	0	32	0	483	48	0	563
239	7	MERCEDES BENZ	0	128	28	460	353	0	969
259	8	AUDI	0	0	0	9	0	0	9
260	8	BMW	0	52	0	543	71	0	666
277	8	MERCEDES BENZ	0	140	27	543	429	0	1139
295	9	AUDI	0	0	0	2	2	0	4
296	9	BMW	0	55	0	486	55	0	596
310	9	MERCEDES BENZ	0	108	19	578	404	0	1109

	month	brand	1300cc	1600cc	1800cc	2000cc	gt2000cc	elec	total
329	10	AUDI	0	0	0	9	0	0	9
330	10	BMW	0	50	0	451	28	0	529
346	10	MERCEDES BENZ	0	96	19	457	280	0	852
362	11	AUDI	0	0	0	9	3	0	12
364	11	BMW	0	64	0	461	35	0	560
378	11	MERCEDES BENZ	1	94	19	449	276	0	839
396	12	AUDI	0	0	0	7	2	0	9
398	12	BMW	0	40	0	269	19	0	328
413	12	MERCEDES BENZ	0	49	5	251	156	0	461

In [4]:
```
# df.sample(n=10)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 430 entries, 0 to 429
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   month     430 non-null    int64 
 1   brand     430 non-null    object
 2   1300cc    430 non-null    int64 
 3   1600cc    430 non-null    int64 
 4   1800cc    430 non-null    int64 
 5   2000cc    430 non-null    int64 
 6   gt2000cc  430 non-null    int64 
 7   elec      430 non-null    int64 
 8   total     430 non-null    int64 
dtypes: int64(8), object(1)
memory usage: 30.4+ KB
```

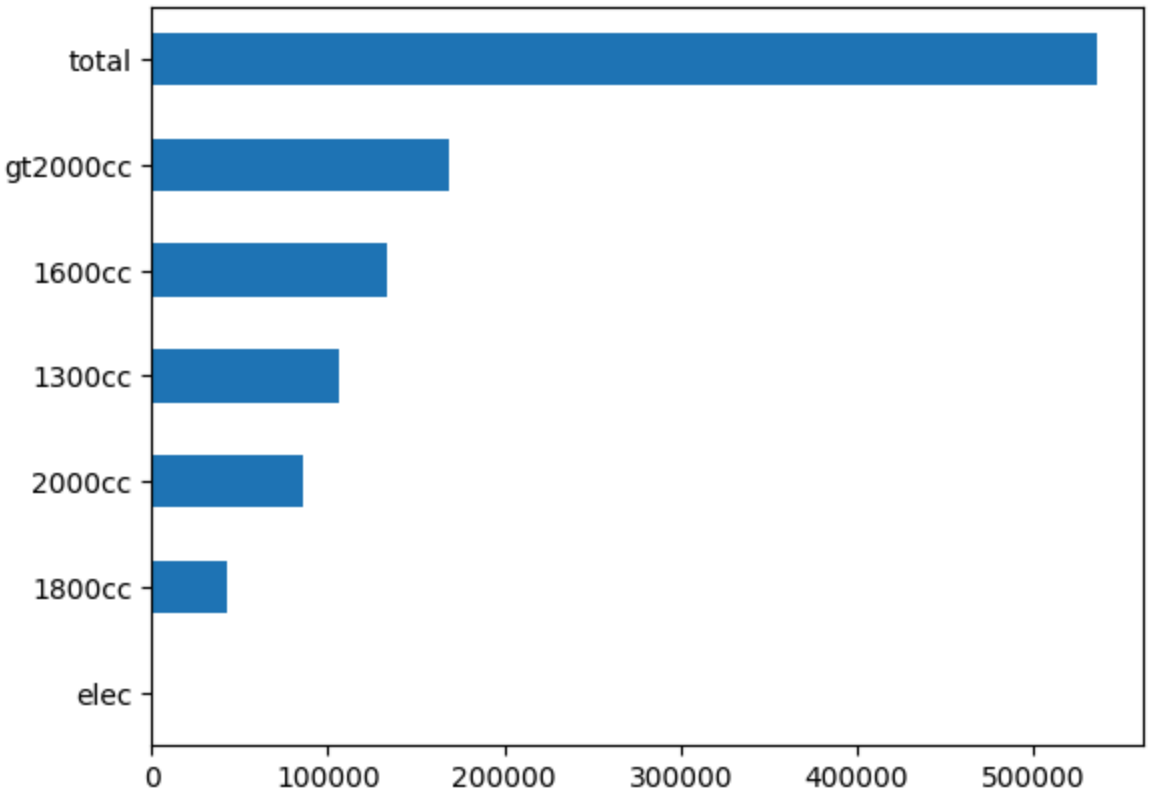
In [6]:
```
t = df.sum(numeric_only=True) # tip: set numeric_only=True
t
```

Out[6]:
```
month         2761
1300cc      106020
1600cc      133028
1800cc       42628
2000cc       85363
gt2000cc    168878
elec             2
total       535919
dtype: int64
```

In [7]:
```
t = df.drop(['month'], axis=1).sum(numeric_only=True) # tip: set numeric_onl
t
```

Out[7]:
```
1300cc      106020
1600cc      133028
1800cc       42628
2000cc       85363
gt2000cc    168878
elec             2
total       535919
dtype: int64
```

In [8]:
```
t.sort_values().plot(kind='barh');
```

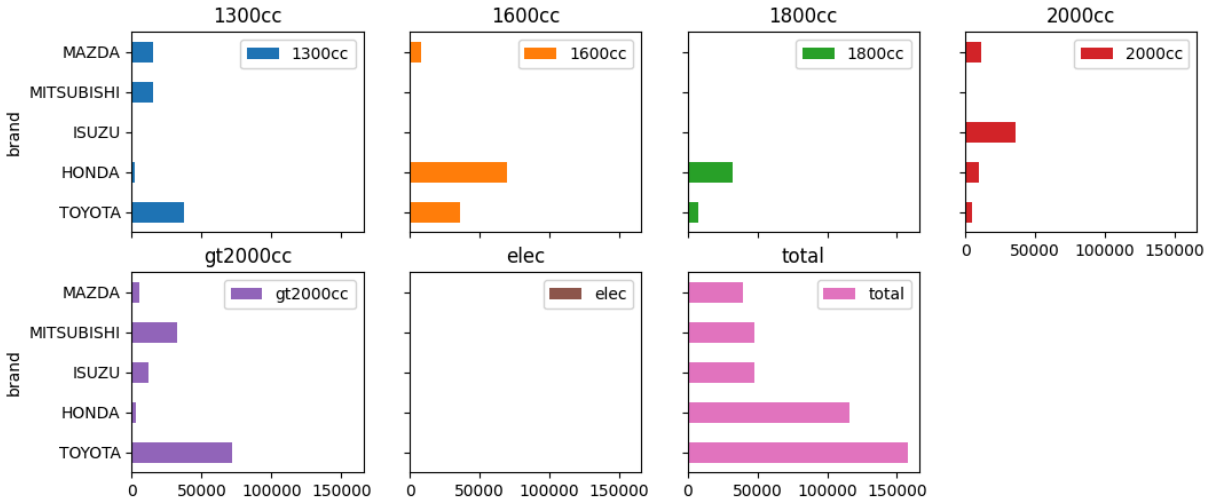


In [9]:
```
df.drop(['month'], axis=1).groupby('brand').sum().nlargest(5, 'total')
```

Out[9]:

brand	1300cc	1600cc	1800cc	2000cc	gt2000cc	elec	total
TOYOTA	37713	36105	7715	4359	72058	0	157950
HONDA	1890	69590	32547	9524	2575	0	116126
ISUZU	0	0	0	35943	12144	0	48087
MITSUBISHI	15342	3	64	173	32286	0	47868
MAZDA	14962	8423	0	11384	5137	0	39906

In [10]:
```
df.drop(['month'], axis=1).groupby('brand').sum().nlargest(5, 'total') \
    .plot(kind='barh', subplots=True, layout=(2, 4), figsize=(12, 5), sharey
```



In [11]:
```
df.groupby('brand').sum().nsmallest(10, 'total')
```

Out[11]:

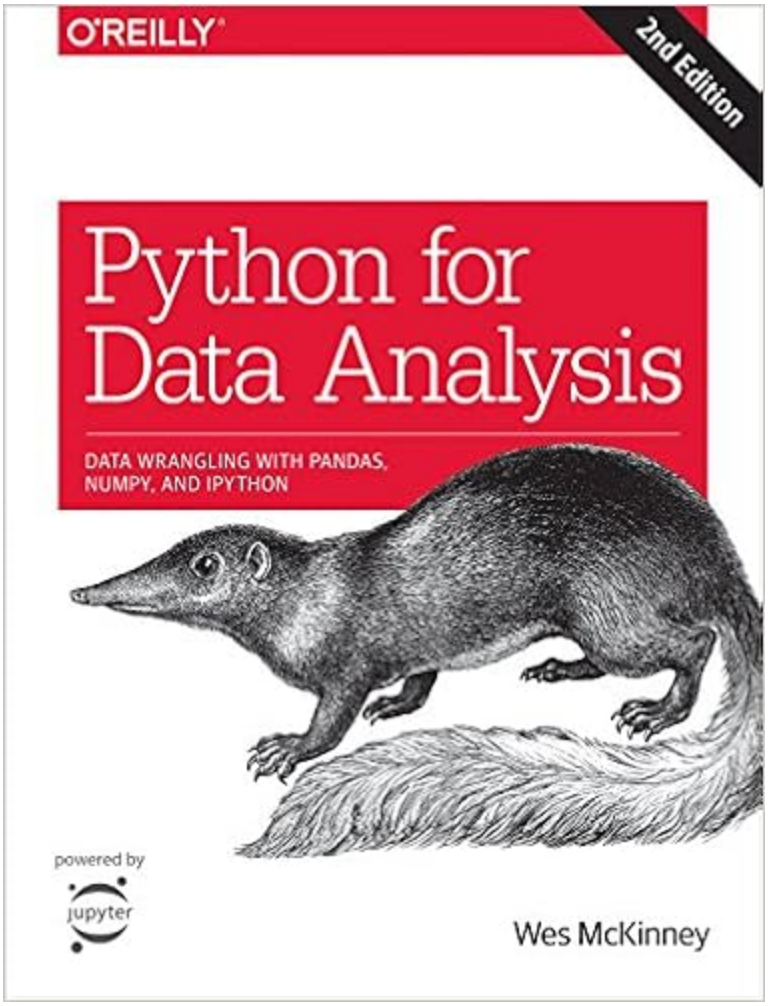
brand	month	1300cc	1600cc	1800cc	2000cc	gt2000cc	elec	total
CHRYSLER	10	0	0	0	1	0	0	1
HUMMER	10	0	0	0	0	1	0	1
LOTUS	12	0	0	0	0	1	0	1
NAZA	21	1	0	0	0	1	0	2
CHERY	12	0	0	0	3	0	0	3
DAIHATSU	7	4	0	0	0	0	0	4
MITSUOKA	6	0	0	0	3	1	0	4
PERODUA	8	0	4	0	0	0	0	4
SKODA	30	0	1	2	1	0	0	4
CADILLAC	32	0	0	0	0	5	0	5

In [12]:
```
# df.to_excel('data/panel_data.xlsx', index=False)
```

In [13]:
```
import requests
from PIL import Image # pillow package (Python Image Library)
import io

r = requests.get('https://images-na.ssl-images-amazon.com/images/I/51cUNf8zu
img = Image.open(io.BytesIO(r.content))
img
```

Out[13]:


O'REILLY
2nd Edition
Python for Data Analysis
DATA WRANGLING WITH PANDAS, NUMPY, AND IPYTHON
powered by
jupyter
Wes McKinney

In [ ]:

Introduction

You'll learn all about **pandas**, the most popular Python library for data analysis.

In this tutorial, you will learn how to create your own data, along with how to work with data that already exists.

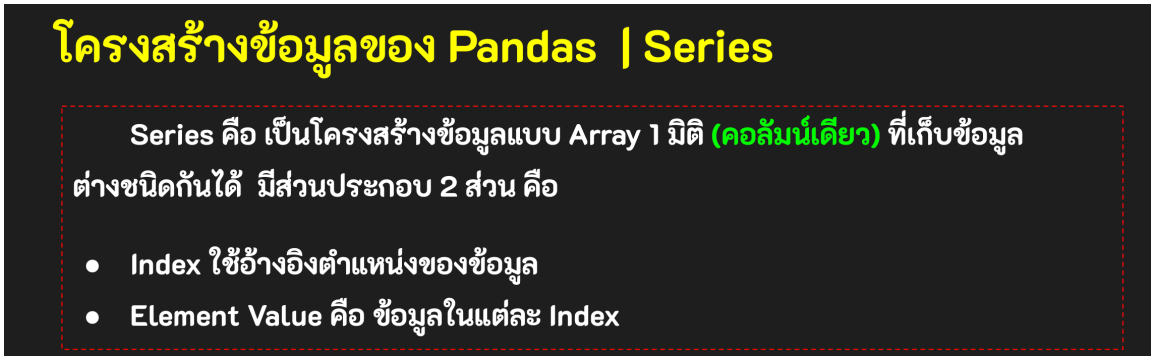
Getting started

To use pandas, you'll typically start with the following line of code.

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

Creating data

There are two core objects in pandas: the **DataFrame** and the **Series**.



DataFrame

A DataFrame is a table. It contains an array of individual *entries*, each of which has a certain *value*. Each entry corresponds to a row (or *record*) and a *column*.

For example, consider the following simple DataFrame:

In [2]:
```
pd.DataFrame({'Yes': [50, 21], 'No': [131, 2]})
```

Out[2]:

	Yes	No
0	50	131
1	21	2

In this example, the "0, No" entry has the value of 131. The "0, Yes" entry has a value of 50, and so on.

DataFrame entries are not limited to integers. For instance, here's a DataFrame whose values are strings:

In [3]:
```
pd.DataFrame({'Bob': ['I liked it.', 'It was awful.'], 'Sue': ['Pretty good.
```

Out[3]:

	Bob	Sue
0	I liked it.	Pretty good.
1	It was awful.	Bland.

We are using the `pd.DataFrame()` constructor to generate these DataFrame objects. The syntax for declaring a new one is a dictionary whose keys are the column names (`Bob` and `Sue` in this example), and whose values are a list of entries. This is the standard way of constructing a new DataFrame, and the one you are most likely to encounter.

The dictionary-list constructor assigns values to the *column labels*, but just uses an ascending count from 0 (0, 1, 2, 3, ...) for the *row labels*. Sometimes this is OK, but oftentimes we will want to assign these labels ourselves.

The list of row labels used in a DataFrame is known as an **Index**. We can assign values to it by using an `index` parameter in our constructor:

In [4]:
```
pd.DataFrame({'Bob': ['I liked it.', 'It was awful.'],
              'Sue': ['Pretty good.', 'Bland.']},
             index=['Product A', 'Product B'])
```

Out[4]:

	Bob	Sue
Product A	I liked it.	Pretty good.
Product B	It was awful.	Bland.

Series

A Series, by contrast, is a sequence of data values. If a DataFrame is a table, a Series is a list. And in fact you can create one with nothing more than a list:

In [5]:
```
pd.Series([1, 2, 3, 4, 5])
```

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

A Series is, in essence, a single column of a DataFrame. So you can assign row labels to the Series the same way as before, using an `index` parameter. However, a Series does not have a column name, it only has one overall `name`:

In [6]:
```
pd.Series([30, 35, 40], index=['2015 Sales', '2016 Sales', '2017 Sales'], na
```

Out[6]:
```
2015 Sales    30
2016 Sales    35
2017 Sales    40
Name: Product A, dtype: int64
```

In [7]:
```
pd.DataFrame(pd.Series([30, 35, 40], index=['2015 Sales', '2016 Sales', '201
```

Out[7]:

	Product A
2015 Sales	30
2016 Sales	35
2017 Sales	40

The Series and the DataFrame are intimately related. It's helpful to think of a DataFrame as actually being just a bunch of Series "glued together". We'll see more of this in the next section of this tutorial.

Reading data files

Being able to create a DataFrame or Series by hand is handy. But, most of the time, we won't actually be creating our own data by hand. Instead, we'll be working with data that already exists.

Data can be stored in any of a number of different forms and formats. By far the most basic of these is the humble CSV file. When you open a CSV file you get something that looks like this:

```
Product A,Product B,Product C,
30,21,9,
35,34,1,
41,11,11
```

So a CSV file is a table of values separated by commas. Hence the name: "Comma-Separated Values", or CSV.

Let's now set aside our toy datasets and see what a real dataset looks like when we read it into a DataFrame. We'll use the `pd.read_csv()` function to read the data into a DataFrame. This goes thusly:

In [8]:
```
wine_reviews = pd.read_csv("datasets/winemag-data-130k-v2.csv")
```

We can use the `shape` attribute to check how large the resulting DataFrame is:

In [9]:
```
wine_reviews.shape
```

Out[9]:
```
(65499, 14)
```

So our new DataFrame has 65,499 records split across 14 different columns. That's almost 1 million entries!

We can examine the contents of the resultant DataFrame using the `head()` command, which grabs the first five rows:

In [10]:
```
wine_reviews.head()   #แสดงผล 5 แถวแรก
```

Out[10]:

	Unnamed: 0	country	description	designation	points	price	province	region_1	
0	0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	
1	1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	
2	2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	W
3	3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	
4	4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	W

In [11]:
```
wine_reviews.head(10)   #แสดงผล 10 แถวแรก
```

Out[11]:

	Unnamed: 0	country	description	designation	points	price	province	region_
0	0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etn
1	1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	Nal
2	2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamett Valle
3	3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lak Michiga Shor
4	4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamett Valle
5	5	Spain	Blackberry and raspberry aromas show a typical...	Ars In Vitro	87	15.0	Northern Spain	Navarr
6	6	Italy	Here's a bright, informal red that opens with ...	Belsito	87	16.0	Sicily & Sardinia	Vittori
7	7	France	This dry and restrained wine offers spice in p...	NaN	87	24.0	Alsace	Alsac
8	8	Germany	Savory dried thyme notes accent sunnier flavor...	Shine	87	12.0	Rheinhessen	Nal
9	9	France	This has great depth	Les Natures	87	27.0	Alsace	Alsac

	Unnamed: 0	country	description	designation	points	price	province	region_
			of flavor with its fresh ...					

In [12]:
```
wine_reviews.tail()   #แสดงผล 5 แถวสุดท้าย
```

Out[12]:

	Unnamed: 0	country	description	designation	points	price	province	region_
65494	65494	France	Made from young vines from the Vaulorent porti...	Fourchaume Premier Cru	90	45.0	Burgundy	Chab
65495	65495	Australia	This is a big, fat, almost sweet-tasting Caber...	NaN	90	22.0	South Australia	McLare Va
65496	65496	US	Much improved over the unripe 2005, Fritz's 20...	Estate	90	20.0	California	D Cre Vall
65497	65497	US	This wine wears its 15.8% alcohol better than ...	Block 24	90	31.0	California	Nap Vall
65498	65498	Spain	A unique take on Manzanilla Sherry, which is o...	Manzanilla	90	10.0	Andalucia	Jer

The `pd.read_csv()` function is well-endowed, with over 30 optional parameters you can specify. For example, you can see in this dataset that the CSV file has a built-in index, which pandas did not pick up on automatically. To make pandas use that column for the index (instead of creating a new one from scratch), we can specify an `index_col`.

In [13]:
```
wine_reviews = pd.read_csv("datasets/winemag-data-130k-v2.csv", index_col=0)
wine_reviews.head()
```

Out[13]:

	country	description	designation	points	price	province	region_1	region_2	t
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	

Your turn

If you haven't started the exercise, you can **get started here**.

Introduction

Selecting specific values of a pandas DataFrame or Series to work on is an implicit step in almost any data operation you'll run, so one of the first things you need to learn in working with data in Python is how to go about selecting the data points relevant to you quickly and effectively.

In [1]:
```
import pandas as pd
reviews = pd.read_csv("datasets/winemag-data-130k-v2.csv", index_col=0)
```

Native accessors

Native Python objects provide good ways of indexing data. Pandas carries all of these over, which helps make it easy to start with.

Consider this DataFrame:

In [2]:
```
reviews
```

Out[2]:

	country	description	designation	points	price	province	region_1	regior
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	N
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	N
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willame Va
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	N
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willame Va
...	...	...	...	...	...	...	...	
65494	France	Made from young vines from the Vaulorent porti...	Fourchaume Premier Cru	90	45.0	Burgundy	Chablis	N
65495	Australia	This is a big, fat, almost sweet-tasting Caber...	NaN	90	22.0	South Australia	McLaren Vale	N
65496	US	Much improved over the unripe 2005, Fritz's 20...	Estate	90	20.0	California	Dry Creek Valley	Sonc
65497	US	This wine wears its 15.8% alcohol	Block 24	90	31.0	California	Napa Valley	Na

	country	description	designation	points	price	province	region_1	regior
		better than ...						
65498	Spain	A unique take on Manzanilla Sherry, which is o...	Manzanilla	90	10.0	Andalucia	Jerez	N

65499 rows × 13 columns

In Python, we can access the property of an object by accessing it as an attribute. A `book` object, for example, might have a `title` property, which we can access by calling `book.title`. Columns in a pandas DataFrame work in much the same way.

Hence to access the `country` property of `reviews` we can use:

การเลือกคอลัมน์จาก DataFrame

- แสดงค่าในคอลัมน์เป็น Series

In [3]:
```
reviews.country
```

Out[3]:
```
0           Italy
1        Portugal
2              US
3              US
4              US
           ...
65494       France
65495    Australia
65496           US
65497           US
65498        Spain
Name: country, Length: 65499, dtype: object
```

If we have a Python dictionary, we can access its values using the indexing (`[]`) operator. We can do the same with columns in a DataFrame:

In [4]:
```
reviews['country']
```

Out[4]:
```
0            Italy
1         Portugal
2               US
3               US
4               US
           ...
65494       France
65495    Australia
65496           US
65497           US
65498        Spain
Name: country, Length: 65499, dtype: object
```

- แสดงค่าในคอลัมน์เป็น DataFrame

In [5]:
```
reviews[['country']]
```

Out[5]:

	country
0	Italy
1	Portugal
2	US
3	US
4	US
...	...
65494	France
65495	Australia
65496	US
65497	US
65498	Spain

65499 rows × 1 columns

In [6]:
```
my_cols = ['country','points','province','taster_name']
reviews[my_cols]
```

Out[6]:

	country	points	province	taster_name
0	Italy	87	Sicily & Sardinia	Kerin O'Keefe
1	Portugal	87	Douro	Roger Voss
2	US	87	Oregon	Paul Gregutt
3	US	87	Michigan	Alexander Peartree
4	US	87	Oregon	Paul Gregutt
...	...	...	...	...
65494	France	90	Burgundy	Roger Voss
65495	Australia	90	South Australia	Joe Czerwinski
65496	US	90	California	NaN
65497	US	90	California	NaN
65498	Spain	90	Andalucia	Michael Schachner

65499 rows × 4 columns

These are the two ways of selecting a specific Series out of a DataFrame. Neither of them is more or less syntactically valid than the other, but the indexing operator `[]` does have the advantage that it can handle column names with reserved characters in them (e.g. if we had a `country providence` column, `reviews.country providence` wouldn't work).

Doesn't a pandas Series look kind of like a fancy dictionary? It pretty much is, so it's no surprise that, to drill down to a single specific value, we need only use the indexing operator `[]` once more:

การเลือกตำแหน่งจาก DataFrame

In [7]:
```
reviews.country[0]
```

Out[7]:
```
'Italy'
```

In [8]:
```
reviews['country'][0]
```

Out[8]:
```
'Italy'
```

Indexing in pandas

The indexing operator and attribute selection are nice because they work just like they do in the rest of the Python ecosystem. As a novice, this makes them easy to pick up

and use. However, pandas has its own accessor operators, `loc` and `iloc`. For more advanced operations, these are the ones you're supposed to be using.

การเลือกแถว และ คอลัมน์

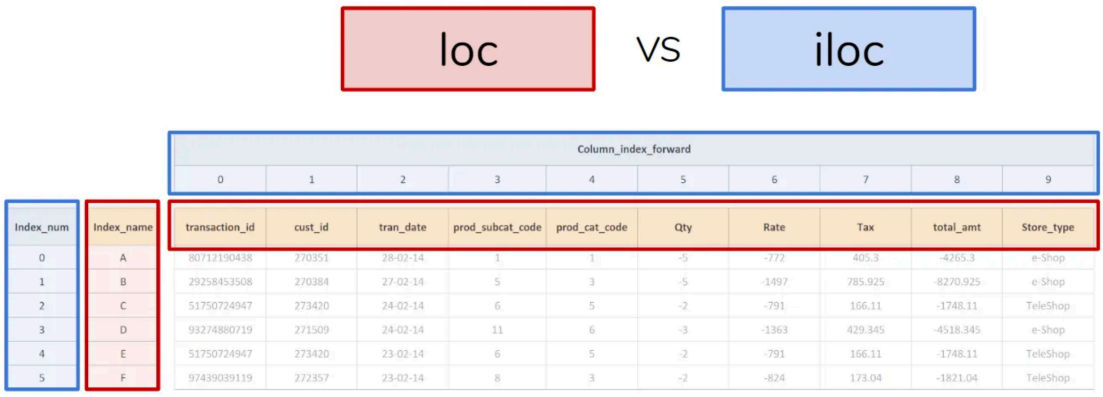
การที่จะเลือกแถว และคอลัมน์ นั้นเราจะต้องระบุว่าต้องการจะเลือกข้อมูลที่อยู่ใน แถว อะไร และคอลัมน์อะไร หรือ Location ที่มีข้อมูลที่ต้องการอยู่นั้นเอง ซึ่งมี 2 วิธี ดังนี้

1. loc : การที่จะใช้ loc นั้น จะต้องระบุชื่อของแถวและ ชื่อของคอลัมน์ลงไปตรงๆ

```
# dataframe.loc['index_name' , 'column_name']
```

2. iloc : การที่จะใช้ iloc นั้นไม่สนใจชื่อ แต่สนใจตำแหน่งของแถว และ ตำแหน่งของคอลัมน์

```
dataframe.iloc['row_index_number' , 'column_index_number']
```



Index-based selection

Pandas indexing works in one of two paradigms. The first is **index-based selection**: selecting data based on its numerical position in the data. `iloc` follows this paradigm.

To select the first row of data in a DataFrame, we may use the following:

In [9]:
```
reviews.iloc[0]  #ถ้าแถวเดียวจะแสดงผลเป็น Series
```

Out[9]:
```
country                                                     Italy
description             Aromas include tropical fruit, broom, brimston...
designation                                          Vulkà Bianco
points                                                         87
price                                                         NaN
province                                          Sicily & Sardinia
region_1                                                    Etna
region_2                                                     NaN
taster_name                                        Kerin O’Keefe
taster_twitter_handle                                 @kerinokeefe
title                                 Nicosia 2013 Vulkà Bianco  (Etna)
variety                                             White Blend
winery                                                  Nicosia
Name: 0, dtype: object
```

In [10]:
```
reviews.iloc[[0]] #ถ้าอยากให้แถวเดียวแสดงผลเป็น DataFrame ต้องใส่ [] คลุมอีกที
```

Out[10]:

	country	description	designation	points	price	province	region_1	region_2	tast
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	

In [11]:
```
reviews.iloc[0:2] #ถ้าหลายแถวจะแสดงผลเป็น DataFrame
```

Out[11]:

	country	description	designation	points	price	province	region_1	region_2	tast
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	R

Both `loc` and `iloc` are row-first, column-second. This is the opposite of what we do in native Python, which is column-first, row-second.

This means that it's marginally easier to retrieve rows, and marginally harder to get retrieve columns. To get a column with `iloc`, we can do the following:

In [12]:
```
reviews.iloc[:, 0]
```

Out[12]:
```
0           Italy
1        Portugal
2              US
3              US
4              US
           ...
65494       France
65495    Australia
65496           US
65497           US
65498        Spain
Name: country, Length: 65499, dtype: object
```

On its own, the `:` operator, which also comes from native Python, means "everything". When combined with other selectors, however, it can be used to indicate a range of values. For example, to select the `country` column from just the first, second, and third row, we would do:

In [13]:
```
reviews.iloc[:3, 0]
```

Out[13]:
```
0       Italy
1    Portugal
2          US
Name: country, dtype: object
```

Or, to select just the second and third entries, we would do:

In [14]:
```
reviews.iloc[1:3, 0]
```

Out[14]:
```
1    Portugal
2          US
Name: country, dtype: object
```

In [15]:
```
reviews.iloc[2:4, :5]
```

Out[15]:

	country	description	designation	points	price
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0

It's also possible to pass a list:

In [16]:
```
reviews.iloc[[2, 3], [0,1,2,3,4]]
```

Out[16]:

	country	description	designation	points	price
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0

Finally, it's worth knowing that negative numbers can be used in selection. This will start counting forwards from the *end* of the values. So for example here are the last five elements of the dataset.

In [17]:
```
reviews.iloc[-5:]
```

Out[17]:

	country	description	designation	points	price	province	region_1	region_2
65494	France	Made from young vines from the Vaulorent porti...	Fourchaume Premier Cru	90	45.0	Burgundy	Chablis	NaN
65495	Australia	This is a big, fat, almost sweet-tasting Caber...	NaN	90	22.0	South Australia	McLaren Vale	NaN
65496	US	Much improved over the unripe 2005, Fritz's 20...	Estate	90	20.0	California	Dry Creek Valley	Sonoma
65497	US	This wine wears its 15.8% alcohol better than ...	Block 24	90	31.0	California	Napa Valley	Napa
65498	Spain	A unique take on Manzanilla Sherry, which is o...	Manzanilla	90	10.0	Andalucia	Jerez	NaN

Label-based selection

The second paradigm for attribute selection is the one followed by the `loc` operator: **label-based selection**. In this paradigm, it's the data index value, not its position, which

For example, to get the first entry in `reviews`, we would now do the following:

In [18]:
```
reviews.loc[0, 'country']
```

Out[18]:
```
'Italy'
```

`iloc` is conceptually simpler than `loc` because it ignores the dataset's indices. When we use `iloc` we treat the dataset like a big matrix (a list of lists), one that we have to index into by position. `loc`, by contrast, uses the information in the indices to do its work. Since your dataset usually has meaningful indices, it's usually easier to do things using `loc` instead. For example, here's one operation that's much easier using `loc`:

In [19]:
```
reviews.loc[:, ['taster_name', 'taster_twitter_handle', 'points']]
```

Out[19]:

	taster_name	taster_twitter_handle	points
0	Kerin O'Keefe	@kerinokeefe	87
1	Roger Voss	@vossroger	87
2	Paul Gregutt	@paulgwine	87
3	Alexander Peartree	NaN	87
4	Paul Gregutt	@paulgwine	87
...	...	...	...
65494	Roger Voss	@vossroger	90
65495	Joe Czerwinski	@JoeCz	90
65496	NaN	NaN	90
65497	NaN	NaN	90
65498	Michael Schachner	@wineschach	90

65499 rows × 3 columns

Choosing between `loc` and `iloc`

When choosing or transitioning between `loc` and `iloc`, there is one "gotcha" worth keeping in mind, which is that the two methods use slightly different indexing schemes.

`iloc` uses the Python stdlib indexing scheme, where the first element of the range is included and the last one excluded. So `0:10` will select entries `0,...,9`. `loc`, meanwhile, indexes inclusively. So `0:10` will select entries `0,...,10`.

Why the change? Remember that loc can index any stdlib type: strings, for example. If we have a DataFrame with index values `Apples, ..., Potatoes, ...`, and we want to select "all the alphabetical fruit choices between Apples and Potatoes", then it's

a lot more convenient to index `df.loc['Apples':'Potatoes']` than it is to index something like `df.loc['Apples', 'Potatoet']` (`t` coming after `s` in the alphabet).

This is particularly confusing when the DataFrame index is a simple numerical list, e.g. `0,...,1000`. In this case `df.iloc[0:1000]` will return 1000 entries, while `df.loc[0:1000]` return 1001 of them! To get 1000 elements using `loc`, you will need to go one lower and ask for `df.loc[0:999]`.

Otherwise, the semantics of using `loc` are the same as those for `iloc`.

Manipulating the index

Label-based selection derives its power from the labels in the index. Critically, the index we use is not immutable. We can manipulate the index in any way we see fit.

The `set_index()` method can be used to do the job. Here is what happens when we `set_index` to the `title` field:

In [20]:
```
reviews.set_index("title")
```

Out[20]:

title	country	description	designation	points	price	province	region_1	
Nicosia 2013 Vulkà Bianco (Etna)	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	
Quinta dos Avidagos 2011 Avidagos Red (Douro)	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	
Rainstorm 2013 Pinot Gris (Willamette Valley)	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	W
St. Julian 2013 Reserve Late Harvest Riesling (Lake Michigan Shore)	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	
Sweet Cheeks 2012 Vintner's Reserve Wild Child Block Pinot Noir (Willamette Valley)	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	W
...	...	...	...	...	...	...	...	
William Fèvre 2005 Fourchaume Premier Cru (Chablis)	France	Made from young vines from the Vaulorent porti...	Fourchaume Premier Cru	90	45.0	Burgundy	Chablis	
Tapestry 2005 Cabernet Sauvignon (McLaren Vale)	Australia	This is a big, fat, almost sweet-tasting Caber...	NaN	90	22.0	South Australia	McLaren Vale	
Estate	US	Much improved	Estate	90	20.0	California	Dry Creek Valley	

title	country	description	designation	points	price	province	region_1
Sauvignon Blanc (Dry Creek Valley)		over the unripe 2005, Fritz's 20...					
Hendry 2004 Block 24 Primitivo (Napa Valley)	US	This wine wears its 15.8% alcohol better than ...	Block 24	90	31.0	California	Napa Valley
Bodegas Dios Baco S.L. NV Manzanilla Sherry (Jerez)	Spain	A unique take on Manzanilla Sherry, which is o...	Manzanilla	90	10.0	Andalucia	Jerez

65499 rows × 12 columns

This is useful if you can come up with an index for the dataset which is better than the current one.

Conditional selection

So far we've been indexing various strides of data, using structural properties of the DataFrame itself. To do *interesting* things with the data, however, we often need to ask questions based on conditions.

For example, suppose that we're interested specifically in better-than-average wines produced in Italy.

We can start by checking if each wine is Italian or not:

In [21]:
```
reviews.country == 'Italy'
```

Out[21]:
```
0         True
1        False
2        False
3        False
4        False
         ...
65494    False
65495    False
65496    False
65497    False
         False
Name: country, Length: 65499, dtype: bool
```

This operation produced a Series of `True` / `False` booleans based on the `country` of each record. This result can then be used inside of `loc` to select the relevant data:

In [22]:
```
reviews.loc[reviews.country == 'Italy']
```

Out[22]:

	country	description	designation	points	price	province	region_1	reg
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	
6	Italy	Here's a bright, informal red that opens with ...	Belsito	87	16.0	Sicily & Sardinia	Vittoria	
13	Italy	This is dominated by oak and oak-driven aromas...	Rosso	87	NaN	Sicily & Sardinia	Etna	
22	Italy	Delicate aromas recall white flower and citrus...	Ficiligno	87	19.0	Sicily & Sardinia	Sicilia	
24	Italy	Aromas of prune, blackcurrant, toast and oak c...	Aynat	87	35.0	Sicily & Sardinia	Sicilia	
...	...	...	...	...	...	...	...	
65466	Italy	Earthy truffle, porcini mushroom, herb and gam...	NaN	88	70.0	Tuscany	Brunello di Montalcino	
65474	Italy	Made of 70% Syrah, 15% Sangiovese and 15% Merl...	Taneto	88	25.0	Tuscany	Toscana	
65476	Italy	Rose, violet, sour berry and tilled earth arom...	Prugnolo	88	25.0	Tuscany	Rosso di Montepulciano	
65477	Italy	Made of 65% Merlot, 25% Cabernet Sauvignon, 5%...	Ruit Hora	88	30.0	Tuscany	Bolgheri	

	country	description	designation	points	price	province	region_1	reg
65478	Italy	Aromas suggesting French oak, coconut and spic...	NaN	88	36.0	Tuscany	Vino Nobile di Montepulciano	

10005 rows × 13 columns

In [23]:
```
reviews[reviews.country == 'Italy']
```

Out[23]:

	country	description	designation	points	price	province	region_1	reg
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	
6	Italy	Here's a bright, informal red that opens with ...	Belsito	87	16.0	Sicily & Sardinia	Vittoria	
13	Italy	This is dominated by oak and oak-driven aromas...	Rosso	87	NaN	Sicily & Sardinia	Etna	
22	Italy	Delicate aromas recall white flower and citrus...	Ficiligno	87	19.0	Sicily & Sardinia	Sicilia	
24	Italy	Aromas of prune, blackcurrant, toast and oak c...	Aynat	87	35.0	Sicily & Sardinia	Sicilia	
...	...	...	...	...	...	...	...	
65466	Italy	Earthy truffle, porcini mushroom, herb and gam...	NaN	88	70.0	Tuscany	Brunello di Montalcino	
65474	Italy	Made of 70% Syrah, 15% Sangiovese and 15% Merl...	Taneto	88	25.0	Tuscany	Toscana	
65476	Italy	Rose, violet, sour berry and tilled earth arom...	Prugnolo	88	25.0	Tuscany	Rosso di Montepulciano	
65477	Italy	Made of 65% Merlot, 25% Cabernet Sauvignon, 5%...	Ruit Hora	88	30.0	Tuscany	Bolgheri	

	country	description	designation	points	price	province	region_1	reg
65478	Italy	Aromas suggesting French oak, coconut and spic...	NaN	88	36.0	Tuscany	Vino Nobile di Montepulciano	

10005 rows × 13 columns

This DataFrame has ~20,000 rows. The original had ~130,000. That means that around 15% of wines originate from Italy.

We also wanted to know which ones are better than average. Wines are reviewed on a 80-to-100 point scale, so this could mean wines that accrued at least 90 points.

We can use the ampersand (`&`) to bring the two questions together:

In [24]:
```
reviews.loc[(reviews.country == 'Italy') & (reviews.points >= 90)]
```

Out[24]:

	country	description	designation	points	price	province	region_1	re
120	Italy	Slightly backward, particularly given the vint...	Bricco Rocche Prapó	92	70.0	Piedmont	Barolo	
130	Italy	At the first it was quite muted and subdued, b...	Bricco Rocche Brunate	91	70.0	Piedmont	Barolo	
133	Italy	Einaudi's wines have been improving lately, an...	NaN	91	68.0	Piedmont	Barolo	
135	Italy	The color is just beginning to show signs of b...	Sorano	91	60.0	Piedmont	Barolo	
140	Italy	A big, fat, luscious wine with plenty of toast...	Costa Bruna	90	26.0	Piedmont	Barbera d'Alba	
...	...	...	...	...	...	...	...	
65225	Italy	You'll love the dark intensity and generous ar...	Vigna Manapetra Riserva	93	58.0	Tuscany	Brunello di Montalcino	
65226	Italy	Brunello Madonna Nera is a new product (this w...	Madonna Nera	92	NaN	Tuscany	Brunello di Montalcino	
65362	Italy	This stunning single-vineyard selection is one...	La Rocca	95	31.0	Veneto	Soave Classico	
65365	Italy	Stunning and sophisticated, it leads with inte...	Sanct Valentin	94	40.0	Northeastern Italy	Alto Adige	

	country	description	designation	points	price	province	region_1	re
65399	Italy	Aesthetics and elegance are important values t...	Nectar Dei	91	65.0	Tuscany	Maremma	

3479 rows × 13 columns

In [25]:
```
reviews[(reviews.country == 'Italy') & (reviews.points >= 90)]
```

Out[25]:

	country	description	designation	points	price	province	region_1	re
120	Italy	Slightly backward, particularly given the vint...	Bricco Rocche Prapó	92	70.0	Piedmont	Barolo	
130	Italy	At the first it was quite muted and subdued, b...	Bricco Rocche Brunate	91	70.0	Piedmont	Barolo	
133	Italy	Einaudi's wines have been improving lately, an...	NaN	91	68.0	Piedmont	Barolo	
135	Italy	The color is just beginning to show signs of b...	Sorano	91	60.0	Piedmont	Barolo	
140	Italy	A big, fat, luscious wine with plenty of toast...	Costa Bruna	90	26.0	Piedmont	Barbera d'Alba	
...	...	...	...	...	...	...	...	
65225	Italy	You'll love the dark intensity and generous ar...	Vigna Manapetra Riserva	93	58.0	Tuscany	Brunello di Montalcino	
65226	Italy	Brunello Madonna Nera is a new product (this w...	Madonna Nera	92	NaN	Tuscany	Brunello di Montalcino	
65362	Italy	This stunning single-vineyard selection is one...	La Rocca	95	31.0	Veneto	Soave Classico	
65365	Italy	Stunning and sophisticated, it leads with inte...	Sanct Valentin	94	40.0	Northeastern Italy	Alto Adige	

	country	description	designation	points	price	province	region_1	re
65399	Italy	Aesthetics and elegance are important values t...	Nectar Dei	91	65.0	Tuscany	Maremma	

3479 rows × 13 columns

Suppose we'll buy any wine that's made in Italy *or* which is rated above average. For this we use a pipe (`|`):

In [26]:
```
reviews[(reviews.country == 'Italy') | (reviews.points >= 90)]
```

Out[26]:

	country	description	designation	points	price	province	region_1	region_
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	Na
6	Italy	Here's a bright, informal red that opens with ...	Belsito	87	16.0	Sicily & Sardinia	Vittoria	Na
13	Italy	This is dominated by oak and oak-driven aromas...	Rosso	87	NaN	Sicily & Sardinia	Etna	Na
22	Italy	Delicate aromas recall white flower and citrus...	Ficiligno	87	19.0	Sicily & Sardinia	Sicilia	Na
24	Italy	Aromas of prune, blackcurrant, toast and oak c...	Aynat	87	35.0	Sicily & Sardinia	Sicilia	Na
...	...	...	...	...	...	...	...	
65494	France	Made from young vines from the Vaulorent porti...	Fourchaume Premier Cru	90	45.0	Burgundy	Chablis	Na
65495	Australia	This is a big, fat, almost sweet-tasting Caber...	NaN	90	22.0	South Australia	McLaren Vale	Na
65496	US	Much improved over the unripe 2005, Fritz's 20...	Estate	90	20.0	California	Dry Creek Valley	Sonom
65497	US	This wine wears its 15.8% alcohol better than ...	Block 24	90	31.0	California	Napa Valley	Nap

	country	description	designation	points	price	province	region_1	region_
65498	Spain	A unique take on Manzanilla Sherry, which is o...	Manzanilla	90	10.0	Andalucia	Jerez	Na

31430 rows × 13 columns

Pandas comes with a few built-in conditional selectors, two of which we will highlight here.

The first is `isin`. `isin` is lets you select data whose value "is in" a list of values. For example, here's how we can use it to select wines only from Italy or France:

In [27]:
```
reviews.loc[reviews.country.isin(['Italy', 'France'])]
```

Out[27]:

	country	description	designation	points	price	province	region_1	region_2
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN
6	Italy	Here's a bright, informal red that opens with ...	Belsito	87	16.0	Sicily & Sardinia	Vittoria	NaN
7	France	This dry and restrained wine offers spice in p...	NaN	87	24.0	Alsace	Alsace	NaN
9	France	This has great depth of flavor with its fresh ...	Les Natures	87	27.0	Alsace	Alsace	NaN
11	France	This is a dry wine, very spicy, with a tight, ...	NaN	87	30.0	Alsace	Alsace	NaN
...	...	...	...	...	...	...	...	...
65485	France	There's a fine balance here between minerality...	Montmains Premier Cru	90	40.0	Burgundy	Chablis	NaN
65486	France	Closed up and firm with a hint of vanilla, hon...	Domaine Long-Depaquit Les Bougnons Premier Cru	90	NaN	Burgundy	Chablis	NaN
65491	France	A big, toasty wine, full of ripe, delicious fr...	Fourchaume Vieilles Vignes Premier Cru	90	36.0	Burgundy	Chablis	NaN
65492	France	A rounded, fruity wine, packed with yellow pea...	Mont-de-Milieu Premier Cru	90	30.0	Burgundy	Chablis	NaN
[illegible]	[illegible]ance	Made from young vines	Fourchaume Premier Cru	90	45.0	Burgundy	Chablis	NaN

	country	description	designation	points	price	province	region_1	region_2
		from the Vaulorent porti...						

21179 rows × 13 columns

The second is `isnull` (and its companion `notnull`). These methods let you highlight values which are (or are not) empty (`NaN`). For example, to filter out wines lacking a price tag in the dataset, here's what we would do:

In [28]:
```
reviews.loc[reviews.price.isnull()]
```

Out[28]:

	country	description	designation	points	price	province	region_1	r
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	
13	Italy	This is dominated by oak and oak-driven aromas...	Rosso	87	NaN	Sicily & Sardinia	Etna	
30	France	Red cherry fruit comes laced with light tannin...	Nouveau	86	NaN	Beaujolais	Beaujolais-Villages	
31	Italy	Merlot and Nero d'Avola form the base for this...	Calanìca Nero d'Avola-Merlot	86	NaN	Sicily & Sardinia	Sicilia	
32	Italy	Part of the extended Calanìca series, this Gri...	Calanìca Grillo-Viognier	86	NaN	Sicily & Sardinia	Sicilia	
...	...	...	...	...	...	...	...	
65252	US	There's a bit of a green bean streak running t...	NaN	87	NaN	Washington	Washington	Wa
65281	Portugal	This wine is all about black and red berry fru...	Casa Américo	86	NaN	Dão	NaN	
65284	Chile	Minerally raw aromas brush up against harsh. T...	Tectonia	86	NaN	Leyda Valley	NaN	
65405	Austria	Intense, crisp acidity shines on this young wi...	Steinagrund Lagenreserve	91	NaN	Wagram-Donauland	NaN	

	country	description	designation	points	price	province	region_1	r
65486	France	Closed up and firm with a hint of vanilla, hon...	Domaine Long-Depaquit Les Bougnons Premier Cru	90	NaN	Burgundy	Chablis	

4670 rows × 13 columns

In [29]:
```
reviews.loc[reviews.price.notnull()]
```

Out[29]:

	country	description	designation	points	price	province	region_1	regior
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	N
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willame Va
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	N
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willame Va
5	Spain	Blackberry and raspberry aromas show a typical...	Ars In Vitro	87	15.0	Northern Spain	Navarra	N
...	...	...	...	...	...	...	...	
65494	France	Made from young vines from the Vaulorent porti...	Fourchaume Premier Cru	90	45.0	Burgundy	Chablis	N
65495	Australia	This is a big, fat, almost sweet-tasting Caber...	NaN	90	22.0	South Australia	McLaren Vale	N
65496	US	Much improved over the unripe 2005, Fritz's 20...	Estate	90	20.0	California	Dry Creek Valley	Sonc
65497	US	This wine wears its 15.8% alcohol	Block 24	90	31.0	California	Napa Valley	Na

	country	description	designation	points	price	province	region_1	regior
		better than ...						
65498	Spain	A unique take on Manzanilla Sherry, which is o...	Manzanilla	90	10.0	Andalucia	Jerez	N

60829 rows × 13 columns

Assigning data

Going the other way, assigning data to a DataFrame is easy. You can assign either a constant value:

In [30]:
```
reviews['critic'] = 'everyone'
reviews
```

Out[30]:

	country	description	designation	points	price	province	region_1	regior
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	N
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	N
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willame Va
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	N
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willame Va
...	...	...	...	...	...	...	...	
65494	France	Made from young vines from the Vaulorent porti...	Fourchaume Premier Cru	90	45.0	Burgundy	Chablis	N
65495	Australia	This is a big, fat, almost sweet-tasting Caber...	NaN	90	22.0	South Australia	McLaren Vale	N
65496	US	Much improved over the unripe 2005, Fritz's 20...	Estate	90	20.0	California	Dry Creek Valley	Sonc
65497	US	This wine wears its 15.8% alcohol	Block 24	90	31.0	California	Napa Valley	Na

	country	description	designation	points	price	province	region_1	regior
		better than ...						
65498	Spain	A unique take on Manzanilla Sherry, which is o...	Manzanilla	90	10.0	Andalucia	Jerez	N

65499 rows × 14 columns

Or with an iterable of values:

In [31]:
```
reviews['index_backwards'] = range(len(reviews), 0, -1)
reviews
```

Out[31]:

	country	description	designation	points	price	province	region_1	regior
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	N
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	N
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willame Va
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	N
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willame Va
...	...	...	...	...	...	...	...	
65494	France	Made from young vines from the Vaulorent porti...	Fourchaume Premier Cru	90	45.0	Burgundy	Chablis	N
65495	Australia	This is a big, fat, almost sweet-tasting Caber...	NaN	90	22.0	South Australia	McLaren Vale	N
65496	US	Much improved over the unripe 2005, Fritz's 20...	Estate	90	20.0	California	Dry Creek Valley	Sonc
65497	US	This wine wears its 15.8% alcohol	Block 24	90	31.0	California	Napa Valley	Na

	country	description	designation	points	price	province	region_1	regior
		better than ...						
65498	Spain	A unique take on Manzanilla Sherry, which is o...	Manzanilla	90	10.0	Andalucia	Jerez	N

65499 rows × 15 columns

Your turn

If you haven't started the exercise, you can start now.

การจัดการข้อมูลด้วย Pandas ใน Python

การปรับแต่งและแปลงข้อมูล

สรุปบทเรียนที่แล้ว

- เรียนรู้วิธีการเลือกข้อมูลจาก DataFrame และ Series
- การเข้าถึงข้อมูลด้วยวิธีต่าง ๆ
- ความสำคัญของการเลือกข้อมูลที่ถูกต้องเพื่อการวิเคราะห์

วัตถุประสงค์

- เรียนรู้การใช้ฟังก์ชันสรุปข้อมูล (Summary Functions)
- เข้าใจการใช้ฟังก์ชันแบบไม่ระบุชื่อ (Anonymous Functions)
- เรียนรู้การแปลงข้อมูลด้วย map() และ apply()
- ฝึกปฏิบัติการแปลงข้อมูลในรูปแบบต่าง ๆ

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

reviews = pd.read_csv("datasets/winemag-data-130k-v2.csv", index_col=0)
```

In [2]:
```
reviews
```

Out[2]:

	country	description	designation	points	price	province	region_1	regior
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	N
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	N
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willame Va
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	N
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willame Va
...	...	...	...	...	...	...	...	
65494	France	Made from young vines from the Vaulorent porti...	Fourchaume Premier Cru	90	45.0	Burgundy	Chablis	N
65495	Australia	This is a big, fat, almost sweet-tasting Caber...	NaN	90	22.0	South Australia	McLaren Vale	N
65496	US	Much improved over the unripe 2005, Fritz's 20...	Estate	90	20.0	California	Dry Creek Valley	Sonc
65497	US	This wine wears its 15.8% alcohol	Block 24	90	31.0	California	Napa Valley	Na

	country	description	designation	points	price	province	region_1	regior
		better than ...						
65498	Spain	A unique take on Manzanilla Sherry, which is o...	Manzanilla	90	10.0	Andalucia	Jerez	N

65499 rows × 13 columns

ฟังก์ชันสรุปข้อมูล (Summary Functions)

- ฟังก์ชันที่ช่วยปรับโครงสร้างข้อมูลให้อยู่ในรูปแบบที่เป็นประโยชน์
- ใช้เพื่อวิเคราะห์และทำความเข้าใจข้อมูลอย่างรวดเร็ว
- For example, consider the describe() method:

In [3]:
```
reviews.describe()
```

Out[3]:

	points	price
count	65499.000000	60829.000000
mean	88.434037	35.232932
std	3.030310	39.477858
min	80.000000	4.000000
25%	86.000000	17.000000
50%	88.000000	25.000000
75%	91.000000	42.000000
max	100.000000	2500.000000

In [4]:
```
# สำหรับข้อมูลตัวเลข
reviews.points.describe()
```

Out[4]:
```
count    65499.000000
mean        88.434037
std          3.030310
min         80.000000
25%         86.000000
50%         88.000000
75%         91.000000
max        100.000000
Name: points, dtype: float64
```

In [5]:
```
# การใช้ describe() กับหลายคอลัมน์สำหรับข้อมูลตัวเลขพร้อมกัน
reviews[['points', 'price']].describe()
```

Out[5]:

	points	price
count	65499.000000	60829.000000
mean	88.434037	35.232932
std	3.030310	39.477858
min	80.000000	4.000000
25%	86.000000	17.000000
50%	88.000000	25.000000
75%	91.000000	42.000000
max	100.000000	2500.000000

In [6]:
```
# สำหรับข้อมูลประเภทข้อความ
reviews.taster_name.describe()
```

Out[6]:
```
count          51856
unique            19
top       Roger Voss
freq           13045
Name: taster_name, dtype: object
```

In [7]:
```
# การใช้ describe() กับหลายคอลัมน์สำหรับข้อมูลประเภทข้อความพร้อมกัน
reviews[['country', 'taster_name']].describe()
```

Out[7]:

	country	taster_name
count	65467	51856
unique	41	19
top	US	Roger Voss
freq	27177	13045

In [8]:
```
# การใช้ describe() กับหลายคอลัมน์สำหรับข้อมูลผสมพร้อมกัน
reviews[['country','taster_name','points']].describe()
```

Out[8]:

	points
count	65499.000000
mean	88.434037
std	3.030310
min	80.000000
25%	86.000000
50%	88.000000
75%	91.000000
max	100.000000

ถ้าเราต้องการข้อมูลสถิติสรุปแบบง่ายๆ เกี่ยวกับคอลัมน์ใน DataFrame หรือ Series โดยทั่วไปจะมีฟังก์ชัน(method) pandas ที่เป็นประโยชน์ซึ่งจะช่วยให้ทำได้ ตัวอย่างเช่น

In [9]:
```
# หาค่าเฉลี่ยใช้ mean()
reviews.points.mean()
```

Out[9]:
```
88.43403716087269
```

In [10]:
```
# ดูค่าที่ไม่ซ้ำใช้ unique()
reviews.taster_name.unique()
```

Out[10]:
```
array(['Kerin O’Keefe', 'Roger Voss', 'Paul Gregutt',
       'Alexander Peartree', 'Michael Schachner', 'Anna Lee C. Iijima',
       'Virginie Boone', 'Matt Kettmann', nan, 'Sean P. Sullivan',
       'Jim Gordon', 'Joe Czerwinski', 'Anne Krebiehl\xa0MW',
       'Lauren Buzzeo', 'Mike DeSimone', 'Jeff Jenssen',
       'Susan Kostrzewa', 'Carrie Dykes', 'Fiona Adams',
       'Christina Pickard'], dtype=object)
```

In [11]:
```
# นับจำนวนค่าที่ซ้ำกันใช้ value_counts()
reviews.taster_name.value_counts()
```

Out[11]:
```
taster_name
Roger Voss            13045
Michael Schachner      7752
Kerin O'Keefe          5313
Paul Gregutt           4851
Virginie Boone         4696
Matt Kettmann          3035
Joe Czerwinski         2605
Sean P. Sullivan       2358
Anna Lee C. Iijima     2134
Jim Gordon             2032
Anne Krebiehl MW       1769
Lauren Buzzeo           938
Susan Kostrzewa         593
Jeff Jenssen            234
Mike DeSimone           231
Alexander Peartree      210
Carrie Dykes             45
Fiona Adams              11
Christina Pickard         4
Name: count, dtype: int64
```

In [12]:
```
reviews.points.value_counts()
```

Out[12]:
```
points
87     8872
88     8423
90     7697
86     6179
91     6016
89     5724
85     5082
92     4917
84     3490
93     3268
94     1905
83     1442
82      923
95      678
81      305
96      262
80      155
97       99
98       39
99       15
100       8
Name: count, dtype: int64
```

ฟังก์ชันแบบไม่ระบุชื่อ (Anonymous Functions)

- เรียกอีกอย่างว่า Lambda Function
- เป็นฟังก์ชันที่ไม่จำเป็นต้องมีชื่อ
- [illegible] ในบรรทัดเดียว

- เหมาะสำหรับฟังก์ชันง่าย ๆ ที่ใช้เพียงครั้งเดียว

In [13]:
```
# ฟังก์ชันแบบปกติ
def f(x):
    return x**2 + x - 1
```

In [14]:
```
# ฟังก์ชันแบบ lambda
g = lambda x: x**2 + x - 1
```

In [15]:
```
# ทั้งสองฟังก์ชันให้ผลลัพธ์เหมือนกัน
x = 10
print('f(x) =',f(x))
print('g(x) =',g(x))
```

```
f(x) = 109
g(x) = 109
```

In [16]:
```
# ฟังก์ชัน lambda ที่รับหลายตัวแปร
h = lambda x, y, z: x**2 + y**2 + z**2
```

In [17]:
```
# ทดสอบการใช้งาน
x, y, z = 0, 1, 1
value = h(x, y, z)
print(value)
```

```
2
```

Map คืออะไร?

- Map คือการแปลงค่าจากชุดข้อมูลหนึ่งไปเป็นอีกชุดข้อมูลหนึ่ง
- ใช้ในการสร้างข้อมูลรูปแบบใหม่จากข้อมูลที่มีอยู่
- ใช้ในการแปลงข้อมูลจากรูปแบบหนึ่งไปเป็นอีกรูปแบบหนึ่ง
- Pandas มีวิธีการ Map สองแบบหลัก ๆ: map() และ apply()

การใช้ map()

- ใช้กับ Series (คอลัมน์เดียว)
- ส่งผ่านค่าแต่ละค่าในคอลัมน์ไปยังฟังก์ชัน
- คืนค่าเป็น Series ใหม่

In [18]:
```
review_points_mean = reviews.points.mean()
reviews.points.map(lambda p: p - review_points_mean)
```

Out[18]:
```
0       -1.434037
1       -1.434037
2       -1.434037
3       -1.434037
4       -1.434037
           ...
65494    1.565963
65495    1.565963
65496    1.565963
65497    1.565963
65498    1.565963
Name: points, Length: 65499, dtype: float64
```

การใช้ apply()

- ใช้กับ DataFrame ทั้งหมด
- สามารถแปลงข้อมูลทั้งแถวหรือทั้งคอลัมน์
- สามารถทำงานที่ซับซ้อนกว่า map()

In [19]:
```
def remean_points(row):
    row['points'] = row['points'] - review_points_mean
    return row
```

In [20]:
```
reviews.apply(remean_points, axis='columns')
```

Out[20]:

	country	description	designation	points	price	province	region_1	re
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	-1.434037	NaN	Sicily & Sardinia	Etna	
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	-1.434037	15.0	Douro	NaN	
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	-1.434037	14.0	Oregon	Willamette Valley	Will
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	-1.434037	13.0	Michigan	Lake Michigan Shore	
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	-1.434037	65.0	Oregon	Willamette Valley	Will
...	...	...	...	...	...	...	...	
65494	France	Made from young vines from the Vaulorent porti...	Fourchaume Premier Cru	1.565963	45.0	Burgundy	Chablis	
65495	Australia	This is a big, fat, almost sweet-tasting Caber...	NaN	1.565963	22.0	South Australia	McLaren Vale	
65496	US	Much improved over the unripe 2005, Fritz's 20...	Estate	1.565963	20.0	California	Dry Creek Valley	S
65497	US	This wine wears its 15.8% alcohol	Block 24	1.565963	31.0	California	Napa Valley	

	country	description	designation	points	price	province	region_1	re
		better than ...						
65498	Spain	A unique take on Manzanilla Sherry, which is o...	Manzanilla	1.565963	10.0	Andalucia	Jerez	

65499 rows × 13 columns

ความแตกต่างของ axis

- `axis='columns'` หรือ `axis=1` : ทำงานกับแถว
- `axis='index'` หรือ `axis=0` : ทำงานกับคอลัมน์

ข้อสังเกต

map() และ apply() จะ return ค่า Series และ DataFrames ที่ถูกแปลงใหม่ตามลำดับ โดยจะไม่แก้ไขข้อมูลเดิมที่เรียกใช้ หากเราดูที่แถวแรกของreviews เราจะเห็นว่ามันยังคงมีค่าคะแนนเดิมอยู่

In [21]:
```
reviews.head(1)
```

Out[21]:

	country	description	designation	points	price	province	region_1	region_2	tast
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	

การใช้ Operator เพื่อความเร็วในการคำนวณ

- Pandas มีการดำเนินการแมป (mapping) ที่ใช้บ่อยหลายอย่างเป็นฟังก์ชัน built-ins ในตัว
- Pandas เข้าใจการทำงานระหว่าง Series กับ single value
- มีความเร็วสูงกว่าการใช้ map() หรือ apply()

In [22]:
```
review_points_mean = reviews.points.mean()
reviews.points - review_points_mean
```

Out[22]:
```
0       -1.434037
1       -1.434037
2       -1.434037
3       -1.434037
4       -1.434037
           ...
65494    1.565963
65495    1.565963
65496    1.565963
65497    1.565963
65498    1.565963
Name: points, Length: 65499, dtype: float64
```

จากคำสั่ง Code ด้านบนนี้ เราจะดำเนินการระหว่างค่าต่างๆ มากมายทางด้านซ้ายมือ (ทุกค่าในซีรีส์) และค่าเดียวทางด้านขวามือ (ค่าเฉลี่ย) Pandas จะพิจารณานิพจน์นี้และคำนวณว่าเราต้องลบค่าเฉลี่ยออกจากค่าทุกค่าในชุดข้อมูล

นอกจากนี้ Pandas ยังเข้าใจด้วยว่าต้องทำอย่างไรหากเราดำเนินการเหล่านี้ระหว่างซีรีส์ที่มีความยาวเท่ากัน ตัวอย่างเช่น วิธีง่ายๆ ในการรวมข้อมูลประเทศและภูมิภาคในชุดข้อมูลคือทำดังต่อไปนี้:

In [23]:
```
# การผสมข้อมูลด้วย Operator : สามารถใช้กับ Series ที่มีความยาวเท่ากัน
# การรวมข้อมูลประเทศกับภูมิภาค
reviews.country + " - " + reviews.region_1
```

Out[23]:
```
0                   Italy - Etna
1                            NaN
2          US - Willamette Valley
3        US - Lake Michigan Shore
4          US - Willamette Valley
                   ...
65494             France - Chablis
65495    Australia - McLaren Vale
65496       US - Dry Creek Valley
65497             US - Napa Valley
65498                Spain - Jerez
Length: 65499, dtype: object
```

ความแตกต่างระหว่าง Operator และ map()/apply()

- Operator: เร็วกว่า แต่ใช้ได้กับการคำนวณพื้นฐาน
- map()/apply(): ยืดหยุ่นกว่า สามารถใช้กับโลจิกที่ซับซ้อน

Your turn

If you haven't started the exercise, you can now.

In [ ]:
```
```

Introduction

Maps allow us to transform data in a DataFrame or Series one value at a time for an entire column. However, often we want to group our data, and then do something specific to the group the data is in.

As you'll learn, we do this with the `groupby()` operation. We'll also cover some additional topics, such as more complex ways to index your DataFrames, along with how to sort your data.

Understanding Groupby¶

ฟังก์ชัน groupby() ใน Pandas จะแบ่งข้อมูลทั้งหมดจากชุดข้อมูลออกเป็นหมวดหมู่หรือกลุ่มต่างๆ ทำให้สามารถ วิเคราะห์ข้อมูล ตามกลุ่มต่างๆ ได้อย่างยืดหยุ่น

Here's a super simple dataframe to illustrate some examples. We'll be grouping the data by the "animal" column where there are four categories of animals:

- alligators
- cats
- snakes
- hamsters

In [1]:
```
import numpy as np
import pandas as pd
import random

# Random pets column
pet_list = ["cat", "hamster", "alligator", "snake"]
pet = [random.choice(pet_list) for i in range(1,15)]

# Random weight of animal column
weight = [random.choice(range(5,15)) for i in range(1,15)]

# Random length of animals column
length = [random.choice(range(1,10)) for i in range(1,15)]

# random age of the animals column
age = [random.choice(range(1,15)) for i in range(1,15)]

# Put everyhting into a dataframe
df = pd.DataFrame()
df["animal"] = pet
df["age"] = age
df["weight"] = weight
df["length"] = length
```

```
# make a groupby object
animal_groups = df.groupby("animal")
```

In [2]:
```
df
```

Out[2]:

	animal	age	weight	length
0	alligator	1	11	2
1	alligator	12	13	6
2	cat	6	14	7
3	alligator	2	9	7
4	alligator	13	9	1
5	cat	14	14	5
6	hamster	14	5	1
7	hamster	3	13	9
8	snake	3	14	7
9	alligator	9	9	1
10	hamster	13	14	1
11	snake	6	14	8
12	cat	13	13	9
13	alligator	7	7	3

- เราสามารถถามเกี่ยวกับข้อมูลสัตว์ได้ก็คือ
- หากต้องการหาค่าเฉลี่ย(**mean**)ของน้ำหนักของสัตว์แต่ละประเภท เราจะจัดกลุ่มสัตว์ตามประเภทของสัตว์ จากนั้นจึงใช้ฟังก์ชันค่าเฉลี่ย
- เราสามารถใช้ฟังก์ชันอื่นๆ ได้เช่นกัน
- เราสามารถใช้ "**sum**" เพื่อหาผลรวมน้ำหนักทั้งหมด
- "**min**" เพื่อค้นหาน้ำหนักต่ำสุด
- "**max**" เพื่อค้นหาน้ำหนักสูงสุด
- หรือ "**count**" เพื่อค้นหาจำนวนสัตว์แต่ละประเภท

Summary statistics	Numpy operations	More complex operations
mean	np.mean	.agg()
median	np.min	agg(["mean", "median"])
min	np.max	agg(custom_function())
max	np.sum	
sum	np.product	
describe		
count or size		

These two lines of code group the animals then apply the mean function to the weight column.

In [3]:
```
# Group by animal category
animal_groups = df.groupby("animal")
# Apply mean function to wieght column
animal_groups['weight'].mean()
```

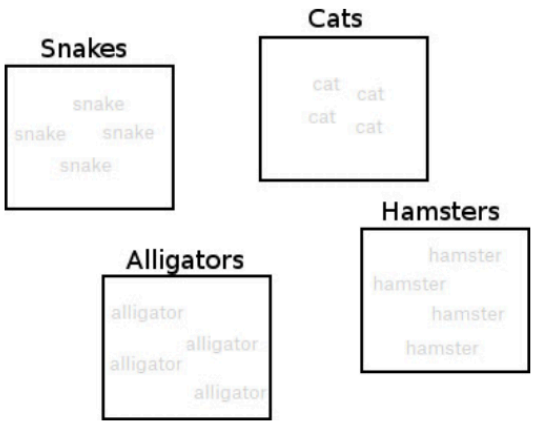
Out[3]:
```
animal
alligator     9.666667
cat          13.666667
hamster      10.666667
snake        14.000000
Name: weight, dtype: float64
```

Here's what happens when you run that code:

1. Group the unique values from the animal column

	animal	age	weight	length
0	snake	4	13	4
1	alligator	3	7	8
2	alligator	8	11	7
3	alligator	10	9	6
4	hamster	7	8	7
5	snake	11	6	7
6	hamster	14	13	2
7	hamster	13	13	5
8	hamster	2	11	9
9	snake	12	13	2
10	cat	8	11	5
11	cat	4	12	7
12	cat	3	11	9
13	alligator	7	9	2

2. Now there's a bucket for each group

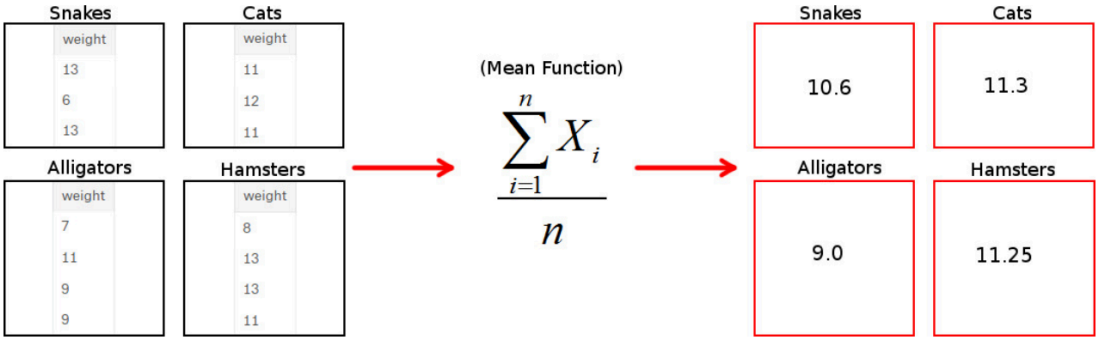


Loading [MathJax]/extensions/Safe.js

3. Toss the other data into the buckets

	animal	age	weight	length
0	snake	4	13	4
1	alligator	3	7	8
2	alligator	8	11	7
3	alligator	10	9	6
4	hamster	7	8	7
5	snake	11	6	7
6	hamster	14	13	2
7	hamster	13	13	5
8	hamster	2	11	9
9	snake	12	13	2
10	cat	8	11	5
11	cat	4	12	7
12	cat	3	11	9
13	alligator	7	9	2

4. Apply a function on the weight column of each bucket



In [4]:
```
df.groupby("animal").size()
```

Out[4]:
```
animal
alligator    6
cat          3
hamster      3
snake        2
dtype: int64
```

In [5]:
```
df.groupby("animal").count()
```

Out[5]:

animal	age	weight	length
alligator	6	6	6
cat	3	3	3
hamster	3	3	3
snake	2	2	2

In [6]:
```
df.groupby("animal")["weight"].count()
```

Out[6]:
```
animal
alligator    6
cat          3
hamster      3
snake        2
Name: weight, dtype: int64
```

In [7]:
```
df.groupby("weight")["animal"].count()
```

Out[7]:
```
weight
5     1
7     1
9     3
11    1
13    3
14    5
Name: animal, dtype: int64
```

In [8]:
```
df.groupby("weight")["animal"].max()
```

Out[8]:
```
weight
5       hamster
7     alligator
9     alligator
11    alligator
13      hamster
14        snake
Name: animal, dtype: object
```

Next Example

In [9]:
```
import pandas as pd

reviews = pd.read_csv("datasets/winemag-data-52.csv", index_col=0)
reviews
```

Out[9]:

	country	description	designation	points	price	province	region_1	reg
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	90	NaN	Sicily & Sardinia	Etna	
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willa
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	90	13.0	Michigan	Lake Michigan Shore	
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	95	65.0	Oregon	Willamette Valley	Willa
5	Spain	Blackberry and raspberry aromas show a typical...	Ars In Vitro	87	15.0	Northern Spain	Navarra	
6	Italy	Here's a bright, informal red that opens with ...	Belsito	92	16.0	Sicily & Sardinia	Vittoria	
7	France	This dry and restrained wine offers spice in p...	NaN	90	24.0	Alsace	Alsace	
8	Germany	Savory dried thyme notes accent sunnier flavor...	Shine	87	12.0	Rheinhessen	NaN	
9	France	This has great depth of flavor with its fresh ...	Les Natures	92	27.0	Alsace	Alsace	
10	US	Soft, supple plum envelopes an	Mountain Cuvée	87	19.0	California	Napa Valley	

	country	description	designation	points	price	province	region_1	reg
		oaky structure ...						
11	France	This is a dry wine, very spicy, with a tight, ...	NaN	87	30.0	Alsace	Alsace	
12	US	Slightly reduced, this wine offers a chalky, t...	NaN	94	34.0	California	Alexander Valley	S
13	Italy	This is dominated by oak and oak-driven aromas...	Rosso	95	NaN	Sicily & Sardinia	Etna	
14	US	Building on 150 years and six generations of w...	NaN	92	12.0	California	Central Coast	C
15	Germany	Zesty orange peels and apple notes abound in t...	Devon	87	24.0	Mosel	NaN	
16	Argentina	Baked plum, molasses, balsamic vinegar and che...	Felix	87	30.0	Other	Cafayate	
17	Argentina	Raw black-cherry aromas are direct and simple ...	Winemaker Selection	87	13.0	Mendoza Province	Mendoza	
18	Spain	Desiccated blackberry, leather, charred wood a...	Vendimia Seleccionada Finca Valdelayegua Singl...	87	28.0	Northern Spain	Ribera del Duero	
19	US	Red fruit aromas pervade on the nose, with cig...	NaN	90	32.0	Virginia	Virginia	
20	US	Ripe aromas of dark berries mingle with ample ...	Vin de Maison	87	23.0	Virginia	Virginia	

	country	description	designation	points	price	province	region_1	reg
21	US	A sleek mix of tart berry, stem and herb, alon...	NaN	92	20.0	Oregon	Oregon	C
22	Italy	Delicate aromas recall white flower and citrus...	Ficiligno	92	19.0	Sicily & Sardinia	Sicilia	
23	US	This wine from the Geneseo district offers aro...	Signature Selection	94	22.0	California	Paso Robles	C
24	Italy	Aromas of prune, blackcurrant, toast and oak c...	Aynat	90	35.0	Sicily & Sardinia	Sicilia	
25	US	Oak and earth intermingle around robust aromas...	King Ridge Vineyard	95	69.0	California	Sonoma Coast	S
26	Italy	Pretty aromas of yellow flower and stone fruit...	Dalila	95	13.0	Sicily & Sardinia	Terre Siciliane	
27	Italy	Aromas recall ripe dark berry, toast and a whi...	NaN	87	10.0	Sicily & Sardinia	Terre Siciliane	
28	Italy	Aromas suggest mature berry, scorched earth, a...	Mascaria Barricato	90	17.0	Sicily & Sardinia	Cerasuolo di Vittoria	
29	US	Clarksburg is becoming a haven for Chenin Blan...	NaN	86	16.0	California	Clarksburg	C
30	France	Red cherry fruit comes laced with light tannin...	Nouveau	86	NaN	Beaujolais	Beaujolais-Villages	

	country	description	designation	points	price	province	region_1	reg
31	Italy	Merlot and Nero d'Avola form the base for this...	Calanìca Nero d'Avola-Merlot	86	NaN	Sicily & Sardinia	Sicilia	
32	Italy	Part of the extended Calanìca series, this Gri...	Calanìca Grillo-Viognier	90	NaN	Sicily & Sardinia	Sicilia	
33	US	Rustic and dry, this has flavors of berries, c...	Puma Springs Vineyard	86	50.0	California	Dry Creek Valley	S
34	US	This shows a tart, green gooseberry flavor tha...	NaN	94	20.0	California	Sonoma Valley	S
35	US	As with many of the Erath 2010 vineyard design...	Hyland	95	50.0	Oregon	McMinnville	Willa
36	Chile	White flower, lychee and apple aromas carry th...	Estate	86	15.0	Colchagua Valley	NaN	
37	Italy	This concentrated Cabernet offers aromas of cu...	Missoni	95	21.0	Sicily & Sardinia	Sicilia	
38	Italy	Inky in color, this wine has plump aromas of r...	I Tratturi	92	11.0	Southern Italy	Puglia	
39	Italy	Part of the natural wine movement, this wine i...	Purato Made With Organic Grapes	86	12.0	Sicily & Sardinia	Sicilia	
40	Italy	Catarratto is one of Sicily's most widely farm...	NaN	86	17.0	Sicily & Sardinia	Sicilia	
41	US	A stiff, tannic wine, this	NaN	92	22.0	Oregon	Willamette Valley	Willa

	country	description	designation	points	price	province	region_1	reg
		slowly opens and br...						
42	France	This is a festive wine, with soft, ripe fruit ...	Nouveau	86	9.0	Beaujolais	Beaujolais	
43	US	The clean, brisk mouthfeel gives this slightly...	NaN	86	14.0	California	Paso Robles	C
44	Chile	A berry aroma comes with cola and herb notes. ...	NaN	86	9.0	Maule Valley	NaN	
45	US	Right out of the starting blocks this is an oa...	#SocialSecret	90	40.0	Virginia	Virginia	
46	Italy	Spicy, fresh and clean, this would pair with f...	Sallier de la Tour	92	13.0	Sicily & Sardinia	Sicilia	
47	US	This is a sweet wine with flavors of white sug...	NaN	90	13.0	California	Lake County	
48	US	This bottling resembles the New Zealand paradi...	NaN	95	16.0	Virginia	Monticello	
49	France	Soft and fruity, this is a generous, ripe wine...	Eté Indien	86	14.0	Beaujolais	Brouilly	
50	Italy	This blend of Nero d'Avola and Syrah opens wit...	Scialo	86	NaN	Sicily & Sardinia	Sicilia	
51	Chile	This is much different than Casa Silva's 2009 ...	Gran Reserva	85	22.0	Colchagua Valley	NaN	

In [10]:
```
#เช็คว่า DataFrame มีคอลัมน์อะไรบ้าง
reviews.columns
```

Out[10]:
```
Index(['country', 'description', 'designation', 'points', 'price', 'provinc
e',
       'region_1', 'region_2', 'taster_name', 'taster_twitter_handle', 'tit
le',
       'variety', 'winery'],
      dtype='object')
```

In [11]:
```
#แสดงผลทางสถิติโดยยึด country เป็นหลัก
reviews.groupby('country').describe()
```

Out[11]:

	points									
	count	mean	std	min	25%	50%	75%	max	count	m
country										
Argentina	2.0	87.000000	0.000000	87.0	87.00	87.0	87.00	87.0	2.0	21.500
Chile	3.0	85.666667	0.577350	85.0	85.50	86.0	86.00	86.0	3.0	15.333
France	6.0	87.833333	2.562551	86.0	86.00	86.5	89.25	92.0	5.0	20.800
Germany	2.0	87.000000	0.000000	87.0	87.00	87.0	87.00	87.0	2.0	18.000
Italy	16.0	90.250000	3.296463	86.0	86.75	90.0	92.00	95.0	11.0	16.727
Portugal	1.0	87.000000	NaN	87.0	87.00	87.0	87.00	87.0	1.0	15.000
Spain	2.0	87.000000	0.000000	87.0	87.00	87.0	87.00	87.0	2.0	21.500
US	20.0	90.850000	3.407036	86.0	87.00	91.0	94.00	95.0	20.0	28.200

In [12]:
```
reviews.groupby('country').size()
```

Out[12]:
```
country
Argentina     2
Chile         3
France        6
Germany       2
Italy        16
Portugal      1
Spain         2
US           20
dtype: int64
```

In [13]:
```
#นับจำนวนทุกคอลัมน์โดยยึด country เป็นหลัก
reviews.groupby('country').count()
```

Out[13]:

country	description	designation	points	price	province	region_1	region_2	taste
Argentina	2	2	2	2	2	2	0	
Chile	3	2	3	3	3	0	0	
France	6	4	6	5	6	6	0	
Germany	2	2	2	2	2	0	0	
Italy	16	14	16	11	16	16	0	
Portugal	1	1	1	1	1	0	0	
Spain	2	2	2	2	2	2	0	
US	20	9	20	20	20	20	14	

In [14]:
```
reviews.groupby('points').size()
```

Out[14]:
```
points
85     1
86    12
87    12
90     9
92     8
94     3
95     7
dtype: int64
```

In [15]:
```
#นับจำนวนทุกคอลัมน์โดยยึด points เป็นหลัก
reviews.groupby('points').count()
```

Out[15]:

points	country	description	designation	price	province	region_1	region_2	taster_
85	1	1	1	1	1	0	0	
86	12	12	8	9	12	10	3	
87	12	12	9	12	12	9	2	
90	9	9	6	7	9	9	0	
92	8	8	5	8	8	8	3	
94	3	3	1	3	3	3	3	
95	7	7	6	6	7	7	3	

In [16]:
```
#นับจำนวนคอลัมน์ price โดยยึด points เป็นหลัก
reviews.groupby('points').price.count()
```

Out[16]:
```
points
85     1
86     9
87    12
90     7
92     8
94     3
95     6
Name: price, dtype: int64
```

In [17]:
```
#นับจำนวนคอลัมน์ country โดยยึด points เป็นหลัก
reviews.groupby('points').country.count()
```

Out[17]:
```
points
85     1
86    12
87    12
90     9
92     8
94     3
95     7
Name: country, dtype: int64
```

In [18]:
```
#นับจำนวนคอลัมน์ country แยกแต่ละ value(ประเทศ) โดยยึด points เป็นหลัก
reviews.groupby('points').country.value_counts()
```

Out[18]:
```
points  country
85      Chile        1
86      Italy        4
        France       3
        US           3
        Chile        2
87      US           3
        Spain        2
        Germany      2
        Argentina    2
        Portugal     1
        Italy        1
        France       1
90      US           4
        Italy        4
        France       1
92      Italy        4
        US           3
        France       1
94      US           3
95      US           4
        Italy        3
Name: count, dtype: int64
```

In [19]:
```
#นับจำนวนคอลัมน์ price แยกแต่ละ value(ราคา) โดยยึด points เป็นหลัก
reviews.groupby('points').price.value_counts()
```

Out[19]:

points	price	
85	22.0	1
86	14.0	2
	9.0	2
	12.0	1
	15.0	1
	16.0	1
	17.0	1
	50.0	1
87	15.0	2
	30.0	2
	12.0	1
	10.0	1
	13.0	1
	14.0	1
	19.0	1
	23.0	1
	24.0	1
	28.0	1
90	13.0	2
	40.0	1
	24.0	1
	35.0	1
	32.0	1
	17.0	1
92	12.0	1
	13.0	1
	16.0	1
	19.0	1
	20.0	1
	22.0	1
	27.0	1
	11.0	1
94	20.0	1
	22.0	1
	34.0	1
95	13.0	1
	16.0	1
	21.0	1
	50.0	1
	65.0	1
	69.0	1

Name: count, dtype: int64

`groupby()` created a group of reviews which allotted the same point values to the given wines. Then, for each of these groups, we grabbed the `points()` column and counted how many times it appeared. `value_counts()` is just a shortcut to this `groupby()` operation.

We can use any of the summary functions we've used before with this data. For example, to get the cheapest wine in each point value category, we can do the following:

In [20]:
```
#หาราคาถูกที่สุดในคอลัมน์ price โดยยึด points เป็นหลัก
reviews.groupby('points').price.min()
```

Out[20]:
```
points
85    22.0
86     9.0
87    10.0
90    13.0
92    11.0
94    20.0
95    13.0
Name: price, dtype: float64
```

You can think of each group we generate as being a slice of our DataFrame containing only data with values that match. This DataFrame is accessible to us directly using the `apply()` method, and we can then manipulate the data in any way we see fit. For example, here's one way of selecting the name of the first wine reviewed from each winery in the dataset:

การใช้ apply() กับ groupby()

เราสามารถใช้ apply() เพื่อทำงานกับข้อมูลในแต่ละกลุ่มได้อย่างยืดหยุ่น

In [21]:
```
#แสดงผลชื่อแรกในคอลัมน์ title โดยยึด points เป็นหลัก
reviews.groupby('points').apply(lambda df: df.title.iloc[0])
```

Out[21]:
```
points
85    Casa Silva 2008 Gran Reserva Petit Verdot (Col...
86    Clarksburg Wine Company 2010 Chenin Blanc (Cla...
87        Quinta dos Avidagos 2011 Avidagos Red (Douro)
90                    Nicosia 2013 Vulkà Bianco  (Etna)
92     Terre di Giurfo 2013 Belsito Frappato (Vittoria)
94    Louis M. Martini 2012 Cabernet Sauvignon (Alex...
95    Sweet Cheeks 2012 Vintner's Reserve Wild Child...
dtype: object
```

In [22]:
```
#แสดงผลชื่อแรกในคอลัมน์ title โดยยึด winery เป็นหลัก(เลือกชื่อไวน์รายการแรกที่รีวิวจากแต่ละโ
reviews.groupby('winery').apply(lambda df: df.title.iloc[0])
```

Out[22]:

winery	
Acrobat (Oregon)	Acrobat 2013 Pinot Noir
Baglio di Pianetto (Sicilia)	Baglio di Pianetto 2007 Ficiligno White
Bianchi (Paso ...	Bianchi 2011 Signature Selection Merlot
Canicattì (Sicilia)	Canicattì 2009 Aynat Nero d'Avola
Casa Silva t (Col...	Casa Silva 2008 Gran Reserva Petit Verdo
Castello di Amorosa yard P...	Castello di Amorosa 2011 King Ridge Vine
Clarksburg Wine Company c (Cla...	Clarksburg Wine Company 2010 Chenin Blan
Domaine de la Madone jolais...	Domaine de la Madone 2012 Nouveau (Beau
Duca di Salaparuta d'Avola-...	Duca di Salaparuta 2010 Calanìca Nero
Envolve (Dry Cr...	Envolve 2010 Puma Springs Vineyard Red
Erath innville)	Erath 2010 Hyland Pinot Noir (McM
Estampa (Colch...	Estampa 2011 Estate Viognier-Chardonnay
Felix Lavaque (Cafayate)	Felix Lavaque 2010 Felix Malbec
Feudi del Pisciotto t Sauv...	Feudi del Pisciotto 2010 Missoni Caberne
Feudi di San Marzano mitivo...	Feudi di San Marzano 2011 I Tratturi Pri
Feudo Montoni (Sicilia)	Feudo Montoni 2011 Catarratto
Feudo di Santa Tresa th Org...	Feudo di Santa Tresa 2011 Purato Made Wi
Gaucho Andino albec ...	Gaucho Andino 2011 Winemaker Selection M
Hawkins Cellars tte Va...	Hawkins Cellars 2009 Pinot Noir (Willame
Heinz Eifel heinhe...	Heinz Eifel 2013 Shine Gewürztraminer (R
Henry Fessy aujolais)	Henry Fessy 2012 Nouveau (Be
Jean-Baptiste Adam t Gris...	Jean-Baptiste Adam 2012 Les Natures Pino
Kirkland Signature aberne...	Kirkland Signature 2011 Mountain Cuvée C
Leon Beyer (Alsace)	Leon Beyer 2012 Gewurztraminer
Louis M. Martini (Alex...	Louis M. Martini 2012 Cabernet Sauvignon
Masseria Setteporte o (Etna)	Masseria Setteporte 2012 Ross
Mirassou	Mirassou 2012 Chardonnay (Centr
Nicosia	Nicosia 2013 Vulkà Bianc

```
o  (Etna)
Pradorey                                Pradorey 2010 Vendimia Seleccionada Finc
a Vald...
Quinta dos Avidagos                         Quinta dos Avidagos 2011 Avidagos Re
d (Douro)
Quiévremont                                          Quiévremont 2012 Meritage
(Virginia)
Rainstorm                                   Rainstorm 2013 Pinot Gris (Willamett
e Valley)
Richard Böcking                               Richard Böcking 2013 Devon Rieslin
g (Mosel)
Robert Hall                                Robert Hall 2011 Sauvignon Blanc (Pas
o Robles)
St. Julian                              St. Julian 2013 Reserve Late Harvest Rie
sling ...
Stemmari                                     Stemmari 2013 Dalila White (Terre S
iciliane)
Sundance                                             Sundance 2011 Merlot (Maul
e Valley)
Sweet Cheeks                            Sweet Cheeks 2012 Vintner's Reserve Wild
Child...
Tandem                                  Tandem 2011 Ars In Vitro Tempranillo-Mer
lot (N...
Tarara                                          Tarara 2010 #SocialSecret Red
(Virginia)
Tasca d'Almerita                        Tasca d'Almerita 2011 Sallier de la Tour
Inzol...
Terre di Giurfo                          Terre di Giurfo 2013 Belsito Frappato
(Vittoria)
The White Knight                             The White Knight 2011 Riesling (Lak
e County)
Trimbach                                            Trimbach 2012 Gewurztraminer
(Alsace)
Trump                                           Trump 2011 Sauvignon Blanc (Mo
nticello)
Vignerons de Bel Air                     Vignerons de Bel Air 2011 Eté Indien
(Brouilly)
Viticultori Associati Canicatti    Viticultori Associati Canicatti 2008 Sci
alo Re...
dtype: object
```

In [23]:
```
reviews.groupby('winery').title.count()
```

Out[23]:
```
winery
Acrobat                          1
Baglio di Pianetto               1
Bianchi                          1
Canicattì                        1
Casa Silva                       1
Castello di Amorosa              1
Clarksburg Wine Company          1
Domaine de la Madone             1
Duca di Salaparuta               2
Envolve                          2
Erath                            1
Estampa                          1
Felix Lavaque                    1
Feudi del Pisciotto              1
Feudi di San Marzano             1
Feudo Montoni                    1
Feudo di Santa Tresa             1
Gaucho Andino                    1
Hawkins Cellars                  1
Heinz Eifel                      1
Henry Fessy                      1
Jean-Baptiste Adam               1
Kirkland Signature               1
Leon Beyer                       1
Louis M. Martini                 1
Masseria Setteporte              1
Mirassou                         1
Nicosia                          1
Pradorey                         1
Quinta dos Avidagos              1
Quiévremont                      2
Rainstorm                        1
Richard Böcking                  1
Robert Hall                      1
St. Julian                       1
Stemmari                         2
Sundance                         1
Sweet Cheeks                     1
Tandem                           1
Tarara                           1
Tasca d'Almerita                 1
Terre di Giurfo                  2
The White Knight                 1
Trimbach                         1
Trump                            1
Vignerons de Bel Air             1
Viticultori Associati Canicatti  1
Name: title, dtype: int64
```

For even more fine-grained control, you can also group by more than one column. For an example, here's how we would pick out the best wine by country *and* province:

In [24]:
```
#เลือกไวน์ที่คะแนนดีที่สุดตามประเทศ
reviews.groupby(['country']).apply(lambda df: df.loc[df.points.idxmax()])
```

Out[24]:

country	country	description	designation	points	price	province	region_1
Argentina	Argentina	Baked plum, molasses, balsamic vinegar and che...	Felix	87	30.0	Other	Cafayate
Chile	Chile	White flower, lychee and apple aromas carry th...	Estate	86	15.0	Colchagua Valley	NaN
France	France	This has great depth of flavor with its fresh ...	Les Natures	92	27.0	Alsace	Alsace
Germany	Germany	Savory dried thyme notes accent sunnier flavor...	Shine	87	12.0	Rheinhessen	NaN
Italy	Italy	This is dominated by oak and oak-driven aromas...	Rosso	95	NaN	Sicily & Sardinia	Etna
Portugal	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN
Spain	Spain	Blackberry and raspberry aromas show a typical...	Ars In Vitro	87	15.0	Northern Spain	Navarra
US	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	95	65.0	Oregon	Willamette Valley

In [25]:
```
#เลือกไวน์ที่คะแนนดีที่สุดตามจังหวัด
reviews.groupby(['province']).apply(lambda df: df.loc[df.points.idxmax()])
```

Out[25]:

province	country	description	designation	points	price	province	region_
Alsace	France	This has great depth of flavor with its fresh ...	Les Natures	92	27.0	Alsace	Alsac
Beaujolais	France	Red cherry fruit comes laced with light tannin...	Nouveau	86	NaN	Beaujolais	Beaujolais Village
California	US	Oak and earth intermingle around robust aromas...	King Ridge Vineyard	95	69.0	California	Sonom Coas
Colchagua Valley	Chile	White flower, lychee and apple aromas carry th...	Estate	86	15.0	Colchagua Valley	Nal
Douro	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	Nal
Maule Valley	Chile	A berry aroma comes with cola and herb notes. ...	NaN	86	9.0	Maule Valley	Nal
Mendoza Province	Argentina	Raw black-cherry aromas are direct and simple ...	Winemaker Selection	87	13.0	Mendoza Province	Mendoz
Michigan	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	90	13.0	Michigan	Lak Michiga Shor
Mosel	Germany	Zesty orange peels and apple notes abound in t...	Devon	87	24.0	Mosel	Nal

province	country	description	designation	points	price	province	region_
Northern Spain	Spain	Blackberry and raspberry aromas show a typical...	Ars In Vitro	87	15.0	Northern Spain	Navarr
Oregon	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	95	65.0	Oregon	Willamett Valle
Other	Argentina	Baked plum, molasses, balsamic vinegar and che...	Felix	87	30.0	Other	Cafayat
Rheinhessen	Germany	Savory dried thyme notes accent sunnier flavor...	Shine	87	12.0	Rheinhessen	Nal
Sicily & Sardinia	Italy	This is dominated by oak and oak-driven aromas...	Rosso	95	NaN	Sicily & Sardinia	Etn
Southern Italy	Italy	Inky in color, this wine has plump aromas of r...	I Tratturi	92	11.0	Southern Italy	Pugli
Virginia	US	This bottling resembles the New Zealand paradi...	NaN	95	16.0	Virginia	Monticell

In [26]:
```
#สามารถจัดกลุ่มได้มากกว่าหนึ่งคอลัมน์
#ตัวอย่างเช่น เราจะเลือกไวน์ที่คะแนนดีที่สุดตามประเทศและจังหวัดได้อย่างไร:
reviews.groupby(['country', 'province']).apply(lambda df: df.loc[df.points.i
```

Loading [MathJax]/extensions/Safe.js

Out[26]:

country	province	country	description	designation	points	price	province
Argentina	**Mendoza Province**	Argentina	Raw black-cherry aromas are direct and simple ...	Winemaker Selection	87	13.0	Mendoza Province
	Other	Argentina	Baked plum, molasses, balsamic vinegar and che...	Felix	87	30.0	Other
Chile	**Colchagua Valley**	Chile	White flower, lychee and apple aromas carry th...	Estate	86	15.0	Colchagua Valley
	Maule Valley	Chile	A berry aroma comes with cola and herb notes. ...	NaN	86	9.0	Maule Valley
France	**Alsace**	France	This has great depth of flavor with its fresh ...	Les Natures	92	27.0	Alsace
	Beaujolais	France	Red cherry fruit comes laced with light tannin...	Nouveau	86	NaN	Beaujolais
Germany	**Mosel**	Germany	Zesty orange peels and apple notes abound in t...	Devon	87	24.0	Mosel
	Rheinhessen	Germany	Savory dried thyme notes accent sunnier flavor...	Shine	87	12.0	Rheinhessen
Italy	**Sicily & Sardinia**	Italy	This is dominated by oak and	Rosso	95	NaN	Sicily & Sardinia

country	province	country	description	designation	points	price	province
			oak-driven aromas...				
	Southern Italy	Italy	Inky in color, this wine has plump aromas of r...	I Tratturi	92	11.0	Southern Italy
Portugal	**Douro**	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro
Spain	**Northern Spain**	Spain	Blackberry and raspberry aromas show a typical...	Ars In Vitro	87	15.0	Northern Spain
US	**California**	US	Oak and earth intermingle around robust aromas...	King Ridge Vineyard	95	69.0	California
	Michigan	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	90	13.0	Michigan
	Oregon	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	95	65.0	Oregon
	Virginia	US	This bottling resembles the New Zealand paradi...	NaN	95	16.0	Virginia

Another `groupby()` method worth mentioning is `agg()`, which lets you run a bunch of different functions on your DataFrame simultaneously. For example, we can generate a simple statistical summary of the dataset as follows:

การใช้ agg() กับ groupby()

ฟังก์ชัน agg() (Aggregate functionคือฟังก์ชันสรุปผล เช่น len,min,max,sum,count,mean,median) ช่วยให้เราสามารถทำการวิเคราะห์หลายรูปแบบพร้อมกัน

In [27]:
```
#เราสามารถใช้คือ agg()
reviews.groupby(['country']).price.agg([len, min, max])
```

```
/var/folders/83/3fg00w111r7bf7rcsz4nznlh0000gn/T/ipykernel_21811/4213695002.
py:2: FutureWarning: The provided callable <built-in function min> is curren
tly using SeriesGroupBy.min. In a future version of pandas, the provided cal
lable will be used directly. To keep current behavior pass the string "min"
instead.
  reviews.groupby(['country']).price.agg([len, min, max])
/var/folders/83/3fg00w111r7bf7rcsz4nznlh0000gn/T/ipykernel_21811/4213695002.
py:2: FutureWarning: The provided callable <built-in function max> is curren
tly using SeriesGroupBy.max. In a future version of pandas, the provided cal
lable will be used directly. To keep current behavior pass the string "max"
instead.
  reviews.groupby(['country']).price.agg([len, min, max])
```

Out[27]:

country	len	min	max
Argentina	2	13.0	30.0
Chile	3	9.0	22.0
France	6	9.0	30.0
Germany	2	12.0	24.0
Italy	16	10.0	35.0
Portugal	1	15.0	15.0
Spain	2	15.0	28.0
US	20	12.0	69.0

Effective use of `groupby()` will allow you to do lots of really powerful things with your dataset.

Multi-indexes

In all of the examples we've seen thus far we've been working with DataFrame or Series objects with a single-label index. `groupby()` is slightly different in the fact that, depending on the operation we run, it will sometimes result in what is called a multi-index.

A multi-index differs from a regular index in that it has multiple levels. For example:

เมื่อค่าของคอลัมน์เดียวไม่เพียงพอที่จะระบุแถวได้อย่างชัดเจน (เช่น ระเบียนหลายรายการในวันที่เดียวกัน ที่หมายความว่าวันที่เพียงอย่างเดียวไม่เหมาะเป็นดัชนี) เมื่อข้อมูลมีลำดับชั้นเชิงตรรกะ ซึ่ง

หมายความว่ามีมิติหรือ "ระดับ" หลายระดับ นอกจากโครงสร้างแล้ว ดัชนีหลายดัชนียังช่วยให้เรียกค้นข้อมูลที่ซับซ้อนในหน่วยความจำได้ค่อนข้างง่าย

In [28]:
```
countries_reviewed = reviews.groupby(['country', 'province']).title.agg([ler
countries_reviewed
```

Out[28]:

		len
country	**province**	
Argentina	**Mendoza Province**	1
	Other	1
Chile	**Colchagua Valley**	2
	Maule Valley	1
France	**Alsace**	3
	Beaujolais	3
Germany	**Mosel**	1
	Rheinhessen	1
Italy	**Sicily & Sardinia**	15
	Southern Italy	1
Portugal	**Douro**	1
Spain	**Northern Spain**	2
US	**California**	10
	Michigan	1
	Oregon	5
	Virginia	4

In [29]:
```
countries_reviewed.index
```

```
Out[29]: MultiIndex([('Argentina',  'Mendoza Province'),
                     ('Argentina',             'Other'),
                     (    'Chile',  'Colchagua Valley'),
                     (    'Chile',      'Maule Valley'),
                     (   'France',           'Alsace'),
                     (   'France',        'Beaujolais'),
                     (  'Germany',             'Mosel'),
                     (  'Germany',       'Rheinhessen'),
                     (    'Italy', 'Sicily & Sardinia'),
                     (    'Italy',    'Southern Italy'),
                     ( 'Portugal',             'Douro'),
                     (    'Spain',    'Northern Spain'),
                     (       'US',        'California'),
                     (       'US',          'Michigan'),
                     (       'US',            'Oregon'),
                     (       'US',          'Virginia')],
                    names=['country', 'province'])
```

```
In [30]: type(countries_reviewed.index)
```

```
Out[30]: pandas.core.indexes.multi.MultiIndex
```

Multi-indices have several methods for dealing with their tiered structure which are absent for single-level indices. They also require two levels of labels to retrieve a value. Dealing with multi-index output is a common "gotcha" for users new to pandas.

The use cases for a multi-index are detailed alongside instructions on using them in the MultiIndex / Advanced Selection section of the pandas documentation.

However, in general the multi-index method you will use most often is the one for converting back to a regular index, the `reset_index()` method:

```
In [31]: countries_reviewed.reset_index()
```

Out[31]:

	country	province	len
0	Argentina	Mendoza Province	1
1	Argentina	Other	1
2	Chile	Colchagua Valley	2
3	Chile	Maule Valley	1
4	France	Alsace	3
5	France	Beaujolais	3
6	Germany	Mosel	1
7	Germany	Rheinhessen	1
8	Italy	Sicily & Sardinia	15
9	Italy	Southern Italy	1
10	Portugal	Douro	1
11	Spain	Northern Spain	2
12	US	California	10
13	US	Michigan	1
14	US	Oregon	5
15	US	Virginia	4

In [32]:
```
type(countries_reviewed.reset_index())
```

Out[32]:
```
pandas.core.frame.DataFrame
```

Sorting

Looking again at `countries_reviewed` we can see that grouping returns data in index order, not in value order. That is to say, when outputting the result of a `groupby`, the order of the rows is dependent on the values in the index, not in the data.

To get data in the order want it in we can sort it ourselves. The `sort_values()` method is handy for this.

In [33]:
```
countries_reviewed = countries_reviewed.reset_index()
countries_reviewed.sort_values(by='len')
```

Out[33]:

	country	province	len
0	Argentina	Mendoza Province	1
1	Argentina	Other	1
3	Chile	Maule Valley	1
6	Germany	Mosel	1
7	Germany	Rheinhessen	1
9	Italy	Southern Italy	1
10	Portugal	Douro	1
13	US	Michigan	1
2	Chile	Colchagua Valley	2
11	Spain	Northern Spain	2
4	France	Alsace	3
5	France	Beaujolais	3
15	US	Virginia	4
14	US	Oregon	5
12	US	California	10
8	Italy	Sicily & Sardinia	15

In [34]:
```
countries_reviewed.sort_values('len')
```

Out[34]:

	country	province	len
0	Argentina	Mendoza Province	1
1	Argentina	Other	1
3	Chile	Maule Valley	1
6	Germany	Mosel	1
7	Germany	Rheinhessen	1
9	Italy	Southern Italy	1
10	Portugal	Douro	1
13	US	Michigan	1
2	Chile	Colchagua Valley	2
11	Spain	Northern Spain	2
4	France	Alsace	3
5	France	Beaujolais	3
15	US	Virginia	4
14	US	Oregon	5
12	US	California	10
8	Italy	Sicily & Sardinia	15

`sort_values()` defaults to an ascending sort, where the lowest values go first. However, most of the time we want a descending sort, where the higher numbers go first. That goes thusly:

In [35]:
```
countries_reviewed.sort_values(by='len').iloc[::-1]
```

Out[35]:

	country	province	len
8	Italy	Sicily & Sardinia	15
12	US	California	10
14	US	Oregon	5
15	US	Virginia	4
5	France	Beaujolais	3
4	France	Alsace	3
11	Spain	Northern Spain	2
2	Chile	Colchagua Valley	2
13	US	Michigan	1
10	Portugal	Douro	1
9	Italy	Southern Italy	1
7	Germany	Rheinhessen	1
6	Germany	Mosel	1
3	Chile	Maule Valley	1
1	Argentina	Other	1
0	Argentina	Mendoza Province	1

In [36]:
```
countries_reviewed.sort_values(by='len', ascending=False)
```

Out[36]:

	country	province	len
8	Italy	Sicily & Sardinia	15
12	US	California	10
14	US	Oregon	5
15	US	Virginia	4
4	France	Alsace	3
5	France	Beaujolais	3
2	Chile	Colchagua Valley	2
11	Spain	Northern Spain	2
0	Argentina	Mendoza Province	1
1	Argentina	Other	1
3	Chile	Maule Valley	1
6	Germany	Mosel	1
7	Germany	Rheinhessen	1
9	Italy	Southern Italy	1
10	Portugal	Douro	1
13	US	Michigan	1

To sort by index values, use the companion method `sort_index()`. This method has the same arguments and default order:

In [37]:
```
countries_reviewed.sort_index()
```

Out[37]:

	country	province	len
0	Argentina	Mendoza Province	1
1	Argentina	Other	1
2	Chile	Colchagua Valley	2
3	Chile	Maule Valley	1
4	France	Alsace	3
5	France	Beaujolais	3
6	Germany	Mosel	1
7	Germany	Rheinhessen	1
8	Italy	Sicily & Sardinia	15
9	Italy	Southern Italy	1
10	Portugal	Douro	1
11	Spain	Northern Spain	2
12	US	California	10
13	US	Michigan	1
14	US	Oregon	5
15	US	Virginia	4

Finally, know that you can sort by more than one column at a time:

In [38]:
```
countries_reviewed.sort_values(by=['country', 'len'])
```

Out[38]:

	country	province	len
0	Argentina	Mendoza Province	1
1	Argentina	Other	1
3	Chile	Maule Valley	1
2	Chile	Colchagua Valley	2
4	France	Alsace	3
5	France	Beaujolais	3
6	Germany	Mosel	1
7	Germany	Rheinhessen	1
9	Italy	Southern Italy	1
8	Italy	Sicily & Sardinia	15
10	Portugal	Douro	1
11	Spain	Northern Spain	2
13	US	Michigan	1
15	US	Virginia	4
14	US	Oregon	5
12	US	California	10

Your turn

If you haven't started the exercise, you can start now.

การจัดการข้อมูลสูญหาย

การรวบรวมข้อมูลมาวิเคราะห์นั้น บางครั้งอาจจะมีข้อมูลที่ได้มา ไม่ครบบ้าง ตกหล่นหรือขาดหายไปบ้างเรียกส่วนนี้ว่า Missing Data หรือ Missing Value ในหัวข้อนี้จะมาตรวจสอบข้อมูลและจัดการข้อมูลสูญหาย (Clean Data)

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

df = pd.read_csv("datasets/Employee.csv")
df
```

Out[1]:

	Name	Job	Age	Salary	Bonus	Address
0	A	Programmer	20.0	30000.0	10%	123.0
1	B	Programmer	18.0	NaN	10%	NaN
2	C	Developer	NaN	32000.0	10%	NaN
3	D	NaN	23.0	40000.0	10%	NaN
4	E	Sale	29.0	40000.0	10%	NaN
5	F	Manager	NaN	75000.0	10%	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN
7	H	Maketing	34.0	60000.0	10%	NaN
8	I	NaN	26.0	100000.0	10%	NaN
9	H	Maketing	34.0	60000.0	10%	NaN
10	E	Sale	29.0	40000.0	10%	NaN

In [2]:
```
import pandas as pd

df = pd.read_csv("datasets/Employee.csv", index_col="Name")
df
```

Out[2]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	NaN	10%	NaN
C	Developer	NaN	32000.0	10%	NaN
D	NaN	23.0	40000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN
F	Manager	NaN	75000.0	10%	NaN
NaN	NaN	NaN	NaN	NaN	NaN
H	Maketing	34.0	60000.0	10%	NaN
I	NaN	26.0	100000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN

In [3]:
```
df.shape
```

Out[3]:
```
(11, 5)
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 11 entries, A to E
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Job      8 non-null      object 
 1   Age      8 non-null      float64
 2   Salary   9 non-null      float64
 3   Bonus    10 non-null     object 
 4   Address  1 non-null      float64
dtypes: float64(3), object(2)
memory usage: 528.0+ bytes
```

In [5]:
```
#การตรวจสอบข้อมูลสูญหายด้วย isnull()
df.isnull()
```

Out[5]:

Name	Job	Age	Salary	Bonus	Address
A	False	False	False	False	False
B	False	False	True	False	True
C	False	True	False	False	True
D	True	False	False	False	True
E	False	False	False	False	True
F	False	True	False	False	True
NaN	True	True	True	True	True
H	False	False	False	False	True
I	True	False	False	False	True
H	False	False	False	False	True
E	False	False	False	False	True

In [6]:
```
#ตรวจสอบว่ามีคอลัมน์ใดบ้างที่ไม่มีข้อมูล
df.isnull().any()
```

Out[6]:
```
Job        True
Age        True
Salary     True
Bonus      True
Address    True
dtype: bool
```

In [7]:
```
#นับจำนวนคอลัมน์ที่ไม่มีข้อมูล
df.isnull().sum()
```

Out[7]:
```
Job         3
Age         3
Salary      2
Bonus       1
Address    10
dtype: int64
```

In [8]:
```
#การตรวจสอบข้อมูลครบถ้วนด้วย notnull()
df.notnull()
```

Out[8]:

Name	Job	Age	Salary	Bonus	Address
A	True	True	True	True	True
B	True	True	False	True	False
C	True	False	True	True	False
D	False	True	True	True	False
E	True	True	True	True	False
F	True	False	True	True	False
NaN	False	False	False	False	False
H	True	True	True	True	False
I	False	True	True	True	False
H	True	True	True	True	False
E	True	True	True	True	False

In [9]:
```
#ตรวจสอบว่ามีคอลัมน์ใดบ้างที่มีข้อมูล
df.notnull().any()
```

Out[9]:
```
Job        True
Age        True
Salary     True
Bonus      True
Address    True
dtype: bool
```

In [10]:
```
#นับจำนวนคอลัมน์ที่มีข้อมูล
df.isnull().sum()
```

Out[10]:
```
Job         3
Age         3
Salary      2
Bonus       1
Address    10
dtype: int64
```

วิธีจัดการข้อมูลสูญหาย

- แทนที่ด้วยค่าเฉลี่ยข้อมูลทั้งหมด
- แทนที่ด้วยค่าตรงๆที่กำหนดขึ้นมา
- แทนที่ด้วยค่าก่อนหน้า
- แทนที่ด้วยค่าถัดไป
- ลบข้อมูล

แทนที่ด้วยค่าเฉลี่ยข้อมูลทั้งหมด

In [11]:
```
df.describe()
```

Out[11]:

	Age	Salary	Address
count	8.000000	9.000000	1.0
mean	26.625000	53000.000000	123.0
std	5.998512	23097.618925	NaN
min	18.000000	30000.000000	123.0
25%	22.250000	40000.000000	123.0
50%	27.500000	40000.000000	123.0
75%	30.250000	60000.000000	123.0
max	34.000000	100000.000000	123.0

In [12]:
```
#นำเข้า DataFrame ใหม่
df = pd.read_csv("datasets/Employee.csv", index_col="Name")
df
```

Out[12]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	NaN	10%	NaN
C	Developer	NaN	32000.0	10%	NaN
D	NaN	23.0	40000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN
F	Manager	NaN	75000.0	10%	NaN
NaN	NaN	NaN	NaN	NaN	NaN
H	Maketing	34.0	60000.0	10%	NaN
I	NaN	26.0	100000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN

In [13]:
```
#แทนที่ด้วยค่าเฉลี่ยข้อมูลทั้งหมด
df['Salary'] = df['Salary'].fillna(df['Salary'].mean())
df
```

Out[13]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	53000.0	10%	NaN
C	Developer	NaN	32000.0	10%	NaN
D	NaN	23.0	40000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN
F	Manager	NaN	75000.0	10%	NaN
NaN	NaN	NaN	53000.0	NaN	NaN
H	Maketing	34.0	60000.0	10%	NaN
I	NaN	26.0	100000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN

แทนที่ด้วยค่าตรงๆที่กำหนดขึ้นมา

In [14]:
```
#นำเข้า DataFrame ใหม่
df = pd.read_csv("datasets/Employee.csv", index_col="Name")
df
```

Out[14]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	NaN	10%	NaN
C	Developer	NaN	32000.0	10%	NaN
D	NaN	23.0	40000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN
F	Manager	NaN	75000.0	10%	NaN
NaN	NaN	NaN	NaN	NaN	NaN
H	Maketing	34.0	60000.0	10%	NaN
I	NaN	26.0	100000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN

In [15]:
```
#แทนที่ด้วยค่าตรงๆที่กำหนดขึ้นมา
df['Salary'] = df['Salary'].fillna(22000)
df
```

Out[15]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	22000.0	10%	NaN
C	Developer	NaN	32000.0	10%	NaN
D	NaN	23.0	40000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN
F	Manager	NaN	75000.0	10%	NaN
NaN	NaN	NaN	22000.0	NaN	NaN
H	Maketing	34.0	60000.0	10%	NaN
I	NaN	26.0	100000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN

แทนที่ด้วยค่าก่อนหน้า

In [16]:
```
#นำเข้า DataFrame ใหม่
df = pd.read_csv("datasets/Employee.csv", index_col="Name")
df
```

Out[16]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	NaN	10%	NaN
C	Developer	NaN	32000.0	10%	NaN
D	NaN	23.0	40000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN
F	Manager	NaN	75000.0	10%	NaN
NaN	NaN	NaN	NaN	NaN	NaN
H	Maketing	34.0	60000.0	10%	NaN
I	NaN	26.0	100000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN

In [17]:
```
#แทนที่ด้วยค่าก่อนหน้า
df.fillna(method='pad')
```

```
/var/folders/83/3fg00w111r7bf7rcsz4nznlh0000gn/T/ipykernel_21816/3352054385.
py:2: FutureWarning: DataFrame.fillna with 'method' is deprecated and will r
aise in a future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method='pad')
```

Out[17]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	30000.0	10%	123.0
C	Developer	18.0	32000.0	10%	123.0
D	Developer	23.0	40000.0	10%	123.0
E	Sale	29.0	40000.0	10%	123.0
F	Manager	29.0	75000.0	10%	123.0
NaN	Manager	29.0	75000.0	10%	123.0
H	Maketing	34.0	60000.0	10%	123.0
I	Maketing	26.0	100000.0	10%	123.0
H	Maketing	34.0	60000.0	10%	123.0
E	Sale	29.0	40000.0	10%	123.0

แทนที่ด้วยค่าถัดไป

In [18]:
```
#นำเข้า DataFrame ใหม่
df = pd.read_csv("datasets/Employee.csv", index_col="Name")
df
```

Out[18]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	NaN	10%	NaN
C	Developer	NaN	32000.0	10%	NaN
D	NaN	23.0	40000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN
F	Manager	NaN	75000.0	10%	NaN
NaN	NaN	NaN	NaN	NaN	NaN
H	Maketing	34.0	60000.0	10%	NaN
I	NaN	26.0	100000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN

In [19]:
```
#แทนที่ด้วยค่าถัดไป
df.fillna(method='bfill')
```

```
/var/folders/83/3fg00w111r7bf7rcsz4nznlh0000gn/T/ipykernel_21816/2964409054.
py:2: FutureWarning: DataFrame.fillna with 'method' is deprecated and will r
aise in a future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method='bfill')
```

Out[19]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	32000.0	10%	NaN
C	Developer	23.0	32000.0	10%	NaN
D	Sale	23.0	40000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN
F	Manager	34.0	75000.0	10%	NaN
NaN	Maketing	34.0	60000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
I	Maketing	26.0	100000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN

ลบข้อมูล

- ลบทิ้งทั้งหมด
- ลบแถวบางส่วน
- ลบคอลัมน์บางส่วน
- ลบค่าซ้ำ

ลบทิ้งทั้งหมด

In [20]:
```
#นำเข้า DataFrame ใหม่
df = pd.read_csv("datasets/Employee.csv", index_col="Name")
df
```

Out[20]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	NaN	10%	NaN
C	Developer	NaN	32000.0	10%	NaN
D	NaN	23.0	40000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN
F	Manager	NaN	75000.0	10%	NaN
NaN	NaN	NaN	NaN	NaN	NaN
H	Maketing	34.0	60000.0	10%	NaN
I	NaN	26.0	100000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN

In [21]:
```
#ลบทิ้งทั้งหมด
df.dropna()
```

Out[21]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0

ลบแถวบางส่วนที่มีค่าว่าง

In [22]:
```
#นำเข้า DataFrame ใหม่
df = pd.read_csv("datasets/Employee.csv", index_col="Name")
df
```

Out[22]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	NaN	10%	NaN
C	Developer	NaN	32000.0	10%	NaN
D	NaN	23.0	40000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN
F	Manager	NaN	75000.0	10%	NaN
NaN	NaN	NaN	NaN	NaN	NaN
H	Maketing	34.0	60000.0	10%	NaN
I	NaN	26.0	100000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN

In [23]:
```
#ลบแถวบางส่วนที่มีค่าว่าง
df.dropna(subset=['Age','Job'])
```

Out[23]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	NaN	10%	NaN
E	Sale	29.0	40000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN

ลบคอลัมน์บางส่วนที่มีค่าว่าง

In [24]:
```
#นำเข้า DataFrame ใหม่
df = pd.read_csv("datasets/Employee.csv", index_col="Name")
df
```

Out[24]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	NaN	10%	NaN
C	Developer	NaN	32000.0	10%	NaN
D	NaN	23.0	40000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN
F	Manager	NaN	75000.0	10%	NaN
NaN	NaN	NaN	NaN	NaN	NaN
H	Maketing	34.0	60000.0	10%	NaN
I	NaN	26.0	100000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN

In [25]:
```
df.dropna(axis='columns')
```

Out[25]:

Name
A
B
C
D
E
F
NaN
H
I
H
E

ลบค่าซ้ำ

In [26]:
```
#นำเข้า DataFrame ใหม่
df = pd.read_csv("datasets/Employee.csv", index_col="Name")
df
```

Out[26]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	NaN	10%	NaN
C	Developer	NaN	32000.0	10%	NaN
D	NaN	23.0	40000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN
F	Manager	NaN	75000.0	10%	NaN
NaN	NaN	NaN	NaN	NaN	NaN
H	Maketing	34.0	60000.0	10%	NaN
I	NaN	26.0	100000.0	10%	NaN
H	Maketing	34.0	60000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN

In [27]:
```
#เช็คค่าซ้ำ
df[df.duplicated]
```

Out[27]:

Name	Job	Age	Salary	Bonus	Address
H	Maketing	34.0	60000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN

In [28]:
```
#ลบค่าซ้ำ
df.drop_duplicates()
```

Out[28]:

Name	Job	Age	Salary	Bonus	Address
A	Programmer	20.0	30000.0	10%	123.0
B	Programmer	18.0	NaN	10%	NaN
C	Developer	NaN	32000.0	10%	NaN
D	NaN	23.0	40000.0	10%	NaN
E	Sale	29.0	40000.0	10%	NaN
F	Manager	NaN	75000.0	10%	NaN
NaN	NaN	NaN	NaN	NaN	NaN
H	Maketing	34.0	60000.0	10%	NaN
I	NaN	26.0	100000.0	10%	NaN

In [ ]:

Introduction

In this tutorial, you'll learn how to investigate data types within a DataFrame or Series. You'll also learn how to find and replace entries.

Dtypes

The data type for a column in a DataFrame or a Series is known as the **dtype**.

You can use the `dtype` property to grab the type of a specific column. For instance, we can get the dtype of the `price` column in the `reviews` DataFrame:

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

reviews = pd.read_csv("datasets/winemag-data-130k-v2.csv", index_col=0)
reviews
```

Out[1]:

	country	description	designation	points	price	province	region_1	regior
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	N
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	N
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willame Va
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	N
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willame Va
...	...	...	...	...	...	...	...	
65494	France	Made from young vines from the Vaulorent porti...	Fourchaume Premier Cru	90	45.0	Burgundy	Chablis	N
65495	Australia	This is a big, fat, almost sweet-tasting Caber...	NaN	90	22.0	South Australia	McLaren Vale	N
65496	US	Much improved over the unripe 2005, Fritz's 20...	Estate	90	20.0	California	Dry Creek Valley	Sonc
65497	US	This wine wears its 15.8% alcohol	Block 24	90	31.0	California	Napa Valley	Na

	country	description	designation	points	price	province	region_1	regior
		better than ...						
65498	Spain	A unique take on Manzanilla Sherry, which is o...	Manzanilla	90	10.0	Andalucia	Jerez	N

65499 rows × 13 columns

In [2]:
```
reviews.price.dtype
```

Out[2]:
```
dtype('float64')
```

Alternatively, the `dtypes` property returns the `dtype` of *every* column in the DataFrame:

In [3]:
```
reviews.dtypes
```

Out[3]:
```
country                  object
description              object
designation              object
points                    int64
price                   float64
province                 object
region_1                 object
region_2                 object
taster_name              object
taster_twitter_handle    object
title                    object
variety                  object
winery                   object
dtype: object
```

Data types tell us something about how pandas is storing the data internally. `float64` means that it's using a 64-bit floating point number; `int64` means a similarly sized integer instead, and so on.

One peculiarity to keep in mind (and on display very clearly here) is that columns consisting entirely of strings do not get their own type; they are instead given the `object` type.

It's possible to convert a column of one type into another wherever such a conversion makes sense by using the `astype()` function. For example, we may transform the `points` column from its existing `int64` data type into a `float64` data type:

In [4]:
```
reviews.points.astype('float64')
```

Out[4]:
```
0        87.0
1        87.0
2        87.0
3        87.0
4        87.0
         ...
65494    90.0
65495    90.0
65496    90.0
65497    90.0
65498    90.0
Name: points, Length: 65499, dtype: float64
```

A DataFrame or Series index has its own `dtype`, too:

In [5]:
```
reviews.index.dtype
```

Out[5]:
```
dtype('int64')
```

Pandas also supports more exotic data types, such as categorical data and timeseries data. Because these data types are more rarely used, we will omit them until a much later section of this tutorial.

Missing data

Entries missing values are given the value `NaN`, short for "Not a Number". For technical reasons these `NaN` values are always of the `float64` dtype.

Pandas provides some methods specific to missing data. To select `NaN` entries you can use `pd.isnull()` (or its companion `pd.notnull()`). This is meant to be used thusly:

In [6]:
```
reviews.region_2
```

Out[6]:
```
0                    NaN
1                    NaN
2        Willamette Valley
3                    NaN
4        Willamette Valley
               ...
65494                NaN
65495                NaN
65496             Sonoma
65497               Napa
65498                NaN
Name: region_2, Length: 65499, dtype: object
```

In [7]:
```
reviews[pd.isnull(reviews.country)]
```

Out[7]:

	country	description	designation	points	price	province	region_1	region_
913	NaN	Amber in color, this wine has aromas of peach ...	Asureti Valley	87	30.0	NaN	NaN	Na
3131	NaN	Soft, fruity and juicy, this is a pleasant, si...	Partager	83	NaN	NaN	NaN	Na
4243	NaN	Violet-red in color, this semisweet wine has a...	Red Naturally Semi-Sweet	88	18.0	NaN	NaN	Na
9509	NaN	This mouthwatering blend starts with a nose of...	Theopetra Malagouzia-Assyrtiko	92	28.0	NaN	NaN	Na
9750	NaN	This orange-style wine has a cloudy yellow-gol...	Orange Nikolaevo Vineyard	89	28.0	NaN	NaN	Na
11150	NaN	A blend of 85% Melnik, 10% Grenache Noir and 5...	NaN	89	20.0	NaN	NaN	Na
11348	NaN	Light and fruity, this is a wine that has some...	Partager	82	NaN	NaN	NaN	Na
14030	NaN	This Furmint, grown in marl soils, has aromas ...	Márga	88	25.0	NaN	NaN	Na
16000	NaN	Jumpy, jammy aromas of foxy black fruits are s...	Valle de los Manantiales Vineyard	86	40.0	NaN	NaN	Na
16749	NaN	Winemaker: Bartho Eksteen. This wooded Sauvy s...	Cape Winemakers Guild Vloekskoot Wooded	91	NaN	NaN	NaN	Na
18075	NaN	Delicate white flowers and a spin of lemon pee...	Askitikos	90	17.0	NaN	NaN	Na

	country	description	designation	points	price	province	region_1	region
26485	NaN	This wine has aromas of black berry, dried red...	NaN	87	13.0	NaN	NaN	Na
26486	NaN	Aromas of green apple and white flowers prepar...	NaN	87	14.0	NaN	NaN	Na
26489	NaN	Balanced aromas of green herbs and citrus zest...	Aliwen Reserva	87	12.0	NaN	NaN	Na
27822	NaN	This is a reasonably rich, concentrated exampl...	NaN	86	19.0	NaN	NaN	Na
36112	NaN	An interesting blend of indigenous Bulgarian a...	Hrumki Melnik 55 Mourvèdre Marselan	89	25.0	NaN	NaN	Na
38240	NaN	Subdued citrus and pear notes on the nose find...	Steirische Klassik	89	24.0	NaN	NaN	Na
38898	NaN	Scents of clover, stem, green herb and red cur...	Wismer-Parke Vineyard	89	34.0	NaN	NaN	Na
44674	NaN	Crisp apple freshness almost tips into full ci...	Steirische Klassik	91	25.0	NaN	NaN	Na
44850	NaN	This blend of Gamay and Prokupe has aromas of ...	Amphora	84	6.0	NaN	NaN	Na
44851	NaN	This wine has aromas of honeysuckle and lemon ...	Royal	84	6.0	NaN	NaN	Na
45247	NaN	Just a whiff of citrus shows on the restrained...	Steirische Klassik	89	25.0	NaN	NaN	Na

	country	description	designation	points	price	province	region_1	region
45402	NaN	Basic cherry aromas turn more earthy and soupy...	Reserva Estate Bottled	85	12.0	NaN	NaN	Na
46352	NaN	A dark color and rich, jammy, baked aromas of ...	Catalina	91	50.0	NaN	NaN	Na
49425	NaN	This blend is comprised of 55% Merlot, 21% Cab...	Getika Made With Organic Grapes	88	28.0	NaN	NaN	Na
49426	NaN	Enticing aromas of blueberry syrup open this b...	Getika Made With Organic Grapes	88	28.0	NaN	NaN	Na
49427	NaN	This dark-garnet wine has aromas of eucalyptus...	Hrumki Syrah Melnik 55 Mourvèdre Marselan	88	19.0	NaN	NaN	Na
49510	NaN	Aromas of cherry, blueberry and rose petal pre...	NaN	91	34.0	NaN	NaN	Na
54222	NaN	Almost caramel in color, this wine offers arom...	Babaneuri Valley	87	30.0	NaN	NaN	Na
57612	NaN	Winemaker: Gordon Newton Johnson. This is such...	Cape Winemakers Guild Windansea	92	NaN	NaN	NaN	Na
59670	NaN	The heady florality of damask rose is joined b...	Steintal	92	38.0	NaN	NaN	Na
60678	NaN	This wine was made for grilled meats, with its...	Dry	86	17.0	NaN	NaN	Na

Replacing missing values is a common operation. Pandas provides a really handy method for this problem: `fillna()`. `fillna()` provides a few different strategies for mitigating such data. For example, we can simply replace each `NaN` with an `"Unknown"`:

In [8]:
```
reviews.region_2.fillna("Unknown")
```

Out[8]:
```
0                  Unknown
1                  Unknown
2        Willamette Valley
3                  Unknown
4        Willamette Valley
               ...
65494              Unknown
65495              Unknown
65496               Sonoma
65497                 Napa
65498              Unknown
Name: region_2, Length: 65499, dtype: object
```

Or we could fill each missing value with the first non-null value that appears sometime after the given record in the database. This is known as the backfill strategy.

Alternatively, we may have a non-null value that we would like to replace. For example, suppose that since this dataset was published, reviewer Kerin O'Keefe has changed her Twitter handle from `@kerinokeefe` to `@kerino`. One way to reflect this in the dataset is using the `replace()` method:

In [9]:
```
reviews.taster_twitter_handle.replace("@kerinokeefe", "@kerino")
```

Out[9]:
```
0           @kerino
1         @vossroger
2        @paulgwine 
3                NaN
4        @paulgwine 
            ...
65494     @vossroger
65495         @JoeCz
65496            NaN
65497            NaN
65498    @wineschach
Name: taster_twitter_handle, Length: 65499, dtype: object
```

The `replace()` method is worth mentioning here because it's handy for replacing missing data which is given some kind of sentinel value in the dataset: things like `"Unknown"`, `"Undisclosed"`, `"Invalid"`, and so on.

Your turn