Getting Started With Python

A *How-To* Guide for Social Scientists

Ruben Bach
University of Mannheim

✉ *r.bach@uni-mannheim.de*
🐦 *@rub3n_luc*

Andreas Küpfer
Technical University of Darmstadt

✉ *andreas.kuepfer@tu-darmstadt.de*
🌐 *andreaskuepfer.github.io*
🐦 *@ankuepfer*

Social Science Data Lab
MZES, University of Mannheim
February 15, 2023

Outline

1. Why Python?

2. Virtual Environments and Packages

3. Graphical User Interfaces and Integrated Development Environments

4. Alternatives

5. Further notes

6. Additional Resources

Why Python?

Why Python?

Python and R can do the same things, e.g., ...

- Analyze data using regression and machine learning techniques
- Collect data from the web, e.g., through scraping and APIs
- Visualize data

Created with the Imgflip Meme Generator

Why Python?

Python, however, is better suited when ...

- Working with computer scientists
- Using state-of-the-art machine learning, deep learning, natural language processing
- Preparing for a data science job outside of academia
- General purpose programming

Full disclosure: If your work is focused on statistical inference and explanation (coefficients, statistical tests, ...) and some ML and NLP, you'll probably be better-off with R. For prediction-focused ML, deep learning, NLP and heavy data science, you'll probably want to consider using python at some point.

Virtual Environments and Packages

Python

- General purpose programming language
- Three major versions, only one (Python 3) still maintained
- Starting today, obvious choice is Python 3

Is python already installed?

```
$ python –version
```

Versioning of python and packages

- Installing the most recent stable release of python in your root environment and packages is a good starting point
- Sometimes, dependencies require other versions of Python and/or packages, however
- Solution: Set up different virtual environments
  - Easy to keep different versions of Python and packages
  - Avoid problems with different dependencies and updates
  - Can easily switch between virtual environments
  - Makes it easier for your code to run on collaborators' setups

Creating and maintaining virtual environments

With PIP and venv

- **venv**: Should come with your python installation
- Tool to set up and manage virtual environments

Create a new virtual environment using venv

```
$ python -m venv <directory>
```

Activate virtual environment (Linux/Mac)

```
$ source <directory>/bin/activate
```

Activate virtual environment (Windows)

```
$ <directory>/Scripts/activate
```

Deactivate virtual environment

```
(<directory>) $ deactivate
```

Versioning of python and packages

With PIP and venv

- **pip**: **P**ip **I**nstalls **P**ackages
- Standard package manager for python packages
- pip3: specifically installs packages for python 3, can be ignored if you do not have python 2 on your system
- Should come with your python installation
- Install packages within a virtual environment

Install pip

```
$ python -m ensurepip –upgrade
```

Install packages using pip

```
$ pip install <package name>
```

Versioning of python and packages

List installed packages

```
$ pip list
```

- Sometimes, a ***requirements.txt*** file is provided
- Contains required packages and versions

Install packages from requirements.txt

```
$ pip install -r requirements.txt
```

- Note
  - Virtual environments are disposable folder structures
  - Do not put code or data into your virtual environment (folder) manually

Creating and maintaining virtual environments

Conda

- ”Package, dependency and environment management for any language”
- Installation
  - Via Anaconda: python (and R) distribution, popular in data science, many pre-installed packages, Anaconda Navigator (GUI)
  - Via miniconda: Fewer packages than Anaconda, no GUI
  - Install through Anaconda website

Check if conda has been installed

```
$ conda –version
```

Creating and maintaining virtual environments

Conda

Create a new virtual environment

```
$ conda create –name mynewenv python=3.11
```

Activate virtual environment (Linux/Mac)

```
$ source activate mynewenv
```

Activate virtual environment (Windows)

```
$ activate mynewenv
```

Dectivate virtual environment (Linux/Mac)

```
$ source deactivate
```

Dectivate virtual environment (Windows)

```
$ deactivate
```

Creating and maintaining virtual environments

List installed packages

```
$ conda list
```

Install packages (including version number)

```
$ conda install <package name>=0.7.0
```

Update packages (or a specific one)

```
$ conda update [<package name>]
```

Sometimes, a package is not available through conda channels. You could still install it using pip

E.g., install package ”lightgbm” using pip

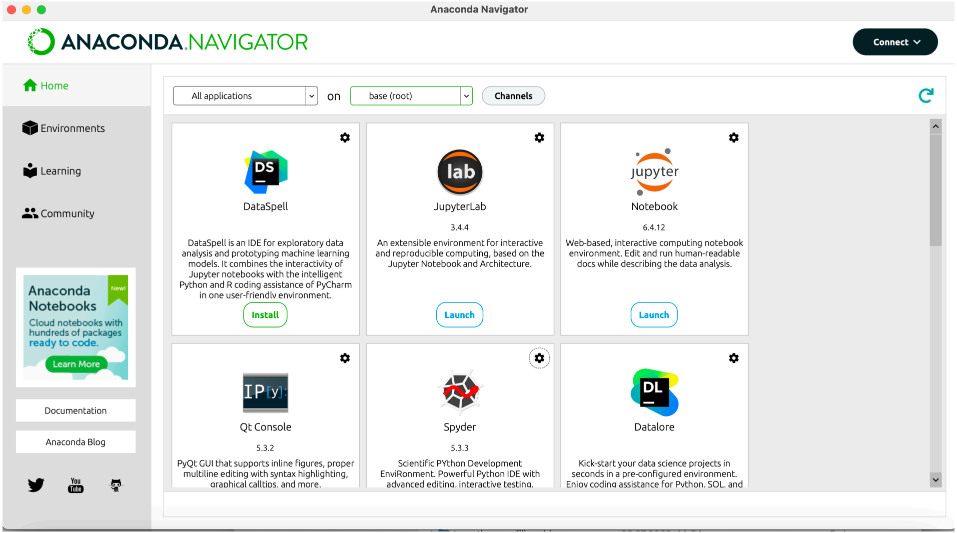
```
$ pip install lightgbm
```

Graphical User Interfaces and Integrated Development Environments

GUIs and IDEs

- There is no one-stop shop like RStudio
- However, Anaconda and Anaconda Navigator integrate many of the tools for a Python data science environment
  - ”Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. (...) Package versions in Anaconda are managed by the package management system conda” Wikipedia
  - ”It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command-line interface (CLI).”

GUIs and IDEs



DataSpell

DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.

JupyterLab

3.4.4

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook

6.4.12

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Qt Console

5.3.2

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder

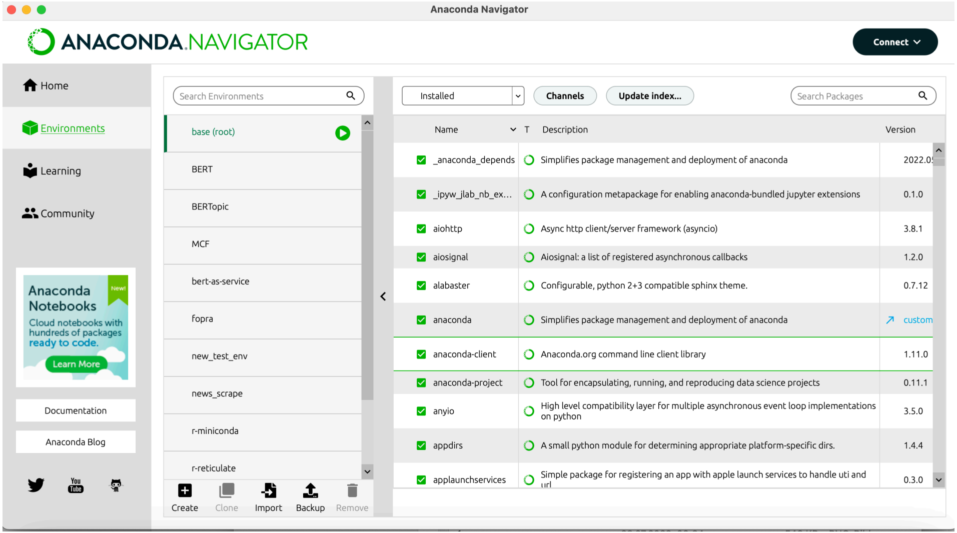
5.3.3

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing,

Datalore

Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and

GUIs and IDEs



	Name	T	Description	Version
☑	_anaconda_depends	○	Simplifies package management and deployment of anaconda	2022.0
☑	_ipyw_jlab_nb_ex...	○	A configuration metapackage for enabling anaconda-bundled jupyter extensions	0.1.0
☑	aiohttp	○	Async http client/server framework (asyncio)	3.8.1
☑	aiosignal	○	Aiosignal: a list of registered asynchronous callbacks	1.2.0
☑	alabaster	○	Configurable, python 2+3 compatible sphinx theme.	0.7.12
☑	anaconda	○	Simplifies package management and deployment of anaconda	↗ custom
☑	anaconda-client	○	Anaconda.org command line client library	1.11.0
☑	anaconda-project	○	Tool for encapsulating, running, and reproducing data science projects	0.11.1
☑	anyio	○	High level compatibility layer for multiple asynchronous event loop implementations on python	3.5.0
☑	appdirs	○	A small python module for determining appropriate platform-specific dirs.	1.4.4
☑	applaunchservices	○	Simple package for registering an app with apple launch services to handle uti and url	0.3.0

IDEs and Notebooks

Several Integrated Development Environments (IDEs), e.g., ...

- Spyder
- PyCharm

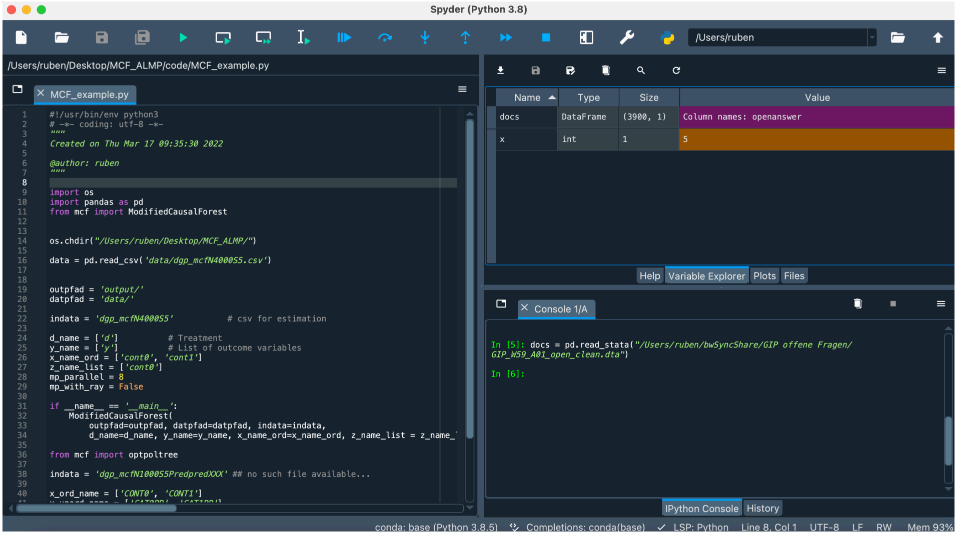
and web-based interactive computing environments like ...

- Jupyter Notebook / JupyterLab
- Google Colaboratory (”Colab”)

Note: Unlike in R/RStudio, you will need to install additional packages through a terminal using pip or conda[1]

[1] Installation through terminal can be sidestepped when using Notebook-type applications.

Spyder

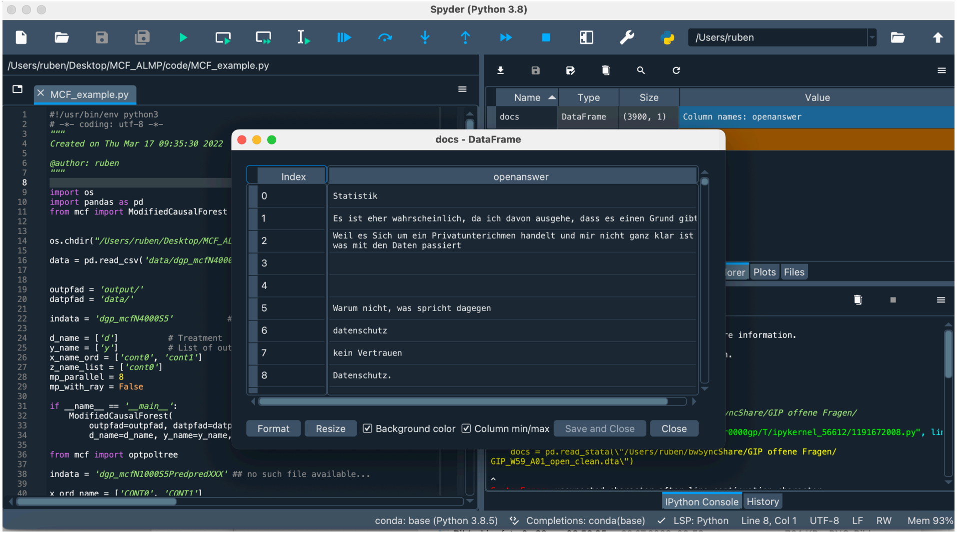


```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 17 09:35:30 2022

@author: ruben
"""

import os
import pandas as pd
from mcf import ModifiedCausalForest


os.chdir("/Users/ruben/Desktop/MCF_ALMP/")

data = pd.read_csv('data/dgp_mcfN4000S5.csv')


outpfad = 'output/'
datpfad = 'data/'

indata = 'dgp_mcfN4000S5'           # csv for estimation

d_name = ['d']           # Treatment
y_name = ['y']           # List of outcome variables
x_name_ord = ['cont0', 'cont1']
z_name_list = ['cont0']
mp_parallel = 8
mp_with_ray = False

if __name__ == '__main__':
    ModifiedCausalForest(
        outpfad=outpfad, datpfad=datpfad, indata=indata,
        d_name=d_name, y_name=y_name, x_name_ord=x_name_ord, z_name_list = z_name_

from mcf import optpoltree

indata = 'dgp_mcfN1000S5PredpredXXX' ## no such file available...

x_ord_name = ['CONT0', 'CONT1']
```

Name ▲	Type	Size	Value
docs	DataFrame	(3900, 1)	Column names: openanswer
x	int	1	5

```
In [5]: docs = pd.read_stata("/Users/ruben/bwSyncShare/GIP offene Fragen/
GIP_W59_A01_open_clean.dta")

In [6]:
```

Spyder



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 17 09:35:30 2022

@author: ruben
"""

import os
import pandas as pd
from mcf import ModifiedCausalForest


os.chdir("/Users/ruben/Desktop/MCF_AL

data = pd.read_csv('data/dgp_mcfN4000


outpfad = 'output/'
datpfad = 'data/'

indata = 'dgp_mcfN4000S5'           #

d_name = ['d']           # Treatment
y_name = ['y']           # List of out
x_name_ord = ['cont0', 'cont1']
z_name_list = ['cont0']
mp_parallel = 8
mp_with_ray = False

if __name__ == '__main__':
    ModifiedCausalForest(
        outpfad=outpfad, datpfad=datp
        d_name=d_name, y_name=y_name,

from mcf import optpoltree

indata = 'dgp_mcfN1000S5PredpredXXX' ## no such file available...

x_ord_name = ['CONT0', 'CONT1']
```

Name	Type	Size	Value
docs	DataFrame	(3900, 1)	Column names: openanswer

docs - DataFrame

Index	openanswer
0	Statistik
1	Es ist eher wahrscheinlich, da ich davon ausgehe, dass es einen Grund gibt
2	Weil es Sich um ein Privatunterichmen handelt und mir nicht ganz klar ist was mit den Daten passiert
3	
4	
5	Warum nicht, was spricht dagegen
6	datenschutz
7	kein Vertrauen
8	Datenschutz.

```
re information.

n.

yncShare/GIP offene Fragen/

r0000gp/T/ipykernel_56612/1191672008.py", lin
    docs = pd.read_stata(\"/Users/ruben/bwSyncShare/GIP offene Fragen/
GIP_W59_A01_open_clean.dta\")

^
```

Jupyter Notebook

- Web-based interactive computing environments
- Combine markdown with code, interactive cells, lots of exporting and publishing options
- Can run bash commands from within notebook using ”!”

Example: List installed packages using bash

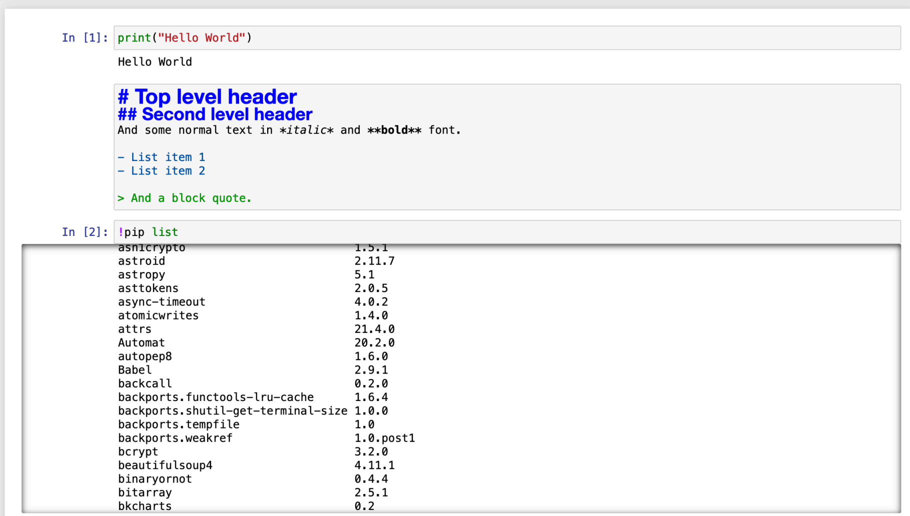
```
!pip list
```

- Magics (% and %%) give additional cell functionality

Example: Include HTML content in notebook

```
%%HTML
```

Jupyter Notebook

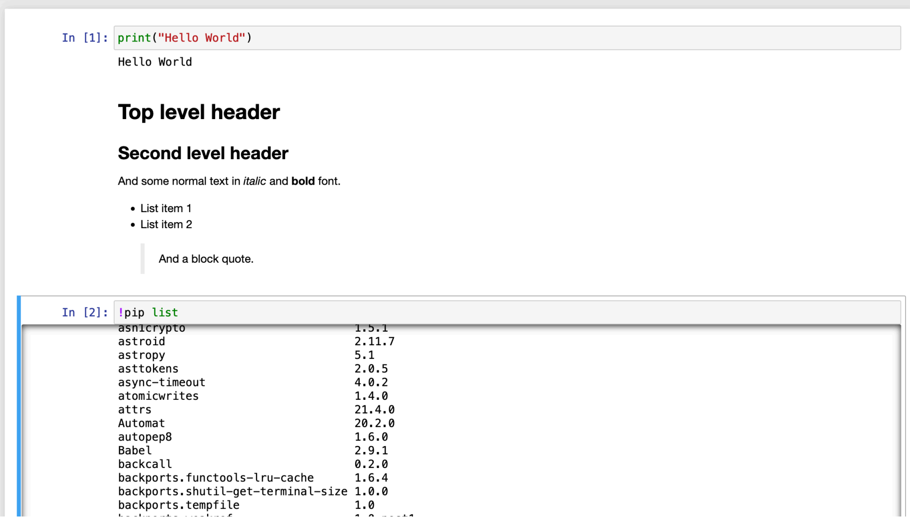


```
In [1]: print("Hello World")

Hello World
```

```
# Top level header
## Second level header
And some normal text in *italic* and **bold** font.

- List item 1
- List item 2

> And a block quote.
```

```
In [2]: !pip list

asn1crypto                         1.5.1
astroid                            2.11.7
astropy                            5.1
asttokens                          2.0.5
async-timeout                      4.0.2
atomicwrites                       1.4.0
attrs                              21.4.0
Automat                            20.2.0
autopep8                           1.6.0
Babel                              2.9.1
backcall                           0.2.0
backports.functools-lru-cache      1.6.4
backports.shutil-get-terminal-size 1.0.0
backports.tempfile                 1.0
backports.weakref                  1.0.post1
bcrypt                             3.2.0
beautifulsoup4                     4.11.1
binaryornot                        0.4.4
bitarray                           2.5.1
bkcharts                           0.2
```

Jupyter Notebook



```
In [1]: print("Hello World")
```

```
Hello World
```

Top level header

Second level header

And some normal text in *italic* and **bold** font.

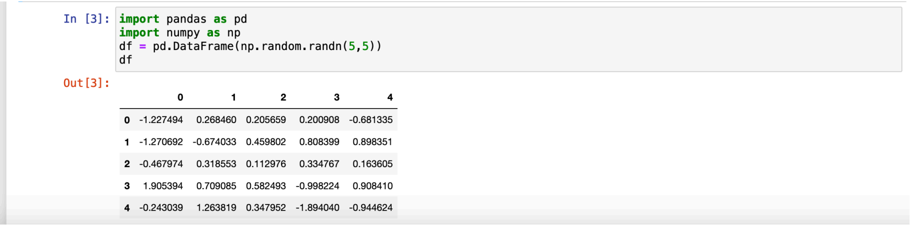
- List item 1
- List item 2

> And a block quote.

```
In [2]: !pip list
```

```
asn1crypto                        1.5.1
astroid                           2.11.7
astropy                           5.1
asttokens                         2.0.5
async-timeout                     4.0.2
atomicwrites                      1.4.0
attrs                             21.4.0
Automat                           20.2.0
autopep8                          1.6.0
Babel                             2.9.1
backcall                          0.2.0
backports.functools-lru-cache     1.6.4
backports.shutil-get-terminal-size 1.0.0
backports.tempfile                1.0
backports.weakref                 1.0.post1
```

Jupyter Notebook



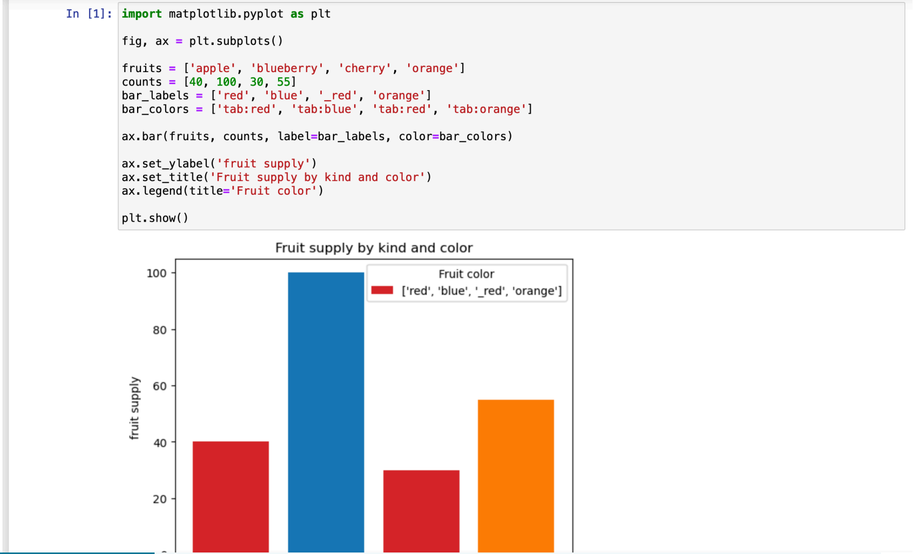
In [3]:

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5,5))
df
```

Out[3]:

	0	1	2	3	4
0	-1.227494	0.268460	0.205659	0.200908	-0.681335
1	-1.270692	-0.674033	0.459802	0.808399	0.898351
2	-0.467974	0.318553	0.112976	0.334767	0.163605
3	1.905394	0.709085	0.582493	-0.998224	0.908410
4	-0.243039	1.263819	0.347952	-1.894040	-0.944624

Jupyter Notebook



```
In [1]: import matplotlib.pyplot as plt

        fig, ax = plt.subplots()

        fruits = ['apple', 'blueberry', 'cherry', 'orange']
        counts = [40, 100, 30, 55]
        bar_labels = ['red', 'blue', '_red', 'orange']
        bar_colors = ['tab:red', 'tab:blue', 'tab:red', 'tab:orange']

        ax.bar(fruits, counts, label=bar_labels, color=bar_colors)

        ax.set_ylabel('fruit supply')
        ax.set_title('Fruit supply by kind and color')
        ax.legend(title='Fruit color')

        plt.show()
```

Fruit supply by kind and color
Fruit color
🟥 ['red', 'blue', '_red', 'orange']
fruit supply
100
80
60
40
20

Google Colaboratory

”Colab”

- Web-based interactive computing environments
- Google account required (to edit code)
- Runs on Google hardware (including free GPU use)
- Powerful hardware available per premium plans
- Markdown-based text cells and code cells, magic commands, bash commands (”!”), R kernels available, LaTeX, ...
- Load data from your Google Drive
- Pay attention when sharing colab notebooks

Google Colaboratory

”Colab” – Downsides

- Lose data and results when runtime stops
- Uploaded files removed when session restarted, no persistent storage
- Sensitive data? Sensitive code? Sensitive research?

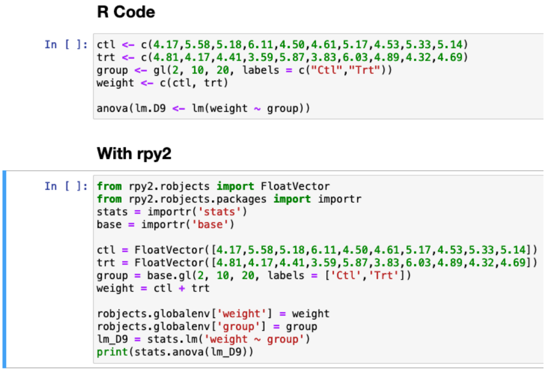
Alternatives

Reticulate

- An R package that allows you to run Python code in R
- Requires miniconda or Anaconda installation
- Management of virtual environments and packages through conda
- Pro: Allows you to switch between R and Python code and access objects created in Python (e.g., data) in R
- Con: Environments and packages need to be managed through conda, troubleshooting can be more difficult

rpy2

- A Python library that allows you to run R code in Python
- You have to ‘translate’ R functions to call them from Python
- Alternatively, write and evaluate R code using rmagic in Jupyter Notebook (based on *rpy2*)



R Code

```
In [ ]: ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
        trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
        group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
        weight <- c(ctl, trt)

        anova(lm.D9 <- lm(weight ~ group))
```

With rpy2

```
In [ ]: from rpy2.robjects import FloatVector
        from rpy2.robjects.packages import importr
        stats = importr('stats')
        base = importr('base')

        ctl = FloatVector([4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14])
        trt = FloatVector([4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69])
        group = base.gl(2, 10, 20, labels = ['Ctl','Trt'])
        weight = ctl + trt

        robjects.globalenv['weight'] = weight
        robjects.globalenv['group'] = group
        lm_D9 = stats.lm('weight ~ group')
        print(stats.anova(lm_D9))
```

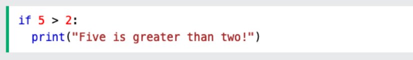
Further notes

Naming conventions

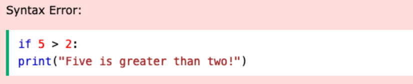
- A name can only consist of characters from three groups: digits (0-9), letters (a-z and A-Z), and underscores (_)
- A name cannot start with a digit
- A name cannot coincide with one of Python's reserved words, which have special meaning to Python, e.g., ...
  - False, not, class, finally, is, return, None, continue, while, del, else, ...

Indentation

- Indentation: spaces at the beginning of a code line
- E.g., in R, indentation is for readability only, in Python it is very important
- Python uses indentation to indicate a block of code
- Number of spaces is up to you, but it has to be at least one and consistent



```python
if 5 > 2:
  print("Five is greater than two!")
```

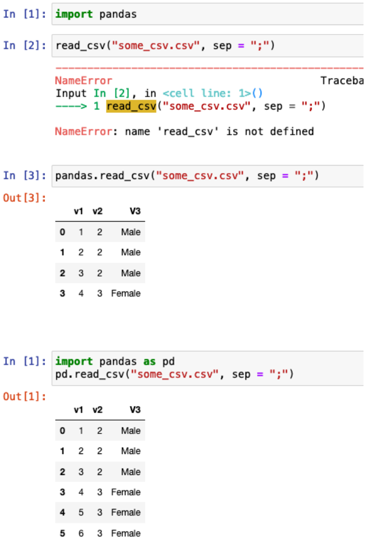


Syntax Error:

```python
if 5 > 2:
print("Five is greater than two!")
```

Importing and Using Packages

- Depending on how you import a package, you will need to reference it when using a function
- Usually a good idea to use alternate names (e.g., pandas -> pd, numpy -> np)
- Can import submodules instead of whole packages



```
In [1]: import pandas

In [2]: read_csv("some_csv.csv", sep = ";")

        ---------------------------------------------------------------
        NameError                                          Traceba
        Input In [2], in <cell line: 1>()
        ----> 1 read_csv("some_csv.csv", sep = ";")

        NameError: name 'read_csv' is not defined

In [3]: pandas.read_csv("some_csv.csv", sep = ";")
Out[3]:
```

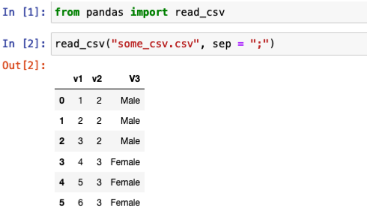
	v1	v2	V3
0	1	2	Male
1	2	2	Male
2	3	2	Male
3	4	3	Female

```
In [1]: import pandas as pd
        pd.read_csv("some_csv.csv", sep = ";")
Out[1]:
```

	v1	v2	V3
0	1	2	Male
1	2	2	Male
2	3	2	Male
3	4	3	Female
4	5	3	Female
5	6	3	Female

Importing and Using Packages

- Depending on how you import a package, you will need to reference it when using a function
- Usually a good idea to use alternate names (e.g., pandas -> pd, numpy -> np)
- Can import elements of a package instead of whole packages



```
In [1]: from pandas import read_csv

In [2]: read_csv("some_csv.csv", sep = ";")
Out[2]:
         v1  v2      V3
      0   1   2    Male
      1   2   2    Male
      2   3   2    Male
      3   4   3  Female
      4   5   3  Female
      5   6   3  Female
```

Additional Resources

Some Helpful Resources

- Virtual environments and managing them with conda
- Jupyter Notebook tutorial
- reticulate
- rpy2
- Datacamp