Machine Learning with Python for Education and Personnel Economists

Michael E. Rose, PhD

Introduction

Who am I?

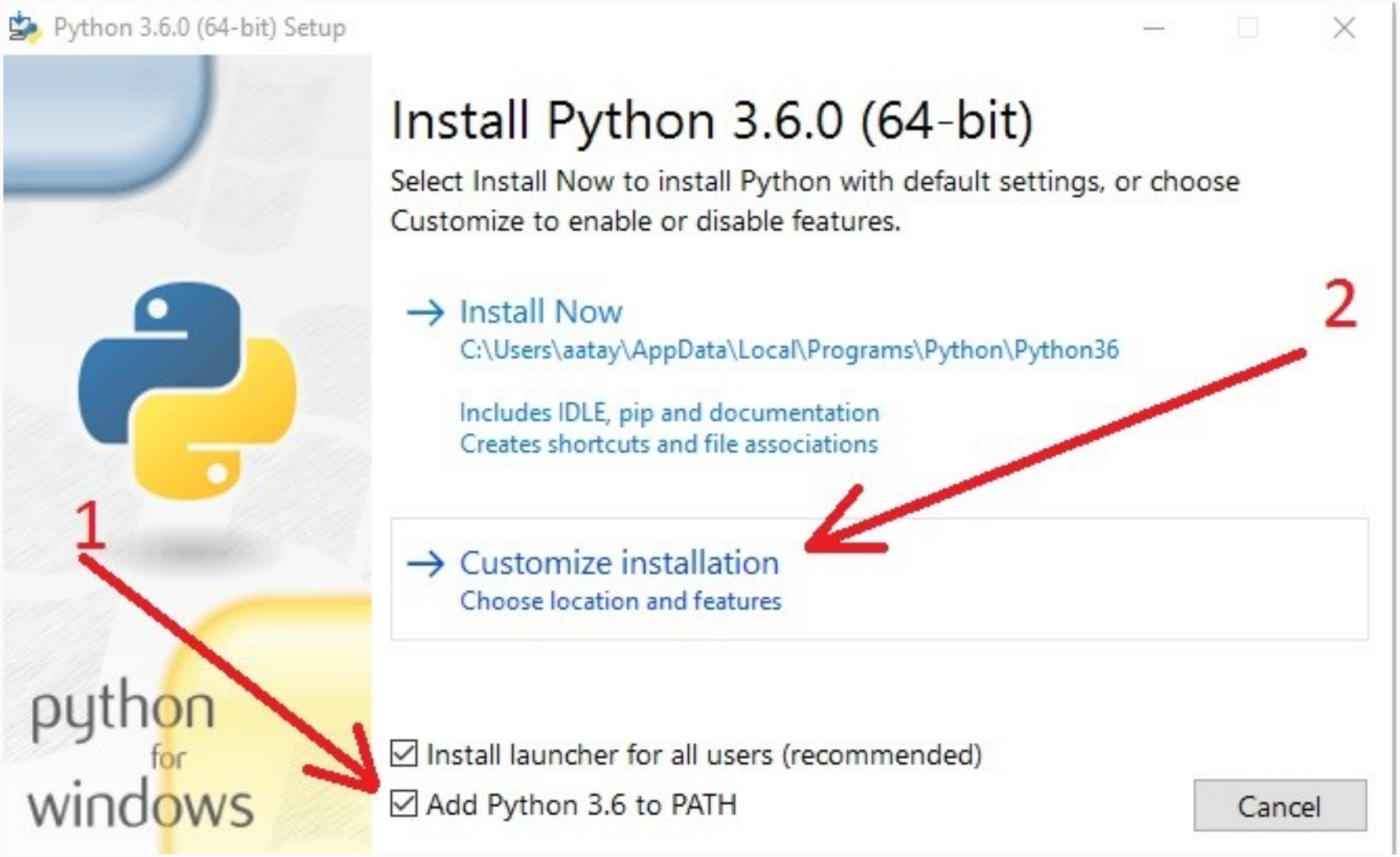
- Senior Research Fellow, Max Planck Institute for Innovation and Competition, Economics PhD from University of Cape Town
- Writing code since 8th grade
- Author of 3 open-source projects: scholarmetrics, pybliometrics, sosia
- Teaching experience:
  - Python, Big Data and Machine Learning for Economists @ 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

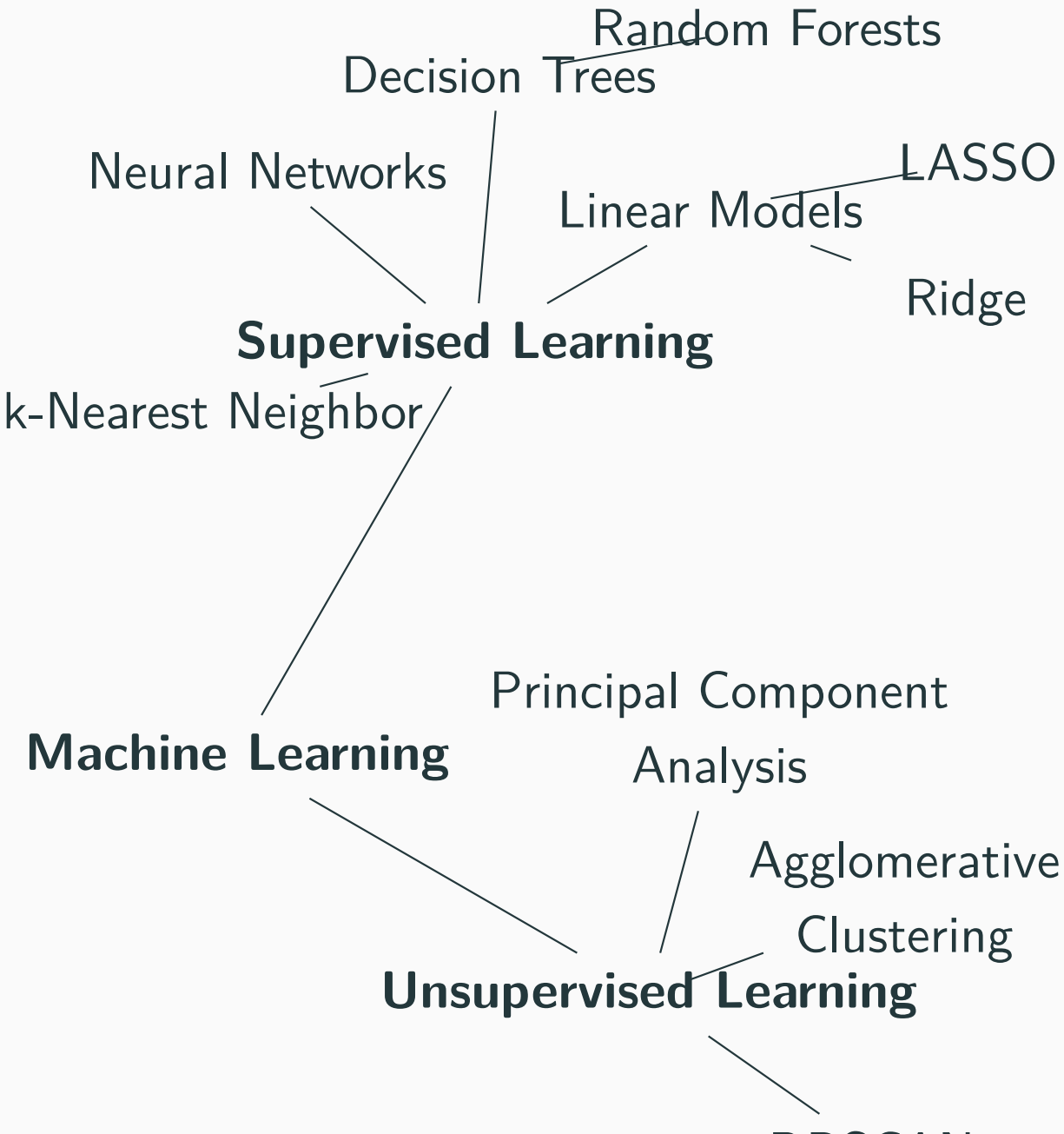
https://www.python.org/downloads/



Course content

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

Course Content, cont.



Course Design

- 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?

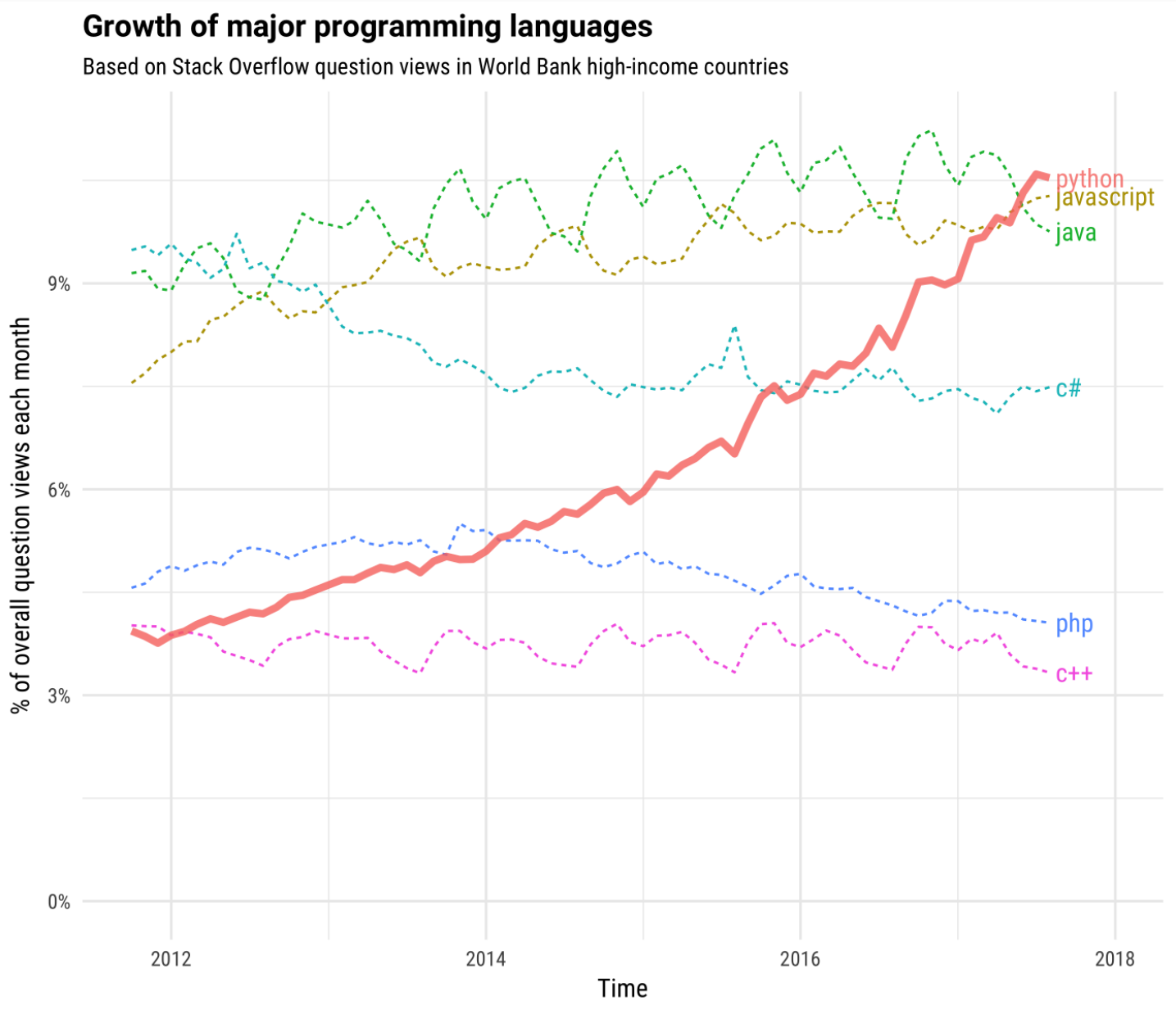
- Interpreted, high-level, general-purpose programming language
- Can be object-oriented, imperative, functional and procedural
- Free (= no licenses)
- Large (= support and many packages)
- Centralized development
- Very good first language

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

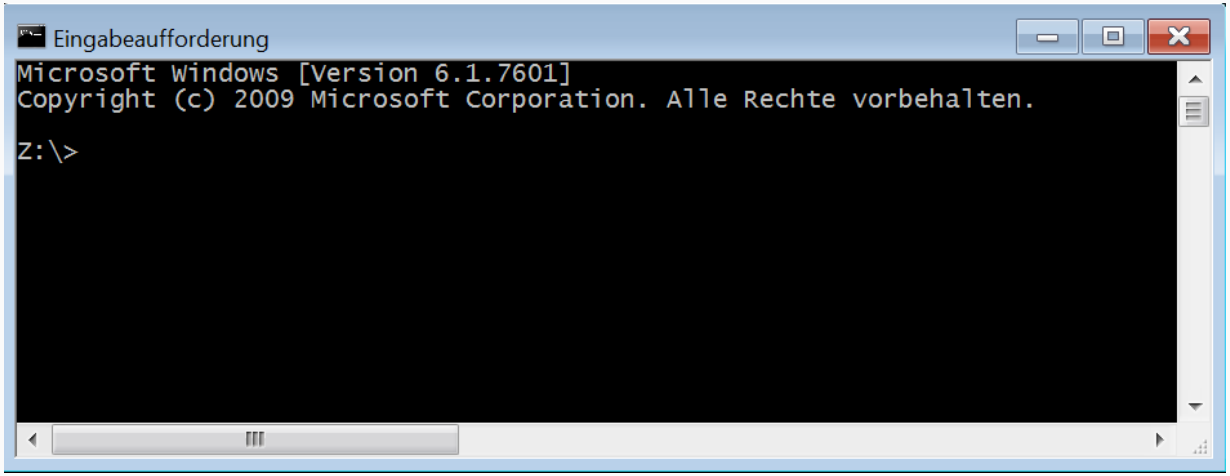
There should be one– and preferably only one –obvious way to do it.
Although that way may not be obvious at first unless you're Dutch. (Tim Peters - The Zen of Python)

Python is popular and increasing in popularity

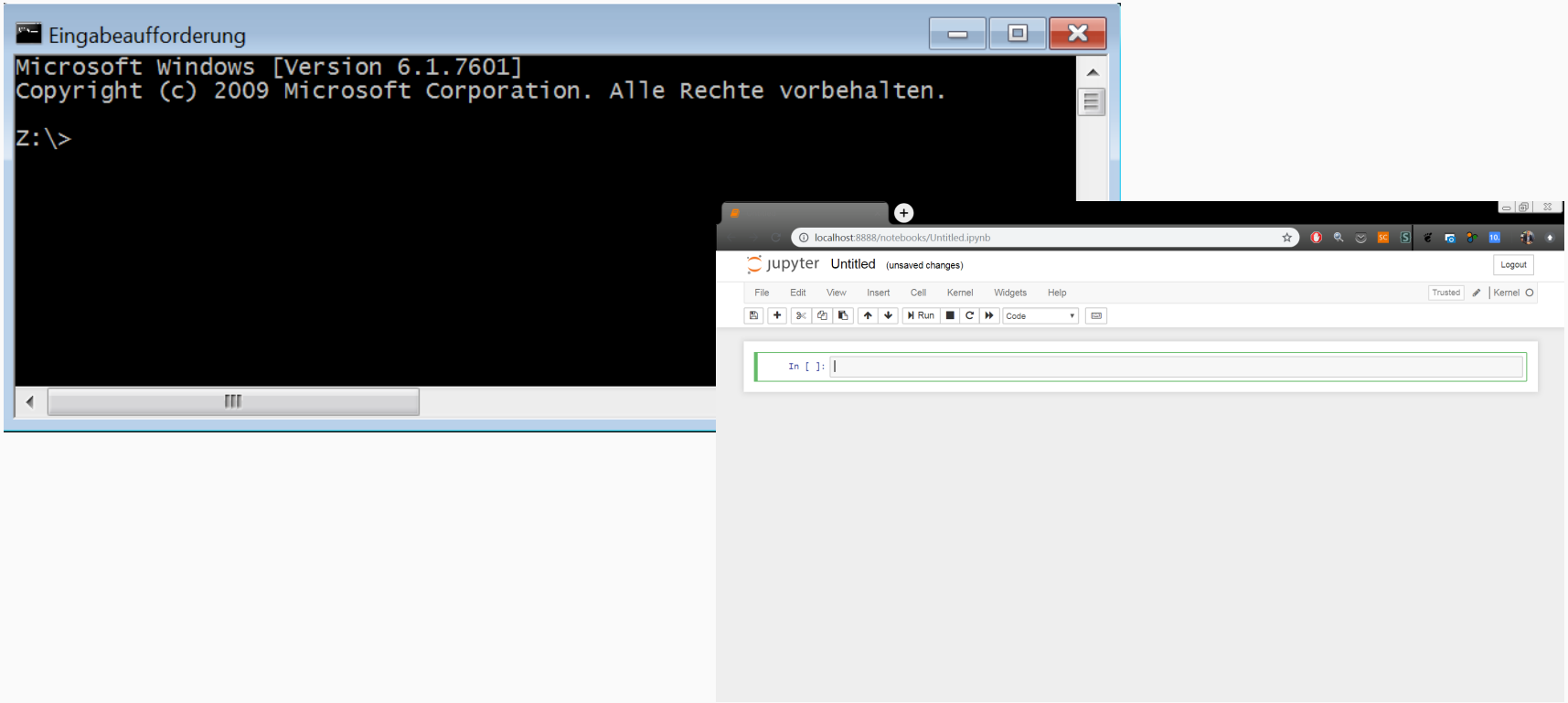


Dave Robinson, "The Incredible Growth of Python", Sep 6th 2017

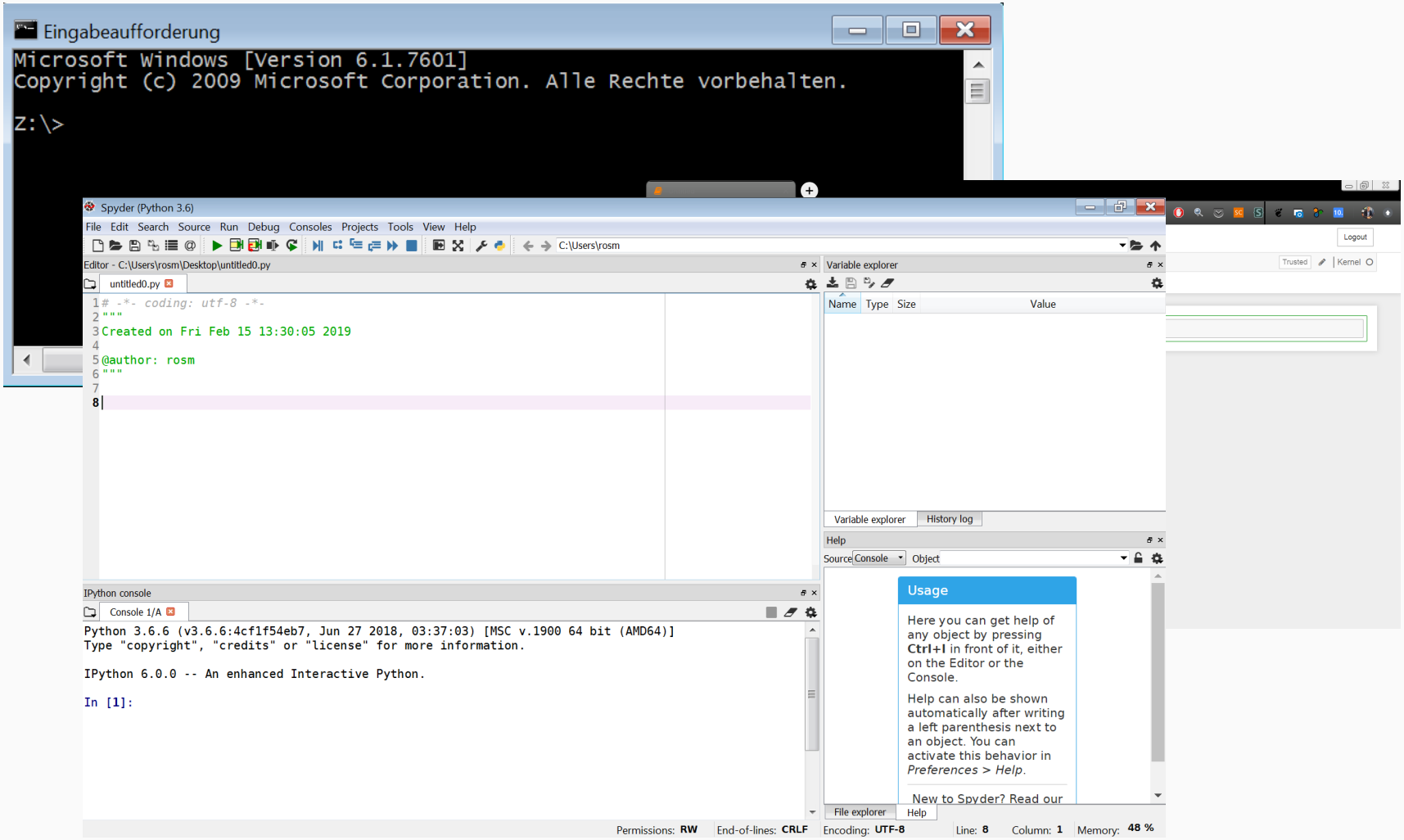
How to use Python?



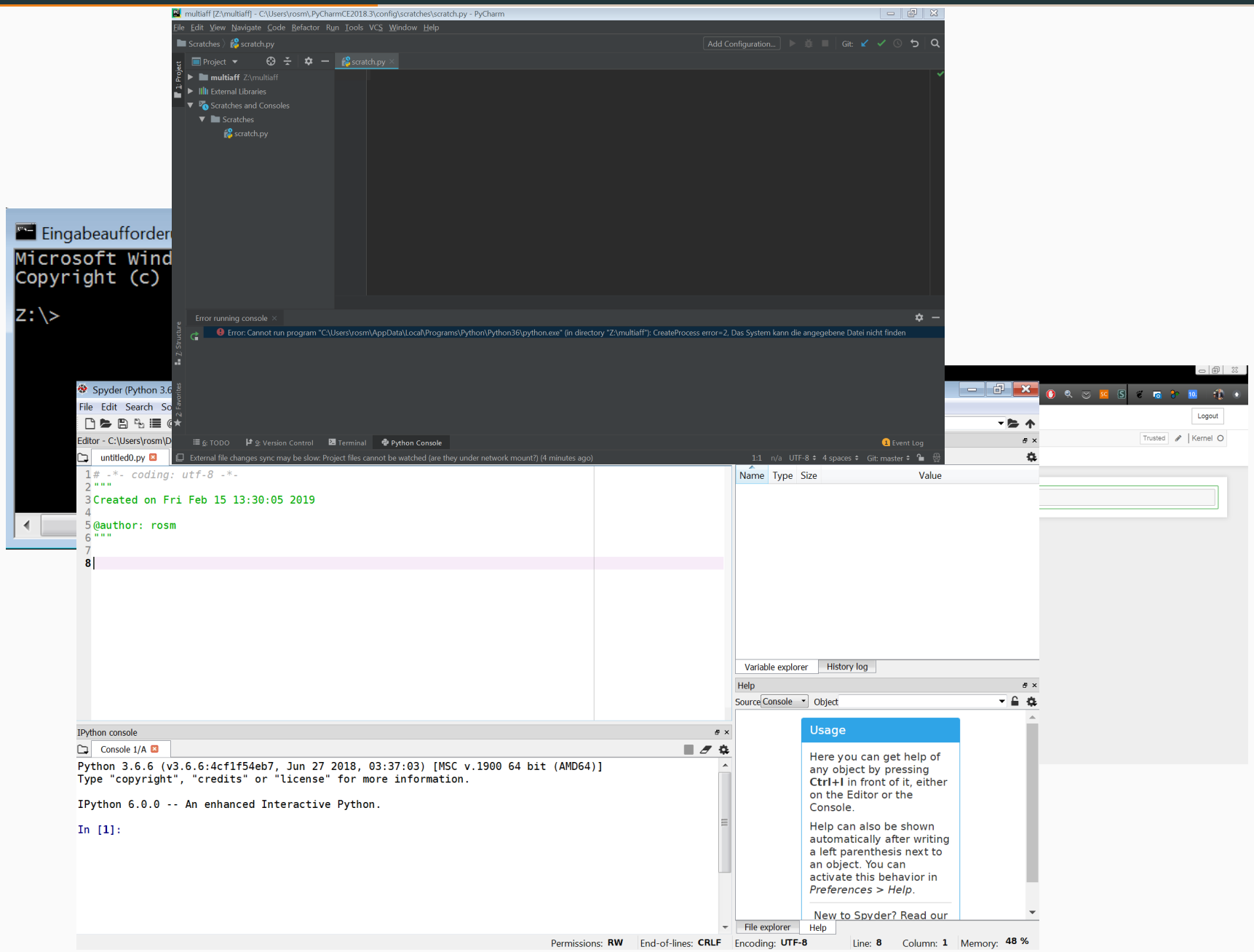
How to use Python?



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

Terminal/Console

- 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)

[1] Windows users: make sure Python is in the environment paths
[2] Windows users: make sure pip is in the environment paths

Terminal/Console

- 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)

```
pip install pandas
pip install matplotlib
pip install numpy
pip install pandas --upgrade
```

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

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

Jupyter Notebook

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

Jupyter Notebook

- 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

Jupyter Notebook, cont.

Type in cell

```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-10, 10, 100)
y = np.sin(x)
plt.plot(x, y, marker="x")
```

Recap Python basics

Recap Python basics

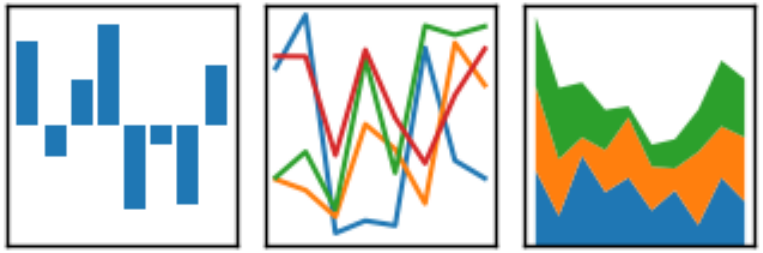
What matters in Python?

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

Pandas

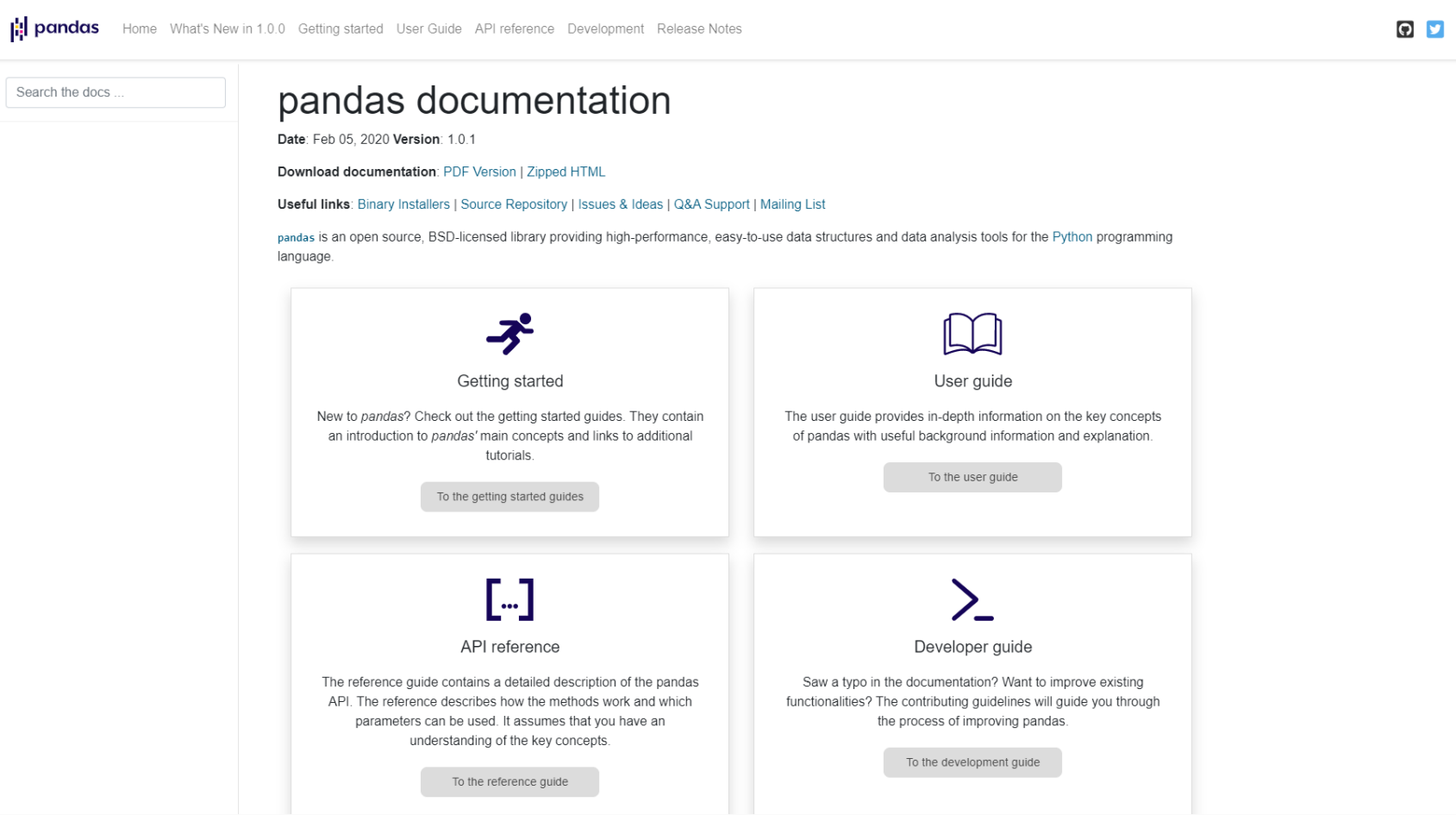
pandas

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



pandas: a library for data manipulation

- Documentation:
  http://pandas.pydata.org/pandas-docs/stable/



Reading in any textfile

```python
import pandas as pd

FNAME = "http://www.stat.ucla.edu/~rgould/datasets/twins.dat"

df = pd.read_csv(FNAME, sep='\t')
```

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

Inspecting the DataFrame

```
df.shape  # Dimensions
df.head()  # First 5 lines (by default)
df.tail(7)  # Last 7 lines
df.columns  # List of variables as list
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

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

Slicing the DataFrame

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

```
df.info()
```

Understanding dtypes

```
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

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

Optimising dtypes

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

Optimising dtypes

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

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

Boolean indexing

```
df[df["AGE"] > 20]
df[(df["AGE"] > 20) & (df["WHITEL"] == 1)]
df[~(df["AGE"] > 20)]
values = (20, 21, 22, 23)
df[df["AGE"].isin(values)]
```

Boolean indexing

```
df[df["AGE"] > 20]
df[(df["AGE"] > 20) & (df["WHITEL"] == 1)]
df[~(df["AGE"] > 20)]
values = (20, 21, 22, 23)
df[df["AGE"].isin(values)]
```

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

Aggregate data

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

Aggregate data

```
df["WHITEL"].value_counts()
pd.crosstab(df["WHITEH"], df["WHITEL"])
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

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

Grouping

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

Grouping

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

→ Full list at 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

Dictionary

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

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

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

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)
```

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

from_records

```
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

Practical Business Python - pbpython.com

Creating DataFrames from other objects, cont.

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

Appending, Concatening and Merging

```
df3 = df1.append(df2)
df4 = pd.concat([df1, df2])

df5 = pd.concat([df1, df2], axis=1)
df6 = df1.merge(df2, left_on="employee", right_on="employee")
```

Appending, Concatening and Merging

```
df3 = df1.append(df2)
df4 = pd.concat([df1, df2])

df5 = pd.concat([df1, df2], axis=1)
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

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

Pivoting and melting

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

- How do you make `reverse` look like `df6` again?

Output

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

Output

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

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