

Machine Learning with Python for Education and Personnel Economists

Michael E. Rose, PhD

Introduction



Who am I?

- Senior Research Fellow, Max Planck Institute for Innovation and Competition, Economics PhD from University of Cape Town
- Writing code since 8th grade
- Author of 3 open-source projects: scholarmetrics, pybliometrics, sosia
- Teaching experience:
 - Python, Big Data and Machine Learning for Economists @ LMU Munich, ifo Institute Munich, Scheller College of Business at Georgia Tech
 - Risk Management Computing Skills [Matlab, SQL, Excel, VBA] @ University of Cape Town
- Michael.Ernst.Rose@gmail.com

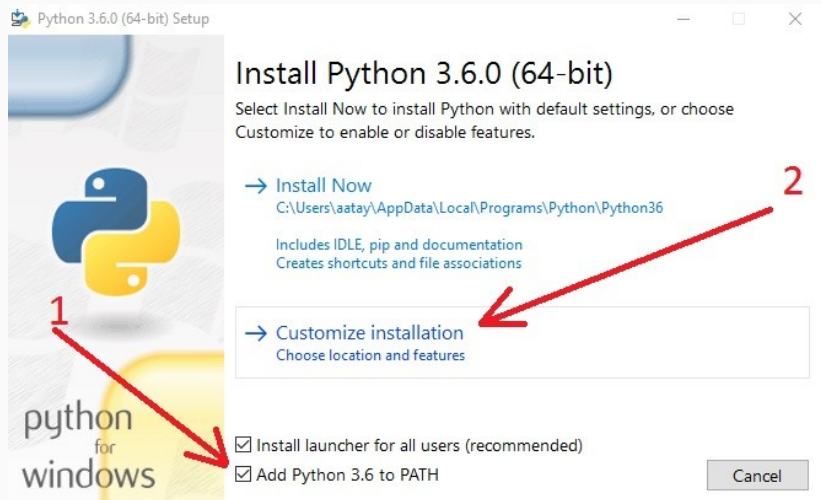


Who are you?

- Name, Status
- Which languages, how long?
- Which operating system?
- Is the computer your slave, or are you your computer's slave?

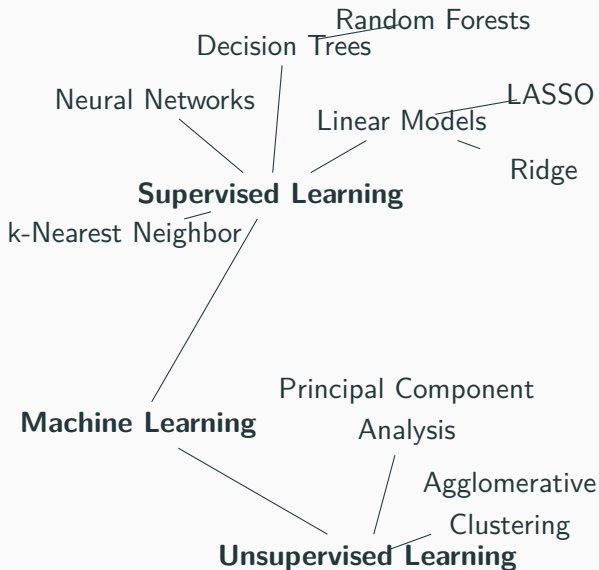
Time to download and install Python

<https://www.python.org/downloads/>



1. Empirical research using Python
2. Project management
3. Supervised Machine Learning
4. Unsupervised Machine Learning
5. Natural Language Processing

Course Content, cont.



- Lecture in the morning, exercises in the afternoon
- Each exercise session starts with a Monty Python sketch
- 10 Minutes breaks after 50 Minutes of Teaching

Learning outcomes

- Programming part
 1. List some of the right basic tools for empirical research
 2. Use python independently
 3. Apply pandas, seaborn, sklearn
 4. Understand coding principles
 5. Use version control with git
- Machine Learning
 1. Apply Neural Networks, Random Forests, Clustering algorithms
 2. Interpret and evaluate machine learning applications
 3. Teach yourself how to apply machine learning algorithms we don't speak about

Why Python?

- Interpreted, high-level, general-purpose programming language
- Can be object-oriented, imperative, functional and procedural
- Free (= no licenses)
- Large (= support and many packages)
- Centralized development
- Very good first language

Why Python?

- Interpreted, high-level, general-purpose programming language
- Can be object-oriented, imperative, functional and procedural
- Free (= no licenses)
- Large (= support and many packages)
- Centralized development
- Very good first language

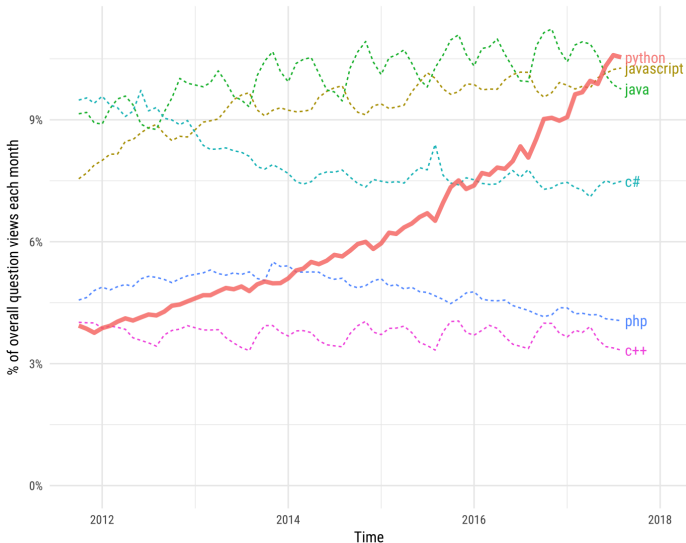
There should be one– and preferably only one –obvious way to do it.

Although that way may not be obvious at first unless you're Dutch. (Tim Peters - The Zen of Python)

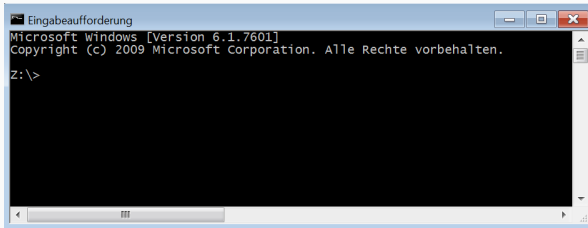
Python is popular and increasing in popularity

Growth of major programming languages

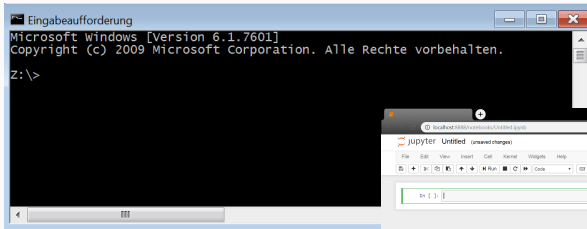
Based on Stack Overflow question views in World Bank high-income countries



How to use Python?

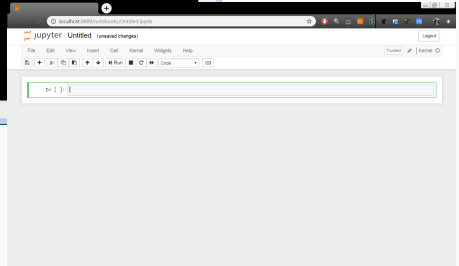


How to use Python?

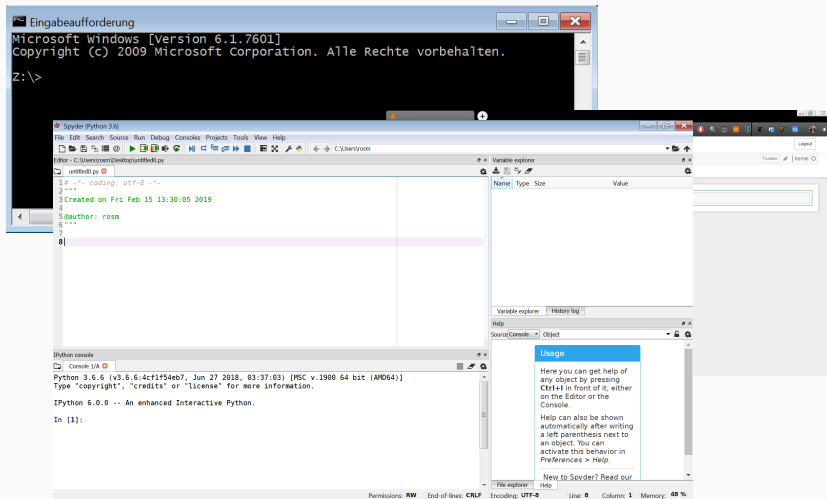


A screenshot of a Windows command prompt window titled "Eingabeaufforderung". The window has a black background with white text. The text displayed is: "Microsoft Windows [Version 6.1.7601] Copyright (c) 2009 Microsoft Corporation. Alle Rechte vorbehalten. Z:\>". The window includes standard Windows window controls (minimize, maximize, close) in the top right corner and a scroll bar on the right side.

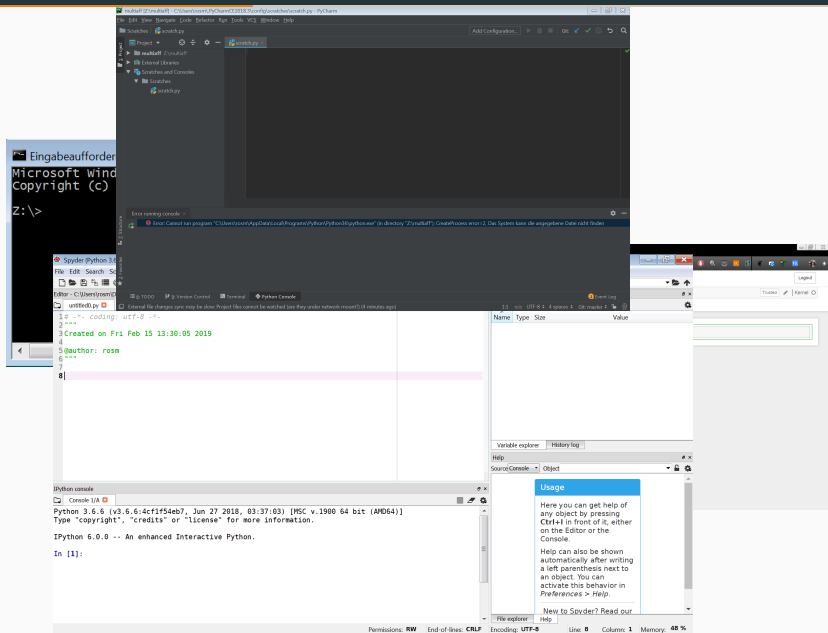
```
Eingabeaufforderung
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Alle Rechte vorbehalten.
Z:\>
```



How to use Python?



How to use Python?



Why I discourage anaconda

- packages shipped with anaconda need to be installed with `conda install`
- packages tend to be outdated
- Overkill/Unnecessary software
- Jupyter and spyder run without anaconda as well

- Console uses DOS language (Windows) or `shell` and `bash`
- Starts python environment, Jupyter, spyder ...¹
- Install packages here using `pip`²
- Execute scripts: `./<scriptname>` (unless you're on Windows)
- Get a proper text editor (Sublime 2, Notepad ++, NOT WordPad)

¹Windows users: make sure Python is in the environment paths

²Windows users: make sure pip is in the environment paths

- Console uses DOS language (Windows) or shell and bash
- Starts python environment, Jupyter, spyder ...¹
- Install packages here using pip²
- Execute scripts: ./<scriptname> (unless you're on Windows)
- Get a proper text editor (Sublime 2, Notepad ++, NOT WordPad)

```
pip install pandas
```

```
pip install matplotlib
```

```
pip install numpy
```

```
pip install pandas --upgrade
```

¹Windows users: make sure Python is in the environment paths

²Windows users: make sure pip is in the environment paths

- Create a folder for this course and navigate there in your terminal

- Create a folder for this course and navigate there in your terminal

- Type

```
pip install --upgrade pip  
pip install notebook  
jupyter notebook
```

- Your browser will fire up, with cells for either text or code
- Files will be saved relative to where you started the Jupyter server

Type in cell

```
1 %matplotlib inline
2
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6 x = np.linspace(-10, 10, 100)
7 y = np.sin(x)
8 plt.plot(x, y, marker="x")
```

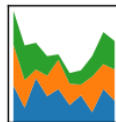
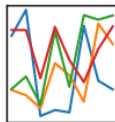
What matters in Python?

- Indentation is key (convention: four spaces)
- Case-sensitive
- Variables must not start with numbers
- It's a language, *not* a program

Pandas

pandas

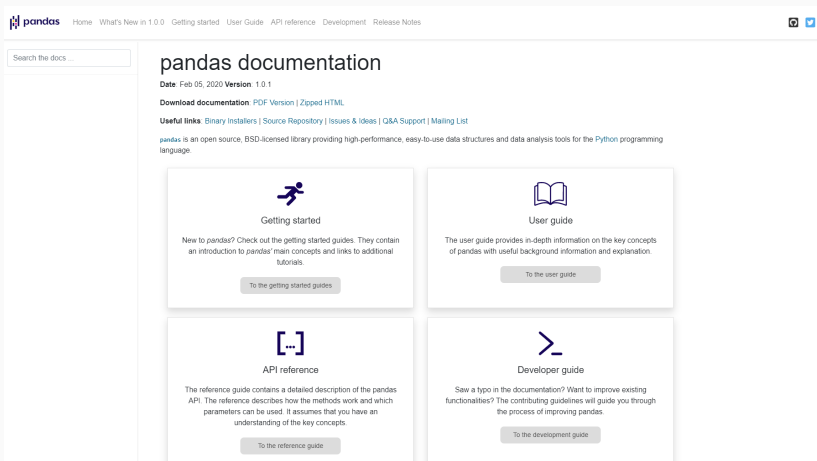
$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



pandas: a library for data manipulation

- Documentation:

<http://pandas.pydata.org/pandas-docs/stable/>



The screenshot shows the pandas documentation website. At the top, there is a navigation bar with the pandas logo and links: Home, What's New in 1.0.0, Getting started, User Guide, API reference, Development, and Release Notes. On the right side of the navigation bar are social media icons for GitHub and Twitter. Below the navigation bar is a search box labeled "Search the docs...". The main content area is titled "pandas documentation". Below the title, it shows the date "Feb 05, 2020" and version "1.0.1". There are links to "Download documentation" in PDF or Zipped HTML format. A "Useful links" section includes links to Binary Installers, Source Repository, Issues & Ideas, Q&A Support, and Mailing List. A paragraph describes pandas as an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Below this, there are four large cards, each representing a different section of the documentation: 1. "Getting started" with a running person icon, a brief description, and a button "To the getting started guides". 2. "User guide" with an open book icon, a brief description, and a button "To the user guide". 3. "API reference" with a code block icon, a brief description, and a button "To the reference guide". 4. "Developer guide" with a greater-than sign icon, a brief description, and a button "To the development guide".


pandas documentation

Date: Feb 05, 2020 Version: 1.0.1

Download documentation: [PDF Version](#) | [Zipped HTML](#)

Useful links: [Binary Installers](#) | [Source Repository](#) | [Issues & Ideas](#) | [Q&A Support](#) | [Mailing List](#)


pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.



Getting started

New to pandas? Check out the getting started guides. They contain an introduction to pandas' main concepts and links to additional tutorials.


[To the getting started guides](#)



User guide

The user guide provides in-depth information on the key concepts of pandas with useful background information and explanation.


[To the user guide](#)



API reference

The reference guide contains a detailed description of the pandas API. The reference describes how the methods work and which parameters can be used. It assumes that you have an understanding of the key concepts.

[To the reference guide](#)



Developer guide

Saw a typo in the documentation? Want to improve existing functionalities? The contributing guidelines will guide you through the process of improving pandas.

[To the development guide](#)

Reading in any textfile

```
1 import pandas as pd
2
3 FNAME = "http://www.stat.ucla.edu/~rgould/datasets/twins.dat"
4
5 df = pd.read_csv(FNAME, sep='\t')
```

- Documentation at <http://www.stat.ucla.edu/~rgould/datasets/twinsexplain.txt>

Inspecting the DataFrame

```
1 df.shape    # Dimensions
2 df.head()   # First 5 lines (by default)
3 df.tail(7)  # Last 7 lines
4 df.columns  # List of variables as list
5 df.describe() # Summary statistics
```

1. How many observations do you have?
2. How many variables do you have?
3. Which variables are numeric?
4. What is the mean of variable "DEDUC1"?

Slicing the DataFrame

```
1 # Selecting rows
2 df.loc[0]    # Row by index name
3 df.iloc[0]   # Row by row number
4 # Selecting columns
5 df.iloc[:, 5:7] # Column range by column indices
6 df["DEDUC1"]  # Column by column name
7 df[["AGE", "LHRWAGEH"]] # Columns by list of names
8 # Selecting values
9 df.iloc[18, 2]
10 df.loc[18, "AGE"]
11 df["AGE"].iloc[18]
```

Slicing the DataFrame

```
1 # Selecting rows
2 df.loc[0]    # Row by index name
3 df.iloc[0]   # Row by row number
4 # Selecting columns
5 df.iloc[:, 5:7] # Column range by column indices
6 df["DEDUC1"]  # Column by column name
7 df[["AGE", "LHRWAGEH"]] # Columns by list of names
8 # Selecting values
9 df.iloc[18, 2]
10 df.loc[18, "AGE"]
11 df["AGE"].iloc[18]
```

1. What is the 6th entry of the 5th column?
2. What is the 5th entry of column "DTEN"?
3. What is the last entry of column "LHRWAGEL"?

Understanding dtypes

```
1 df.info()
```

Understanding dtypes

```
1 df.info()
```

Pandas	Python	Purpose
object	unicode	Text
int64	int	Integers
float64	float	Floating numbers
bool	bool	True and False values
datetime64		Date and time values
timedelta[ns]		Differences between two datetimes
category		Finite list of text values

Changing dtypes

```
1 df["DMARRIED"] = df["DMARRIED"].astype(bool)
2 df["WHITEH"] = df["WHITEH"].astype("category")
3 df["LHRWAGEH"] = pd.to_numeric(df["LHRWAGEH"], errors="coerce")
```

Optimising dtypes

```
1 df.info(memory_usage=True)
```

Optimising dtypes

```
1 df.info(memory_usage=True)
```

```
1 bools = ['WHITEH', 'MALEH', 'WHITEL', 'MALEL']  
2 df[bools] = df[bools].astype(bool)  
3 df['DMARRIED'] = df['DMARRIED'].astype('int8')  
4 df.info(memory_usage=True)
```

Boolean indexing

```
1 df[df["AGE"] > 20]
2 df[(df["AGE"] > 20) & (df["WHITE"] == 1)]
3 df[~(df["AGE"] > 20)]
4 values = (20, 21, 22, 23)
5 df[df["AGE"].isin(values)]
```

Boolean indexing

```
1 df[df["AGE"] > 20]
2 df[(df["AGE"] > 20) & (df["WHITE"] == 1)]
3 df[~(df["AGE"] > 20)]
4 values = (20, 21, 22, 23)
5 df[df["AGE"].isin(values)]
```

1. How many observations have "WHITE" equal to 0?
2. How many observations have "WHITE" equal to 1 and "DEDUC1" unequal to 0?
3. In how many rows do the values for "WHITE" and "WHITE" differ?
4. What is the mean age of twins whose L-sibling is a non-white male? (Use "DMARRIED", "WHITE" and "MALE")

Aggregate data

```
1 df["WHITEH"].value_counts()
2 pd.crosstab(df["WHITEH"], df["WHITEH"])
3 df[["DEDUC2", "EDUCL"]].corr()
```

Aggregate data

```
1 df["WHITE"].value_counts()
2 pd.crosstab(df["WHITE"], df["WHITE"])
3 df[["DEDUC2", "EDUCL"]].corr()
```

1. What is the most common value in "EDUCL"?
2. What is the most common combination of "MALEH" and "MALEL"?
3. What is the Spearman correlation between "EDUCH" and "EDUCL"? What is the Pearson correlation? (Check documentation!)

Manipulation

```
1  # Representation
2  df = df.sort_values(by='HRWAGEH')  # Sorting by column
3  df = df[sorted(df.columns)]  # Re-order columns alphabetically
4  # Mathematical operations
5  df['new'] = 9  # Add new column
6  df['AGETR'] = df['AGE']**3
7  df['combined'] = df['MALEH'] + df['EDUCH']
8  # Missing data
9  df["HRWAGEH_new"] = df["HRWAGEH"].fillna(0)  # Fill missings with 0
10 df = df.dropna(subset=["HRWAGEH"])  # Drop rows missing in "HRWAGEH"
```

Grouping

```
1 grouped = df.groupby(['MALEH'])  
2 print(grouped['AGE'].mean())  
3 print(grouped['EDUCH'].agg(['mean', 'sum']))  
4 print(grouped[['EDUCH', 'AGE']].agg(['mean', 'std']))
```

Grouping

```
1 grouped = df.groupby(['MALEH'])
2 print(grouped['AGE'].mean())
3 print(grouped['EDUCH'].agg(['mean', 'sum']))
4 print(grouped[['EDUCH', 'AGE']].agg(['mean', 'std']))
```

→ Full list at [http:](http://pandas.pydata.org/pandas-docs/stable/getting_started/basics.html#descriptive-statistics)

[//pandas.pydata.org/pandas-docs/stable/getting_started/basics.html#descriptive-statistics](http://pandas.pydata.org/pandas-docs/stable/getting_started/basics.html#descriptive-statistics)

- What is the "AGE" variance for "MALEL" == 0 individuals?
- What are the second and the third quartile of years of schooling for female L-siblings? (Use "EUDCL" and "MALEL" == 0)
- What is the average "AGE" for twins where both siblings are female?

Creating DataFrames from other objects

Creating Pandas DataFrames from Python Lists and Dictionaries

Row Oriented

```
sales = [{'account': 'Jones LLC', 'Jan': 150, 'Feb': 200, 'Mar': 140},  
        {'account': 'Alpha Co', 'Jan': 200, 'Feb': 210, 'Mar': 215},  
        {'account': 'Blue Inc', 'Jan': 50, 'Feb': 90, 'Mar': 95}]  
df = pd.DataFrame(sales)
```

default

	account	Jan	Feb	Mar
0	Jones LLC	150	200	140
1	Alpha Co	200	210	215
2	Blue Inc	50	90	95

from_records

List

```
sales = [('Jones LLC', 150, 200, 50),  
        ('Alpha Co', 200, 210, 90),  
        ('Blue Inc', 140, 215, 95)]  
labels = ['account', 'Jan', 'Feb', 'Mar']  
df = pd.DataFrame.from_records(sales, columns=labels)
```

Column Oriented

```
sales = {'account': ['Jones LLC', 'Alpha Co', 'Blue Inc'],  
        'Jan': [150, 200, 50],  
        'Feb': [200, 210, 90],  
        'Mar': [140, 215, 95]}  
df = pd.DataFrame.from_dict(sales)
```

from_dict

```
sales = [{'account', ['Jones LLC', 'Alpha Co', 'Blue Inc']},  
        {'Jan', [150, 200, 50]},  
        {'Feb', [200, 210, 90]},  
        {'Mar', [140, 215, 95]}]  
df = pd.DataFrame.from_items(sales)
```

from_items

When using a dictionary, column order is not preserved.
Explicitly order them:
`df = df[['account', 'Jan', 'Feb', 'Mar']]`

Practical Business Python - pbpython.com

Creating DataFrames from other objects, cont.

```
1 d = {'employee': ['Hannes', 'Fabiana', 'George', 'Olga'],
2      'group': ['Accounting', 'Engineering', 'Engineering', 'HR']}
3 df1 = pd.DataFrame.from_dict(d)
4 t = [('Hannes', 2004), ('Fabiana', 2008), ('George', 2012), ('Olga', 2014)]
5 df2 = pd.DataFrame.from_records(t, columns=["employee", "hire_date"])
```

Appending, Concatening and Merging

```
1 df3 = df1.append(df2)
2 df4 = pd.concat([df1, df2])
3
4 df5 = pd.concat([df1, df2], axis=1)
5 df6 = df1.merge(df2, left_on="employee", right_on="employee")
```

Appending, Concatening and Merging

```
1 df3 = df1.append(df2)
2 df4 = pd.concat([df1, df2])
3
4 df5 = pd.concat([df1, df2], axis=1)
5 df6 = df1.merge(df2, left_on="employee", right_on="employee")
```

- How do objects df3 and df4 differ?
- How do objects df5 and df6 differ?

What is pivot?

Pivot

df

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t



```
df.pivot(index='foo',  
          columns='bar',  
          values='baz')
```

bar	A	B	C
foo			
one	1	2	3
two	4	5	6

from: "Reshaping and Pivot Tables"

Pivoting and melting

```
1 pivoted = df6.pivot(index='employee', columns='group', values='hire_date')
2 reverse = (pivoted.reset_index()
3             .melt(id_vars="employee", value_name="hire_date"))
```

Pivoting and melting

```
1 pivoted = df6.pivot(index='employee', columns='group', values='hire_date')
2 reverse = (pivoted.reset_index()
3             .melt(id_vars="employee", value_name="hire_date"))
```

- How do you make reverse look like df6 again?

- https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html

- https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html

```
1 df.to_csv(FNAME, sep=";")
2 df.to_html(FNAME, decimal=".", justify="center")
3 df.to_stata(FNAME, write_index=False)
```
