Python Programming and Machine Learning for Economists (August 2022)

Michael E. Rose, PhD

Introduction

Who am I?

- Senior Research Fellow, Max Planck Institute for Innovation and Competition, PhD in Econ (University of Cape Town)
- Writing code since 8th grade
- Author of 3 open-source projects: `pybliometrics`, `sosia`, `scholarmetrics`
- Teaching experience:
  - *This course* @ Kiel Institute for the World Economy (ASP), University of Zurich, ifo Institute Munich, LMU Munich, Scheller College of Business at Georgia Tech, TU Munich
  - 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?
- Who is more in control, your computer or you?

Course content

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

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

Exercises (= mini projects)

- Difficulty increases as the course progresses

Data sets
in tutorials

Data sets in
the wild

- Your grades depend on the exercises of days 3, 4 and 5
- The exercise on the 2nd day is optional, but recommend to all newbies

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 PyCharm
  6. Understand and use version control and use git
- Machine Learning
  1. Apply simple Neural Networks, clustering algorithms and Principal Component Analysis
  2. Interpret and evaluate any machine learning application
  3. Teach yourself how to apply machine learning algorithms we don't speak about

Required Readings

- Shapiro, J. and M. Gentzkow: "Code and Data for the Social Sciences: A Practitioners Guide" - *Short paper on project management by Economists, read it all today*
- Athey, S. and G. Imbens (ARE 2019): "Machine Learning Methods That Economists Should Know About" - *Well-written overview that introduces all the technical terms for meachine learning, read it until 3rd day*
- Gentzkow, M., B. Kelly and M. Taddy (JEL 2019): "Text as Data" - *Well-written introduction to language processing, read it until last day*

How to use Python

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

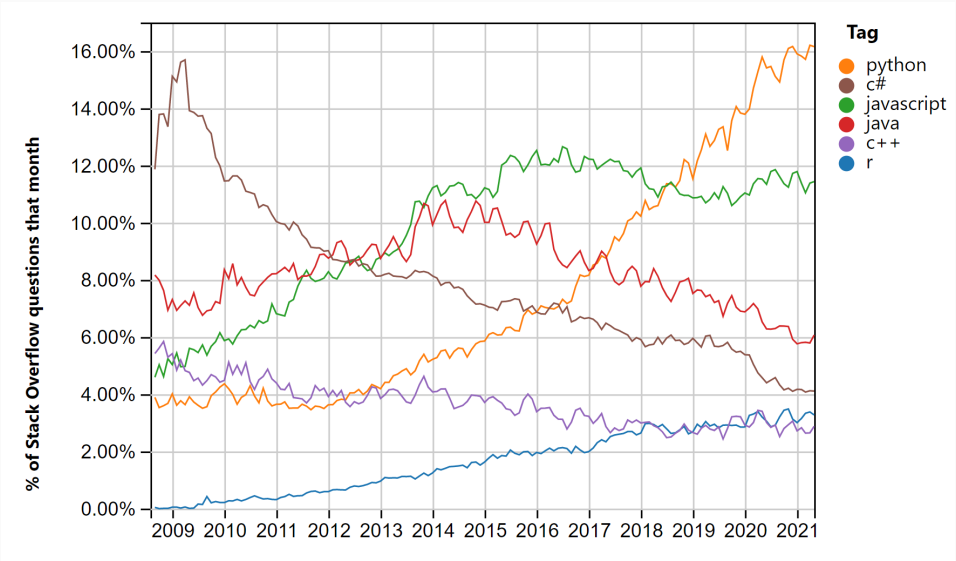
There should be one– and preferably only one –obvious way to do it. Although that way may not be obvious at first unless you're Dutch. (Tim Peters - The Zen of Python)

Credit where Credit is due

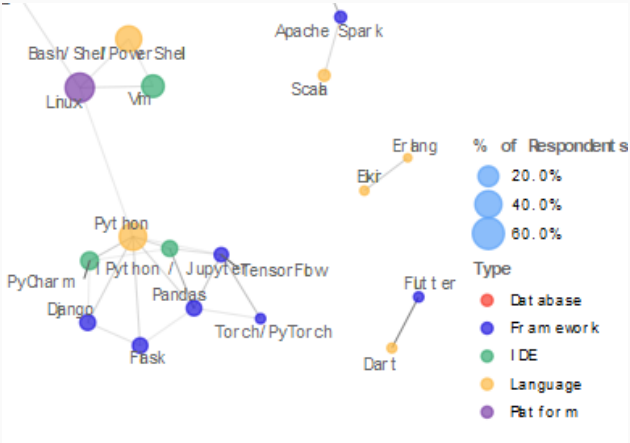
- Guido van Rossum
  created Python in his Christmas holidays 1989 as
  > *"a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus)."*

- Since 2019 5-member steering committee at the Python Foundation heads the development of Python

Python is popular and increasing in popularity



Python's local technology cluster



StackOverflow.com: "Developer Survey Results 2019"

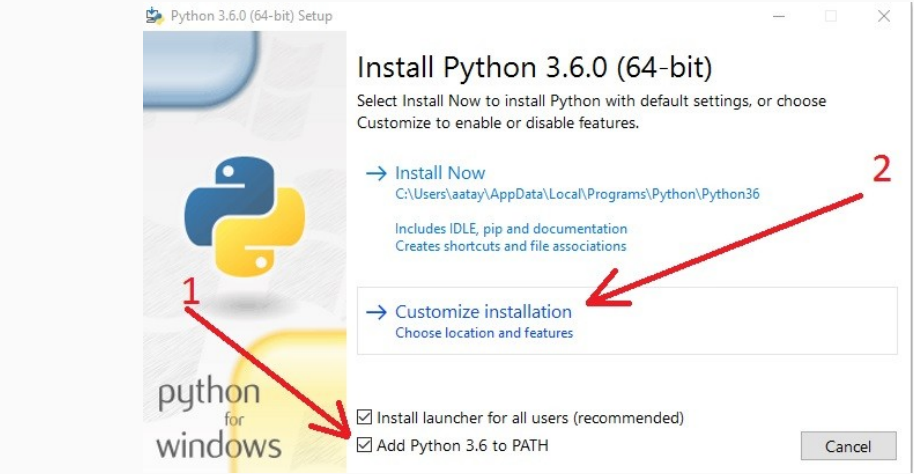
Why I discourage anaconda

- packages provided by anaconda need to be installed with `conda install` (they will ONLY be in the conda environment)
- Main difference in the past: `conda` used to be a better package manager than `pip`
- packages part of conda might be outdated
- Overkill/Unnecessary software (RStudio)
- Jupyter and spyder run without anaconda as well
- Actually not *that* popular: 19% of Python installations via Anaconda[1]

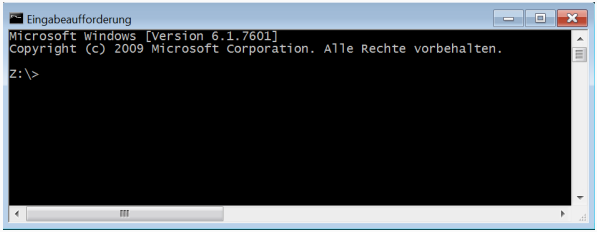
[1] Python Developers Survey 2020 Results

Installing Python and pip

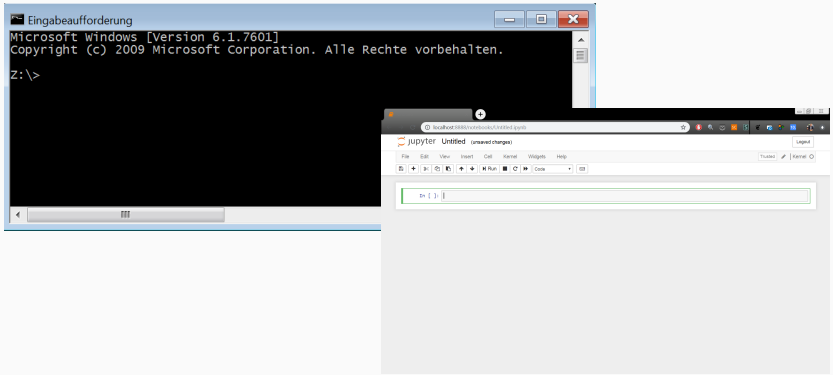
https://www.python.org/downloads/



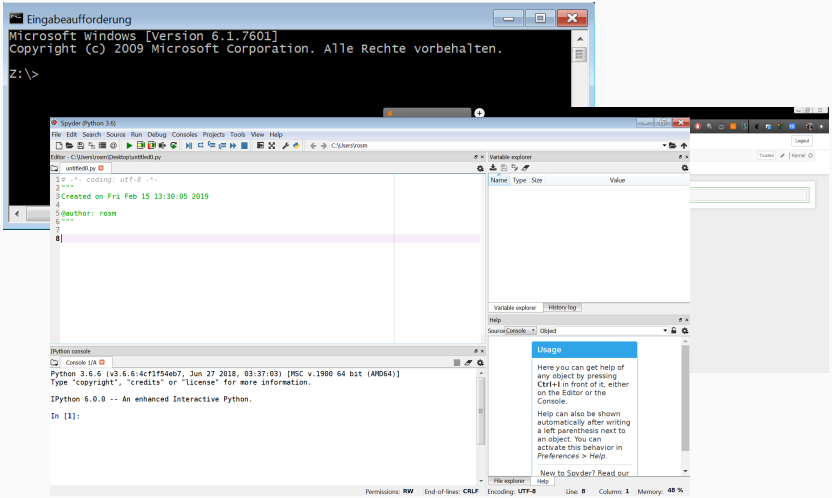
Different ways to use Python



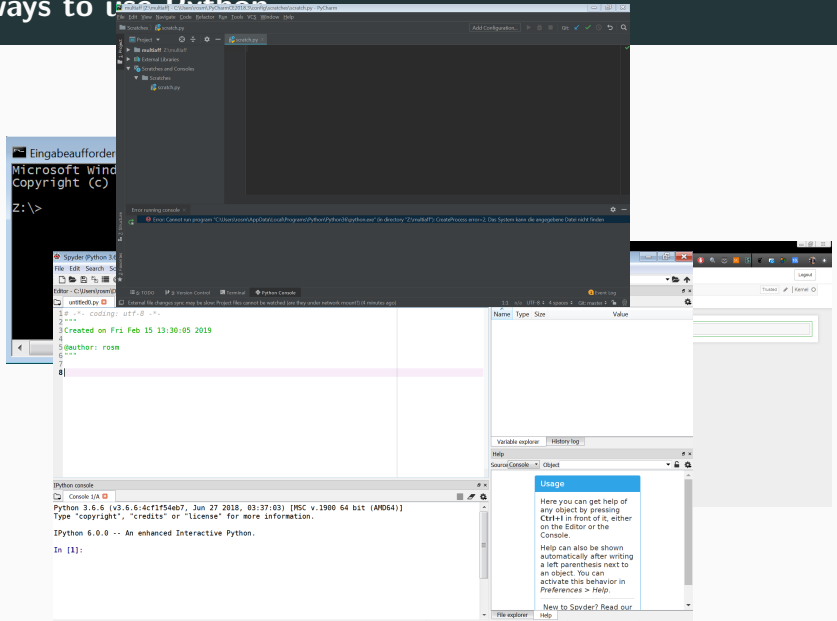
Different ways to use Python



Different ways to use Python



Different ways to use Python



Terminal/Console

- Console uses DOS language (Windows) or `shell` and `bash` (Linux and Apple)
- Starts python environment, Jupyter, and executes scripts

Terminal/Console

- >_ Console uses DOS language (Windows) or `shell` and `bash` (Linux and macOS)
- >_ Starts python environment, Jupyter, and executes scripts
- >_ Install packages here:
  - Windows: `python -m pip install pandas seaborn`
  - Linux/macOS: `python3 -m pip install pandas seaborn`
- >_ Shortcut (which is not platform-independent)
  - Windows: `pip install pandas seaborn`
  - Linux/macOS: `pip3 install pandas seaborn`

Jupyter Notebook on your computer

- Create a folder for this course and navigate there in your terminal (alternatively, open the "PowerShell" via context menu after ⇧+rightclick)

Jupyter Notebook on your computer

- Create a folder for this course and navigate there in your terminal (alternatively, open the "PowerShell" via context menu after ⇧+rightclick)
- Install the jupyter notebook if necessary

```
python3 -m pip install notebook
jupyter notebook
```

- Your browser will fire up (i.e., you started your own server)

Jupyter Notebook on your computer

- Create a folder for this course and navigate there in your terminal (alternatively, open the "PowerShell" via context menu after ⇧+rightclick)
- Install the jupyter notebook if necessary

```
python3 -m pip install notebook
jupyter notebook
```

- Your browser will fire up (i.e., you started your own server)
- Click on New in the upper right corner to start a new notebook

Notebooks will be saved in the folder where you invoked the jupyter server

Jupyter notebook in the ☁

- colab.research.google.com: requires Google account; stores notebooks in your Drive; integrates with GitHub; potentially older packages
- kaggle.com/code: requires Kaggle account; allows for R as well
- mybinder.org: requires GitHub account; builds from a GitHub repository

Recap some Python basics

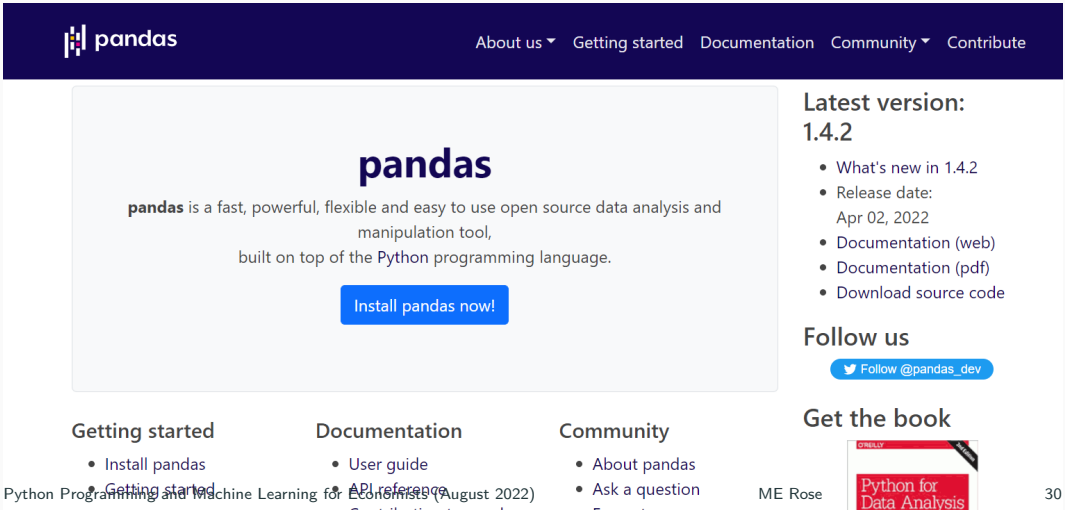
What matters in Python?

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

Pandas

pandas: the library for data manipulation

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



Let's start with a dataset on twins...

```
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

pandas functionality relevant for the course

- 10 minutes to pandas
- IO tools (text, CSV, HDF5, ...)
- Indexing and selecting data
- Reshaping and pivot tables
- Working with missing data
- Computational tools

Let's inspect our data

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

```python
# Selecting columns
df["DEDUC1"]  # Column by column name
df[["AGE", "LHRWAGEH"]]  # Columns by list of column names
df.iloc[:, 5:7]  # Column range by column indices

# Selecting rows
df.loc[0]  # Row by index name (also accepts lists)
df.iloc[0]  # Row by row number (also accepts lists)

# Selecting values
df.loc[18, "AGE"]  # Name of row and column
df.iloc[18, 2]  # Index of row and column
```

Slicing the DataFrame

```python
# Selecting columns
df["DEDUC1"]  # Column by column name
df[["AGE", "LHRWAGEH"]]  # Columns by list of column names
df.iloc[:, 5:7]  # Column range by column indices

# Selecting rows
df.loc[0]  # Row by index name (also accepts lists)
df.iloc[0]  # Row by row number (also accepts lists)

# Selecting values
df.loc[18, "AGE"]  # Name of row and column
df.iloc[18, 2]  # Index of row and column
```

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 & False values
datetime64		Date and time values
timedelta[ns]		Differences between two datetimes
category		Finite list of text values

Changing dtypes

```
df["WHITEH"] = df["WHITEH"].astype(bool)
df["DMARRIED"] = df["DMARRIED"].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]  # One condition
df[(df["AGE"] > 20) & (df["WHITEL"] == 1)]  # Multiple conditions
df[~(df["AGE"] > 20)]  # Tilde inverses boolean
values = (20, 21, 22, 23)
df[df["AGE"].isin(values)]  # Select specific values
```

Boolean indexing

```
df[df["AGE"] > 20]  # One condition
df[(df["AGE"] > 20) & (df["WHITEL"] == 1)]  # Multiple conditions
df[~(df["AGE"] > 20)]  # Tilde inverses boolean
values = (20, 21, 22, 23)
df[df["AGE"].isin(values)]  # Select specific 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 with either 12 or 14 years of education? (Use "WHITEL", "MALEL" and "EDUCHL",)

Aggregate data

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

Aggregate data

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

1. What is the most common value in "EDUCL"?
2. What is the most common combination of "MALEH" and "MALEL"?

Manipulation

```python
# Representation
df = df.sort_values(by='HRWAGEH')  # Sorting by column
df = df[sorted(df.columns)]  # Re-order columns alphabetically
# Work on columns
df = df.drop('AGESQ', axis=1)  # Drop a column
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']))
```

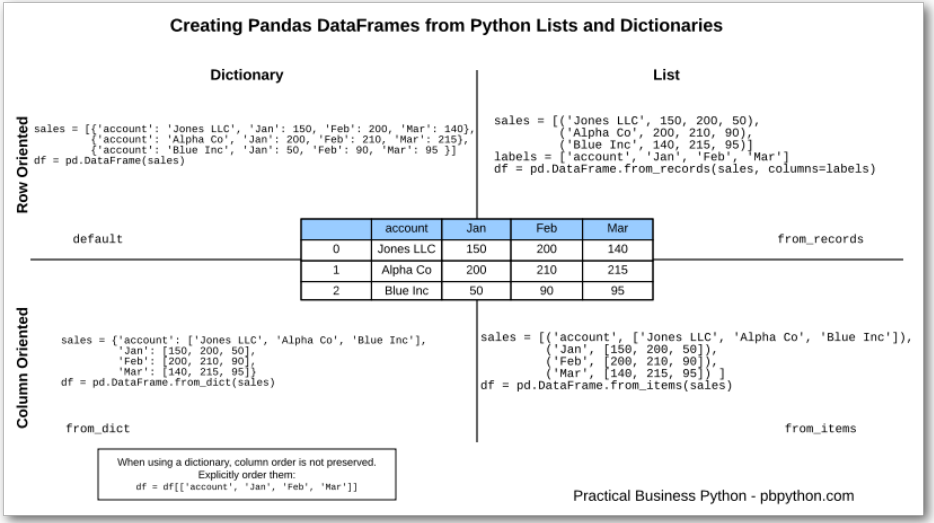
Grouping

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

→ Full list at https://pandas.pydata.org/pandas-docs/stable/user_guide/groupby.html#aggregation

- 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



To become a Master...

- 10 minutes to pandas
- Wes McKinney: "Python for Data Analysis. Data Wrangling with Pandas, NumPy, and IPython", O'Reilly (2017)
- Fabio Nelli: "Python Data Analytics. Data Analysis and Science Using Pandas, matplotlib, and the Python Programming Language", Apress (2015)

Plotting w/ pandas (matplotlib), and w/ seaborn

Visualization with pandas

- Straightforward plotting as DataFrame methods for all kinds: barplots, areas, histograms, violin plots, timeseries, etc.:
  https://pandas.pydata.org/pandas-docs/stable/visualization.html
- Has `matplotlib` under the hood - for aesthetics
  `import matplotlib.pyplot as plt`
- Set global styles with `plt.style.use('<style>')` (list all styles with `plt.style.available`)

! Beware: Have DataFrame in correct format (long vs. wide)

Statistical plotting with seaborn

- seaborn: wrapper for `matplotlib`, optimized for quick statistical plotting: Error bars, distributions, regressions, etc.

- Use seaborn's toy datasets using `.load_dataset()`

👉 If downloading example datasets via `.load_dataset()` doesn't work, get them from github.com/mwaskom/seaborn-data and store them in `~./seaborn-data/`

Seaborn's plotting philosophy

- Statistical relation between numeric values?
  - ➔ `relplot()` for Scatter and Line (→ Documentation)
- Categorical data?
  - ➔ `catplot()` for Scatter-like (Swarm and Strip), Distributions (Box, Violin, Boxen) and Estimations (Point, Bar, Count) (→ Documentation)
- Linear relationships?
  - ➔ `regplot()` (→ Documentation)

Pandas plotting vs. seaborn

- In Jupyter, remember to write and execute `%matplotlib inline` in first cell to show figures
- Use pandas when you do the aggregations yourself
- Use seaborn when you use raw data – seaborn will aggregate itself

Excourse: colormaps

List of named colors in matplotlib

Color maps in matplotlib

Color maps in seaborn

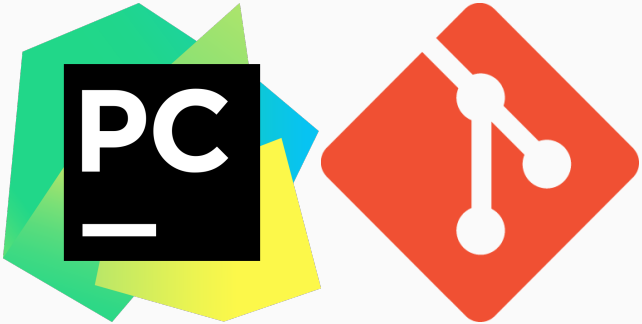
To become a Master...

- Fabio Nelli: "Python Data Analytics. Data Analysis and Science Using Pandas, matplotlib, and the Python Programming Language", Apress (2015)
- matplotlib Tutorials
- seaborn User guide and tutorial

Recap Day 1

- Use the Terminal/Console to install new packages, upgrade using `--upgrade` flag
- Consult the package's documentation for parameter names, defaults and examples
- Python is object-orientated: don't forget to reassign after working with an object

Project Management with PyCharm and git



Proper Data Management

- ... increasingly required by funders (as of 2021, ERC grant holders have to have a Research Data Management Plan in place)
- ... usually entails a backup system, maybe with versioning
- ... enables you to keep track of your progress
- ... facilitates working with others

Proper Data Management

- ... increasingly required by funders (as of 2021, ERC grant holders have to have a Research Data Management Plan in place)
- ... usually entails a backup system, maybe with versioning
- ... enables you to keep track of your progress
- ... facilitates working with others

! Remember: You are your first re-user of your data
  - Documentation
  - Accuracy
  - Replicability

Ten Simple Rules for Reproducible Computational Research

1. For Every Result, **Keep Track** of How It Was Produced
2. Avoid **Manual Data Manipulation** Steps
3. **Archive** the Exact Versions of All External Programs Used
4. **Version Control** All Custom Scripts
5. Record All **Intermediate Results**, When Possible in Standardized Formats
6. For Analyses That Include Randomness, Note Underlying **Random Seeds**
7. Always Store **Raw Data** behind Plots
8. Generate **Hierarchical Analysis Output**, Allowing Layers of Increasing Detail to Be Inspected
9. Connect Textual Statements to **Underlying Results**
10. Provide **Public Access** to Scripts, Runs, and Results

Geir K. Sandve et al. (2013): "Ten Simple Rules for Reproducible Computational Research", Plos ONE.

More control for users

- Show file endings - How?
- Show hidden files - How?

Simple rules for an Economist's project directory

- "Automate everything that can be automated."
- "Store code and data under version control."
- "Separate directories by function."
- "Separate files into inputs and outputs."
- "Manage tasks with a task management system."

Why PyCharm?

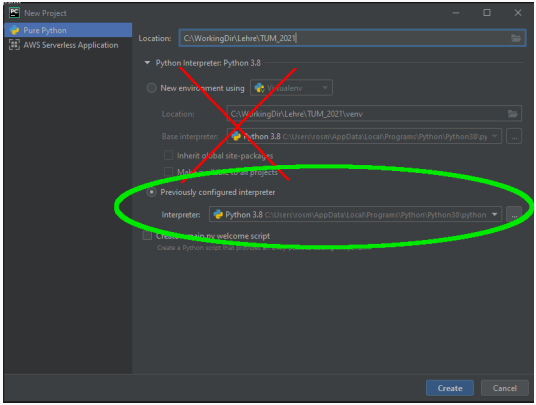
- Integrated Developer Environment (IDE), i.e. terminal, editor, object explorer, etc. in a single window
- Project-aware: Knows of usage of imported functions elsewhere etc.
- Integrates with version control systems and also Amazon Web Services (AWS)
- Community edition is free (→ Download)

🏆 Most used editor or IDE in 2020, with 33% of developers[2]

[2] Python Developers Survey 2020 Results

Starting a project in PyCharm

1. (Install and)Open PyCharm
2. In the Welcome screen, click on "Open" and open the folder where you saved your notebook yesterday
3. Do **NOT** create a new/virtual environment (*venv*), rather (set and)use the system interpreter(to your python installation)
4. `main.py` Welcome Script not necessary



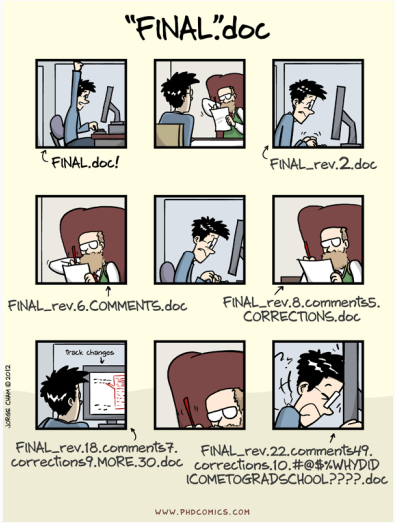
jetbrains.com/help/pycharm/creating-and-running-your-first-python-project.htmls

Why does git exist?

- Git protects yourself and others from yourself and others
- You can modify/change/break/improve your code and data, secure in the knowledge that you can not ruin your work too badly
- **No** commercial software is written without Version Control!
- Lots of open-source projects as well:
  - pandas, scikit-learn, seaborn, ggplot2, ⋯
- Very handy to compare recent changes against history
- Almost all Python developers use version control at least sometimes[3]

[3] Python Developers Survey 2020 Results

With git you *never* change the file name



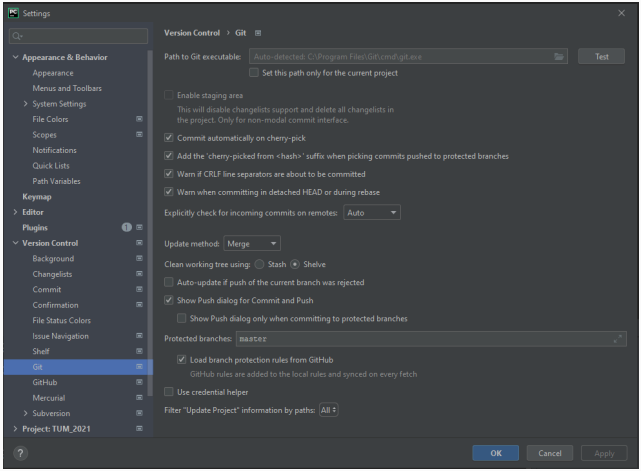
How does git work?

1. You tell git which files to keep track of ("checking-in")
2. ... eventually to store snapshots of changes of tracked files ("committing")
3. ... on top of previous commits ("repository")

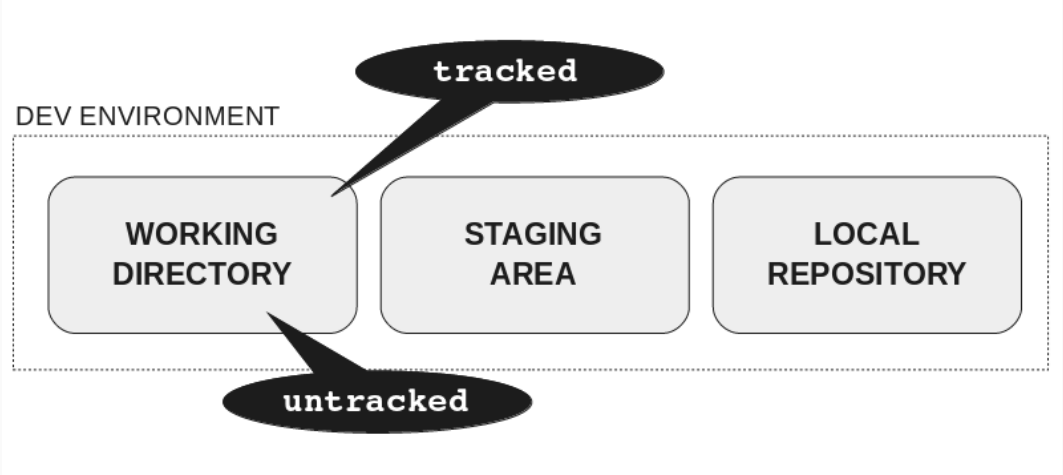
$\longrightarrow$ git manages changes to a project without overwriting any part of it

Configuring git in PyCharm

1. (Install git from git-scm.com/download)
2. File | Settings > Version Control > Git → Set "Path to Git executable" (often auto-deteced)
3. VCS | Enable Version Control Integration → select "Git"
4. Use green marker to open git dialogue



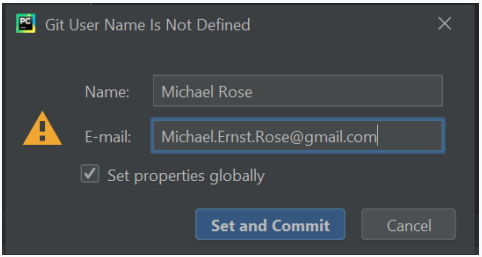
git's architecture



from: Rachel Carmena (2018): "How to teach Git"

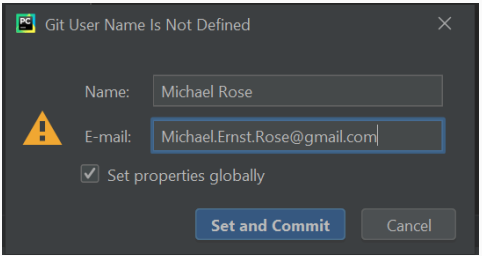
Telling git who you are

On first commit, PyCharm prompts for name and email address



Telling git who you are

On first commit, PyCharm prompts for name and email address



Alternatively, you may state your identity via the terminal:

```
$ git config --global user.name "<Your real name>"
$ git config --global user.email <Your real email address>
```

If you plan to use git outside of PyCharm also set the editor

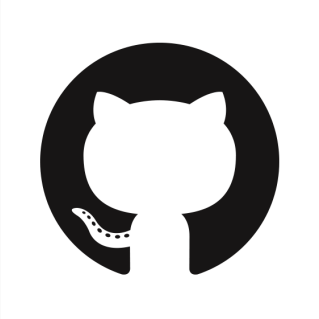
The `.gitignore` file

- Small file to specify files and folders you do not want to track → Documentation
  - PyCharm's `.idea` folder
  - temp files from Stata, Python, R, etc.
  - Windows' database files
- Works best with regex → Templates
- Hidden on *nix systems; show with ctrl + h

To become a Master...

- PyCharm's playlist Getting Started with PyCharm (13 videos)
- PyCharm's Knowledge Base

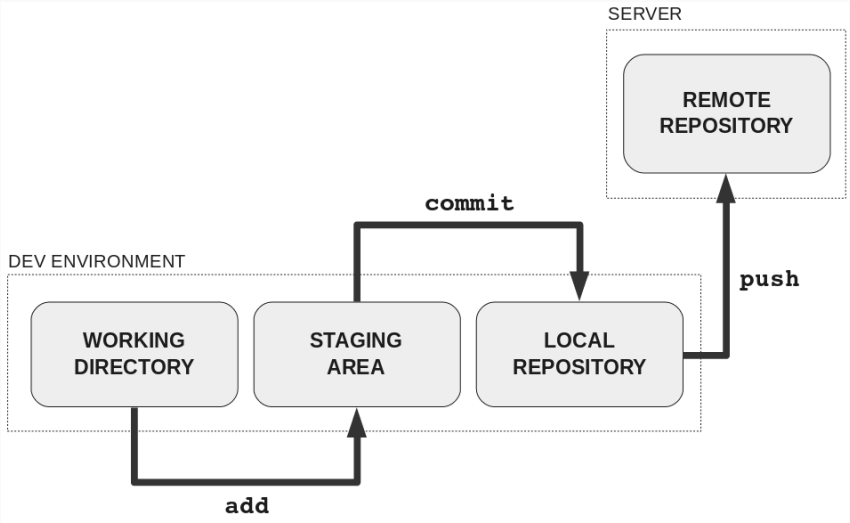
Collaborating with GitHub and/or GitLab



What's the difference?

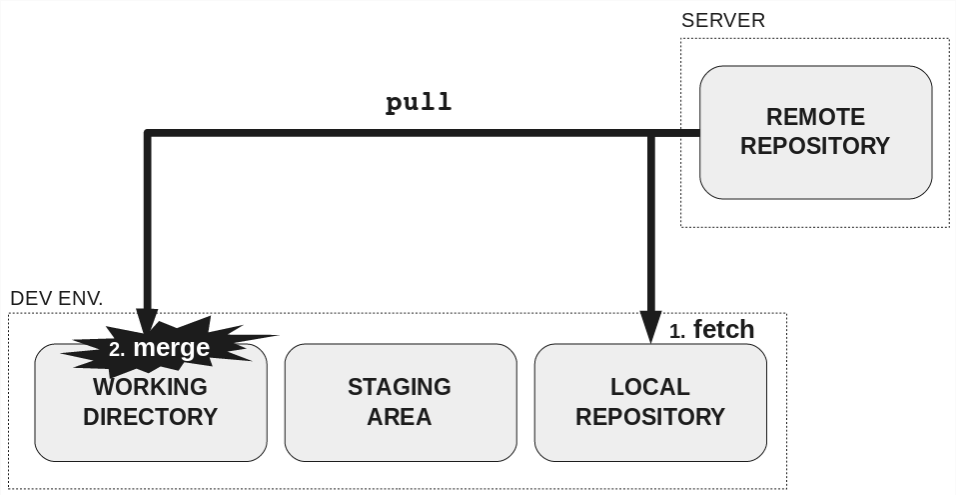
- git: Version control *on your machine*
- GitHub: Cloud storage accessible from git
- GitLab: GitHub for projects that require continuous integration (CI), i.e. web-apps

How do your changes make it to GitHub/GitLab?



from: Rachel Carmena (2018): "How to teach Git"

How do others' changes make it to your system?



from: Rachel Carmena (2018): "How to teach Git"

Configuring GitHub in PyCharm

1. File | Settings > Version Control > Git → check "Credential Helper"
2. File | Settings > Version Control > GitHub → Click "Add Account"
   - Create an account, or
   - Sign in

⚠ If in future your commits don't make it to GitHub, verify on this page that you're still connected to GitHub

---

If you plan to use GitHub outside PyCharm:

```
$ git config --global credential.helper cache
$ git config --global user.password "<Your GitHub password>"
```

Option 1: You have a local repo and want to have it on GitHub

1. Open PyCharm in the folder you want to have on GitHub
2. (Have at least one commit in repo)
3. Git | GitHub > Share Project on GitHub → Type repository name(and check Private)

👉 With GitLab this doesn't work (yet)

Option 2: You have a repo on GitHub/GitLab and want it locally ("cloning")

1. Create a (preferably private) repository on github.com (click "+" top right)
2. Open PyCharm anywhere
3. Either click on
   - VCS | Get from Version Control
   - Git | Clone...
4. In the new window, select "GitHub <your account name>" on the left
5. From the list of repos, select the new one; then on the bottom set the location

👉 PyCharm creates a new folder, turns it into a projects and establishes the connection to GitHub

👉 Do not attempt to clone a remote repo into another local one!

GitHub

- Repos have unlimited space but no file may be larger than 100MB
- Stars a repo on GitHub to save to your favorites and to say Thank you
- Get Pro benefits for free via GitHub Student Developer Pack) (Added benefit: GitHub hosts a simple private webpage)

To become a Master...

- GitHub's Learning Lab

Debugging

Bad things that can happen to your code

- Syntax Errors: Prevent your code from running (i.e. pre-runtime)
- Runtime Error: Occur during runtime (Exception)
- Semantic Error: Code runs, but not the way you like (Bugs)

Bad things that can happen to your code

- Syntax Errors: Prevent your code from running (i.e. pre-runtime)
- Runtime Error: Occur during runtime (Exception)
- Semantic Error: Code runs, but not the way you like (Bugs)

? Which one of these is a syntax error, which one is a bug, and which one will throw an exception?

1. Attempting to divide by 0
2. Not closing a parenthesis
3. Not dividing by 100 when computing a percentage

Avoid bugs in the first place

- Write easy code
- Experiment to check your hypotheses
  - `print()` objects to see what they contain
  - `print(type())` objects to see what they are
- Scaffolding: Write, check, repeat (Get something working and keep it working)
- Think formally (unlike in natural languages)
  - No ambiguity
  - Less redundancy
  - Always literal

Avoid bugs in the first place

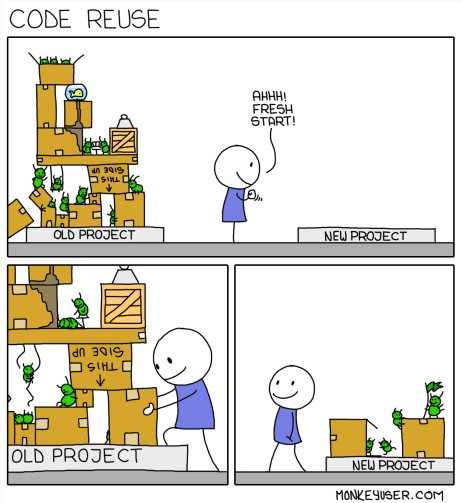
- Write easy code
- Experiment to check your hypotheses
  - `print()` objects to see what they contain
  - `print(type())` objects to see what they are
- Scaffolding: Write, check, repeat (Get something working and keep it working)
- Think formally (unlike in natural languages)
  - No ambiguity
  - Less redundancy
  - Always literal

⌨ The problem always sits behind the keyboard

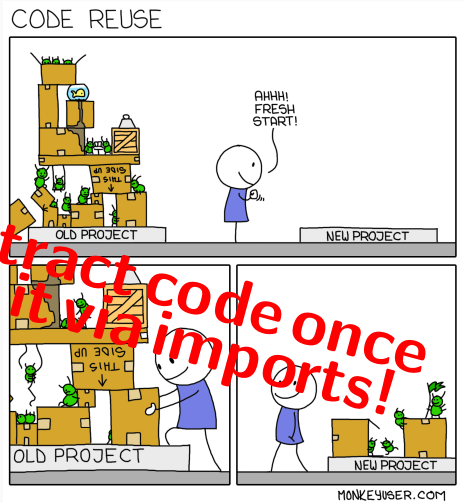
How to hunt down the bug

- You will spend most of the time debugging
- It's detective work: Where does the bug come from, how to fix it w/o breaking other things
- Tracebacks help you: What kind of error & where (approximately)

Avoid reusing bad code



Avoid reusing bad code



Write abstract code once and reuse it via imports!

Make use of tracebacks!

```
Traceback (most recent call last):
  File "./test.py", line 21, in <module>
    main()
  File "./test.py", line 14, in main
    data=tips, legend=False)
  File "/usr/local/lib/python3.6/dist-packages/seaborn/relational.py", line 1613, in relplot
    **plot_kws)
  File "/usr/local/lib/python3.6/dist-packages/matplotlib/__init__.py", line 1810, in inner
    return func(ax, *args, **kwargs)
  File "/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_axes.py", line 4300, in scatter
    collection.update(kwargs)
  File "/usr/local/lib/python3.6/dist-packages/matplotlib/artist.py", line 916, in update
    ret = [_update_property(self, k, v) for k, v in props.items()]
  File "/usr/local/lib/python3.6/dist-packages/matplotlib/artist.py", line 916, in <listcomp>
    ret = [_update_property(self, k, v) for k, v in props.items()]
  File "/usr/local/lib/python3.6/dist-packages/matplotlib/artist.py", line 912, in _update_property
    raise AttributeError('Unknown property %s' % k)
AttributeError: Unknown property xcol
```

Inspecting the object

```
1 my_list = {'syntax': 10, 'runtime': 99}
2 print(type(my_list))
```

- What is the type of object `my_list`?

Checking the version

Every decent package has a magic attribute `.__version__`:

```
import pandas as pd

pd.__version__
```

Useful to check whether your version is outdated; assure you're on the latest version before bothering developers

Know your error I

```
x = "9"
y = 1
z = x + y
```

Know your error I

```
1 x = "9"
2 y = 1
3 z = x + y
```

- `TypeError`: you try to combine two objects that are not compatible

Know your error II

```
currencies = ["dollar", "euro"]
print(currency)
```

Know your error II

```
1 currencies = ["dollar", "euro"]
2 print(currency)
```

- `NameError`: you refer to an object that does not exist

Know your error III

```
int("9.0")
```

Know your error III

```
1 int("9.0")
```

- `ValueError`: the value you passed to a parameter does not pass the function's limitations on the value

Know your error IV

```
1 marks = [1, 1, 4]
2 print(marks[4])
```

Know your error IV

```
1 marks = [1, 1, 4]
2 print(marks[4])
```

- `IndexError`: you are referring to an element in a container that does not exist

Know your error V

```
capitals = {'ger': 'berlin', 'aut': 'vienna'}
print(capitals['fra'])
```

Know your error V

```
capitals = {'ger': 'berlin', 'aut': 'vienna'}
print(capitals['fra'])
```

- `KeyError`: you are referring to a key in a dict (or dict-like object) that does not exist

Know your error VI

```
1 my_list = "dbcea"
2 my_list.sort()
```

Know your error VI

```
1 my_list = "dbcea"
2 my_list.sort()
```

- `AttributeError`: what you want to do with an object is not possible (mostly: the object is not what you think it is)

Handling exceptions with try-except clauses

To find out how your objects look like *exactly* when code fails, use a try-except clause

```
    try:
        average = sum(a_list) / len(a_list)
    except ZeroDivisionError:
        print(a_list)
```

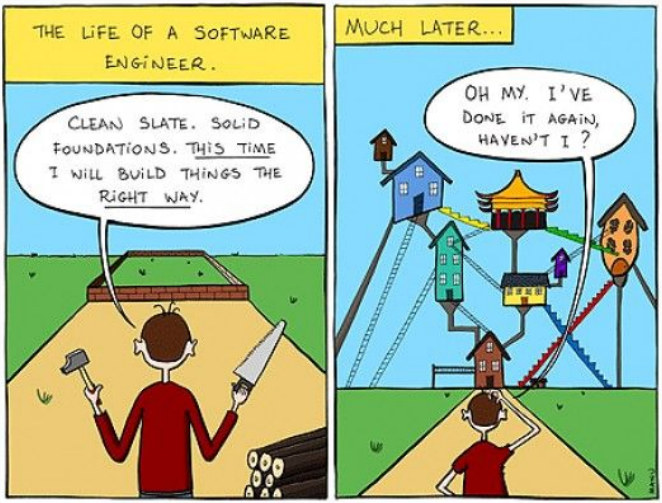
General rule: Catch only specific errors!

Warnings

- Warnings are messages only
- Warnings do not break runtime
- Most of the time you have `DeprecationWarnings` and pandas' https://www.dataquest.io/blog/settingwithcopywarning/`SettingwithCopyWarning`

If you call me for help saying you have an *error* when in fact you have a *warning*, you own me a beer

Refactor as needed



To become a Master...

- Allen B. Downey: "Think Python 2e", Green Tea Press (2015)
- Arthur Turrell: "Coding for Economists" (2021)
- "How to Think Like a Computer Scientist: Interactive Edition"
- Garret Christensen, Jeremy Freese and Edward Miguel "Transparent and Reproducible Social Science Research: How to Do Open Science" UC Press (2019)