

# Command-line interfaces

CREATING ROBUST PYTHON WORKFLOWS

Martin Skarzynski

Co-Chair, Foundation for Advanced  
Education in the Sciences (FAES)



# Synergy with the shell



Photo by Adrianna Calvo from Canva

- Command-line interfaces (CLIs)
  - Pass shell arguments to Python scripts

```
python fit_model.py --alpha 0.2
```
  - Avoid using comments as an on/off switch

```
# To retrain the model,  
# Uncomment the line below.  
# model.fit(x_train, y_train)
```

# Build command-line interfaces

- argparse
  - Instantiate the `ArgumentParser` class
  - Call the `add_argument()` method
  - Call the `parse_args()` method
  - Arguments are `Namespace` attributes
- docopt
  - Setup up CLI in file docstring
  - Pass `__doc__` to `docopt()`
  - Arguments are dictionary values

```
parser = argparse.ArgumentParser()  
  
parser.add_argument('ARGUMENT_NAME')  
  
namespace = parser.parse_args()  
  
namespace.ARGUMENT_NAME
```

```
"""Pass a single argument to FILENAME.  
Usage: FILENAME.py ARGUMENT_NAME"""  
  
argument_dict = docopt.docopt(__doc__)
```

# Argparse and methods

```
import datetime
from get_namespace import get_namespace

if __name__ == '__main__':
    namespace = get_namespace()
    print(datetime.datetime.now()
          .strftime(namespace.format))
```

```
$ python now.py %H:%M
```

8:30

```
import argparse as ap

def get_namespace() -> ap.Namespace:
    parser = ap.ArgumentParser()
    parser.add_argument('format')
    return parser.parse_args()
```

# Docopt and docstrings

```
import datetime as dt
from get_arg_dict import get_arg_dict

if __name__ == '__main__':
    arg_dict = get_arg_dict()
    print(dt.datetime.now()
          .strftime(arg_dict['FORMAT']))
```

```
$ python now.py %B/%d
```

March 8

"""Get the current date or time.

Usage: now.py FORMAT"""

```
from typing import Dict
```

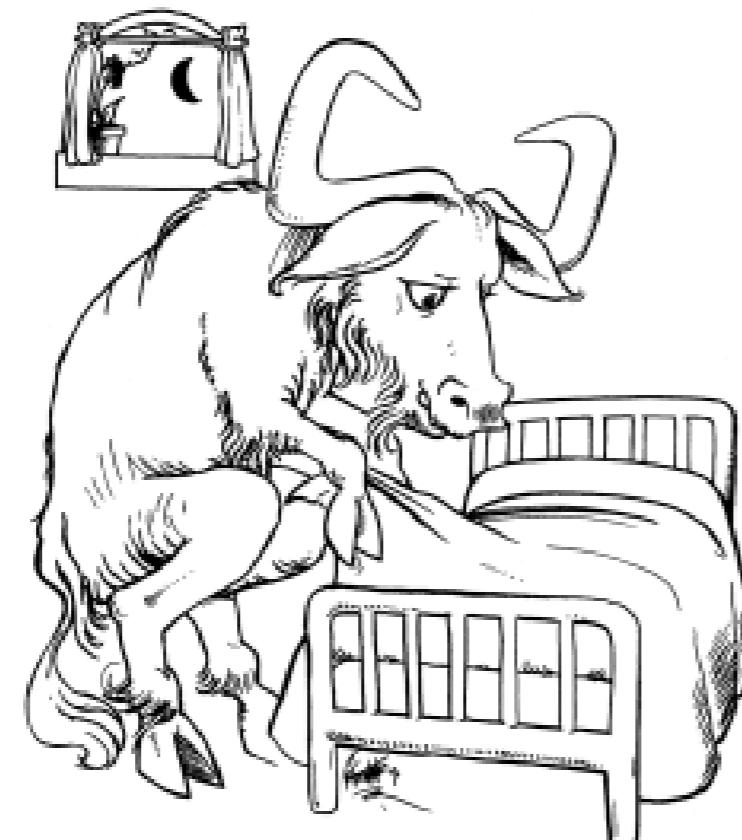
```
import docopt
```

```
def get_arg_dict() -> Dict[str, str]:
```

```
    return docopt.docopt(__doc__)
```

# GNU Make

A Program for Directed Compilation



for *GNU Make* Version 3.81

by Richard M. Stallman,  
Roland McGrath and Paul D. Smith

Automate execution of

- Shell
  - Commands: `echo hello`
  - Scripts: `bash myscript.sh`
- Python
  - Scripts: `python myscript.py`
  - Modules: `python -m timeit -h`

# Make shell command

Makefile:

```
time:  
    python now.py %H:%M
```

```
$ make
```

```
python now.py  
19:30
```

Makefile structure:

```
TARGET:  
RECIPE
```

# Make dependencies

Makefile:

```
time:  
    python now.py %H:%M  
  
date:  
    python now.py %d/%m  
  
all: date time
```

Makefile structure:

```
TARGET:  
    RECIPE  
  
TARGET:  
    RECIPE  
  
TARGET: DEPENDENCIES
```

# Make all

Makefile:

```
time:  
    python now.py %H:%M
```

```
date:  
    python now.py %m-%d
```

```
all: date time
```

```
$ make all
```

```
python now.py %H:%M  
19:32  
python now.py %d/%m  
08/03
```

# Command-line interface documentation

## Nbless

A Python package for  
programmatic Jupyter  
notebook workflows



## Navigation

Contents:

[Project overview](#)

[Command-line interface](#)

- [nbless](#)
- [nbuild](#)
- [nbconv](#)
- [nbexec](#)

[Module reference](#)

[Test reference](#)

## Quick search

 Go

## nbuild

Create an unexecuted Jupyter notebook from markdown and code files.

```
nbuild [OPTIONS] IN_FILES...
```

### Options

**-o, --out\_file <out>**

### Arguments

#### IN\_FILES

Required argument(s)

## nbconv

Convert a notebook into various formats using `nbformat` exporters.

```
nbconv [OPTIONS] IN_FILE
```

### Options

**-e, --exporter <exporter>**

**-o, --out\_file <out>**

### Arguments

#### IN\_FILE

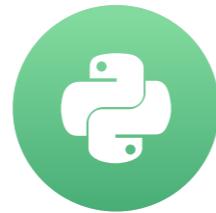
Required argument

# Let's practice building CLIs!

CREATING ROBUST PYTHON WORKFLOWS

# Git version control

CREATING ROBUST PYTHON WORKFLOWS



Martin Skarzynski

Co-Chair, Foundation for Advanced  
Education in the Sciences (FAES)

# Git

- Records and manages modifications made to projects
- Prevents lost work or unwanted changes



git

# Basic git workflow



- Make changes in the working directory ( `( . )` )
- Add changes to the index ( `( ./ .git/index )` )
- Commit changes to version control history ( `( ./ .git )` )
- A "commit" ~ a milestone

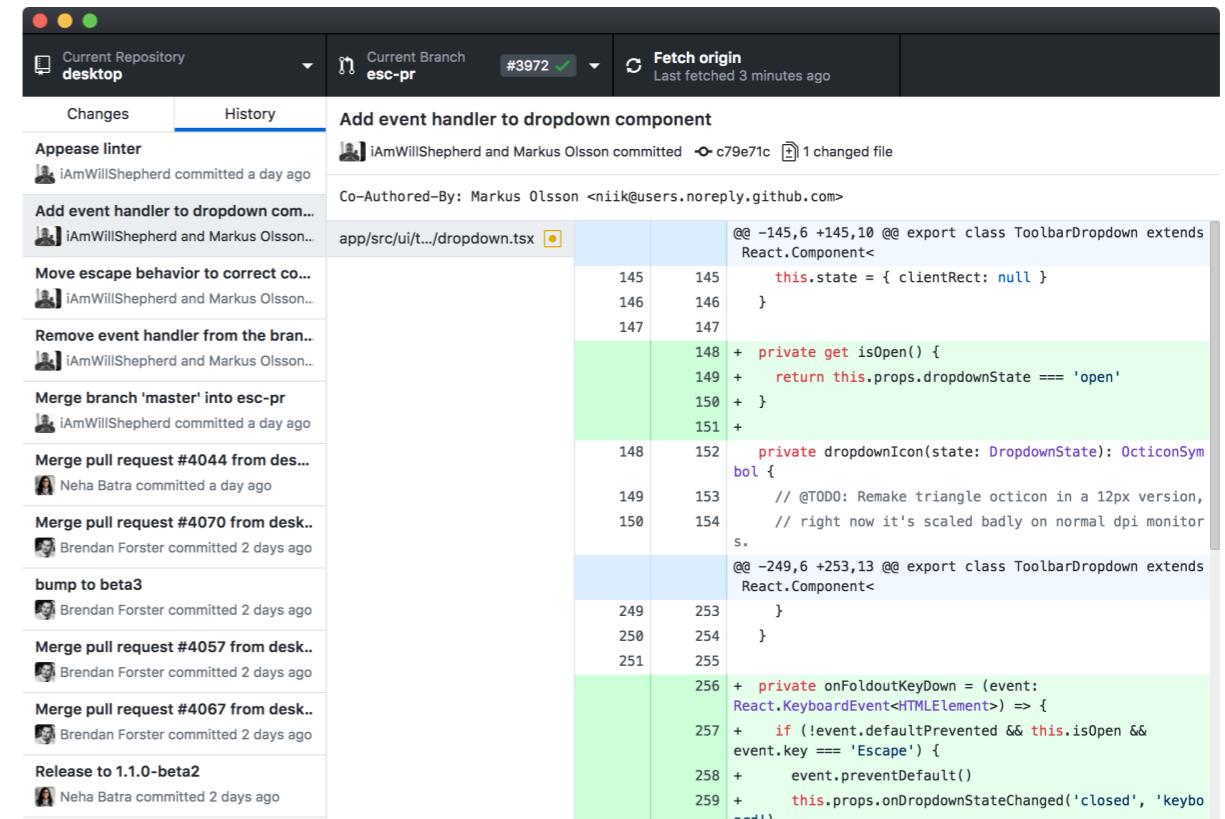
# Shell versus API

Use git via

- Graphical User Interface (GUI)
- Command line (shell)
- Application Programming Interface (API)

GitPython (Python interface to git)

- Create a version controlled directory
  - Called a repository
- Add changes to the index
- Commit changes to version control history



A screenshot of a GitHub pull request interface. The top bar shows "Current Repository desktop", "Current Branch esc-pr #3972", and "Fetch origin Last fetched 3 minutes ago". The main area has tabs for "Changes" and "History". The "History" tab is selected, showing a list of commits:

- Appease linter
- Add event handler to dropdown component
- Move escape behavior to correct co...
- Remove event handler from the bran...
- Merge branch 'master' into esc-pr
- Merge pull request #4044 from des...
- Merge pull request #4070 from desk..
- bump to beta3
- Merge pull request #4057 from desk..
- Merge pull request #4067 from desk..
- Release to 1.1.0-beta2

The "Changes" tab shows a detailed view of the commit "Add event handler to dropdown component". It displays the file "app/src/ui/t.../dropdown.tsx" with a diff between lines 145 and 259. The diff highlights additions in green and deletions in red. The code snippet includes imports, class definitions, and methods like "isOpen()".



# Git init

Run `git init` shell command:

```
$ git init
```

```
Initialized empty Git repository in  
/Users/USERNAME/REPONAME/.git/
```

Run Python code in a file called `init.py`:

```
import git  
print(git.Repo.init())
```

```
$ python init.py
```

```
<git.Repo "/Users/USER/REPO/.git">
```

# Make init

Run `git init` using Make:

```
.git/:  
    git init
```

```
$ make
```

```
git init  
make: '.git/' is up to date.
```

Run `init.py` using Make:

```
.git/:  
    python init.py
```

```
$ make
```

```
python init.py  
make: '.git/' is up to date.
```

# Untracked files

```
import git  
  
repo = git.Repo()
```

```
repo.untracked_files
```

```
['Makefile', 'init.py', 'commit.py']
```

# Add untracked files

```
import git  
  
repo = git.Repo()  
  
add_list = repo.untracked_files  
  
if add_list:  
    repo.index.add(add_list)
```

```
repo.untracked_files
```

```
[]
```

# Commit message

```
import git

repo = git Repo()
add_list = repo.untracked_files

if add_list:
    repo.index.add(add_list)
    new = f"New files: {', '.join(add_list)}."
```

# Commit new files

```
import git

repo = git.Repo()
add_list = repo.untracked_files

if add_list:
    repo.index.add(add_list)
    new = f"New files: {', '.join(add_list)}"
    print(repo.index.commit(new).message)
```

```
$ python commit.py
```

```
New files: Makefile, init.py, commit.py.
```

# Diff all files

```
import git  
  
repo = git.Repo()  
diff = repo.index.diff(None)
```

# Diff modified files

```
import git

repo = git.Repo()
diff = repo.index.diff(None).iter_change_type('M')
edit_list = [file.a_path for file in diff]
```

# Commit modified files

```
import git

repo = git.Repo()

diff = repo.index.diff(None).iter_change_type('M')

edit_list = [file.a_path for file in diff]

if edit_list:

    repo.index.add(edit_list)

    modified = f"Modified files: {', '.join(edit_list)}"

    print(repo.index.commit(modified).message)
```

# Make commit

```
$ make
```

```
python commit.py
```

Modified files: Makefile, commit.py.

```
$ make
```

```
python commit.py
```

Current Repository: desktop  
Current Branch: esc-pr #3972  
Fetch origin: Last fetched 3 minutes ago

Add event handler to dropdown component

Appease linter  
iAmWillShepherd committed a day ago

Add event handler to dropdown com...  
iAmWillShepherd and Markus Olsson...

Move escape behavior to correct co...  
iAmWillShepherd and Markus Olsson...

Remove event handler from the bran...  
iAmWillShepherd and Markus Olsson...

Merge branch 'master' into esc-pr  
iAmWillShepherd committed a day ago

Merge pull request #4044 from des...  
Neha Batra committed a day ago

Merge pull request #4070 from desk..  
Brendan Forster committed 2 days ago

bump to beta3  
Brendan Forster committed 2 days ago

Merge pull request #4057 from desk..  
Brendan Forster committed 2 days ago

Merge pull request #4067 from desk..  
Brendan Forster committed 2 days ago

Release to 1.1.0-beta2  
Neha Batra committed 2 days ago

app/src/ui/t.../dropdown.tsx

```
@@ -145,6 +145,10 @@ export class ToolbarDropdown extends React.Component<
    this.state = { clientRect: null }
  }

  + private get isOpen() {
  +   return this.props.dropdownState === 'open'
  + }

  private dropdownIcon(state: DropdownState): OcticonSymbol {
    // @TODO: Remake triangle octicon in a 12px version,
    // right now it's scaled badly on normal dpi monitor
    // ...
  }

  @@ -249,6 +253,13 @@ export class ToolbarDropdown extends React.Component<
    }

    + private onFoldoutKeyDown = (event: React.KeyboardEvent<HTMLElement>) => {
    +   if (!event.defaultPrevented && this.isOpen && event.key === 'Escape') {
    +     event.preventDefault()
    +     this.props.onDropdownStateChanged('closed', 'keybo
  
```

~/src/libgit2 ➔ ↵ development ➔

**Let's practice  
automating version  
control workflows!**

**CREATING ROBUST PYTHON WORKFLOWS**

# Virtual environments

CREATING ROBUST PYTHON WORKFLOWS



Martin Skarzynski

Co-Chair, Foundation for Advanced  
Education in the Sciences (FAES)

# Virtual environments



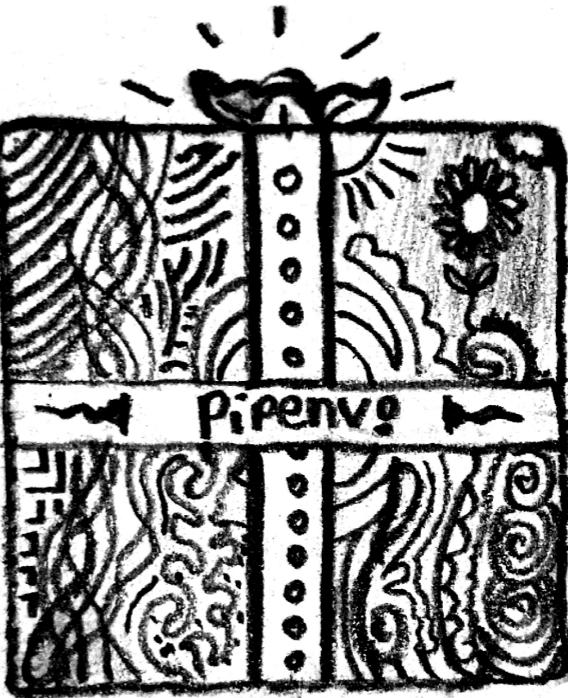
- Are directories that
  - Contain separate Python installations
  - Facilitate
    - Dependency management
    - Reproducible analysis
    - Python package development

Photo by Katarzyna Modrzejewska from Canva

# Environment managers

- Tools that create and manage virtual environments

Manager	Superpower
venv	Part of the standard library
virtualenv	Widely-used
pipenv	High-level interface
conda	Not only for Python



**CONDA**

# Dependency managers

## Dependency manager

pip

pipenv

conda

- Dependency managers = tools that install and manage dependencies
- `venv` and `virtualenv` use `pip`
- `pipenv` and `conda` manage dependencies and environments

# Dependency manager files

Dependency manager	Dependency management file
pip	requirements.txt
pipenv	Pipfile
conda	environment.yml

- Dependency names and versions can be
  - Specified in dependency management files
- All three dependency managers use requirements.txt
  - Exact ( jupyter == 4.4.0 )
  - Minimum ( python >= 3.6.0 )

# Install dependencies

Create environment

```
$ python -m venv .venv
```

Option 1: Activate environment before `pip install`

```
$ source .venv/bin/activate  
$ pip install --requirement requirements.txt
```

Option 2: Use path to virtual environment Python to `pip install`

```
$ .venv/bin/python -m pip install -r requirements.txt
```

# Make venv

1. Create a virtual environment in `.venv`
2. Install the dependencies in the virtual environment
3. Update the "Last Modified" time for the target (`.venv/bin/activate`)

```
.venv:  
    python -m venv .venv  
  
.venv/bin/activate: .venv requirements.txt  
    .venv/bin/python -m pip install -r requirements.txt  
    touch .venv/bin/activate
```

# Make test

- Run all tests in a virtual environment
- Avoid failed tests due to missing or outdated project dependencies

```
.venv:  
    python -m venv .venv  
.venv/bin/activate: .venv requirements.txt  
    .venv/bin/python -m pip install -r requirements.txt  
    touch .venv/bin/activate  
  
test: .venv/bin/activate  
    .venv/bin/python -m pytest src tests
```

# Venv module

```
import venv  
venv.create('.venv')
```

1. Create a `venv` environment
  - o With `venv` API
    - `create()` function

# Subprocess module

```
import subprocess  
  
cp = subprocess.run(  
    ['.venv/bin/python', '-m',  
     'pip', 'list'], stdout=-1  
)  
  
print(cp.stdout.decode())
```

Package	Version
PACKAGE_NAME	0.0.1
...	

1. Create a `venv` environment
  - o With `venv` API
    - `create()` function
2. List packages
  - o With `pip` and `subprocess`
    - `subprocess.run()` function
    - Returns `CompletedProcess`
    - `stdout` or `capture_output`
    - `pip list`

# Package information

```
cp = subprocess.run(['python', '-m', 'pip', 'show', 'pandas'], stdout=-1)

print(cp.stdout.decode())
```

Name: pandas

Version: 0.24.1

Summary: Powerful data structures for data analysis, time series, and statistics

Home-page: <http://pandas.pydata.org>

...

# Summary

Environment manager	Dependency manager	Dependency management file	Environment manager Superpower
venv	pip	requirements.txt	Part of the standard library

- venv API:
  - create()
- Use subprocess.run()
  - To run shell commands, such as
    - pip install
    - pip list
    - pip show

# Summary

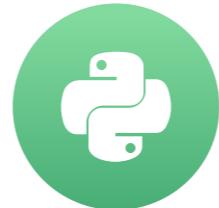
Environment manager	Dependency manager	Dependency management file	Environment manager Superpower
venv	pip	requirements.txt	Part of the standard library
virtualenv	pip	requirements.txt	Widely-used
pipenv	pipenv	Pipfile	High-level interface
conda	conda	environment.yml	Not only for Python

**Let's practice using  
virtual  
environments!**

**CREATING ROBUST PYTHON WORKFLOWS**

# Persistence and packaging

CREATING ROBUST PYTHON WORKFLOWS

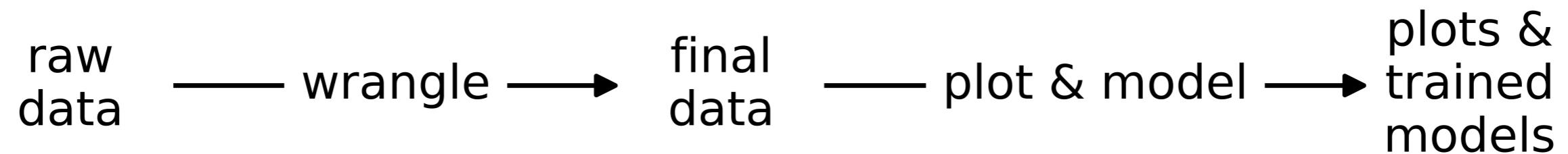


Martin Skarzynski

Co-Chair, Foundation for Advanced  
Education in the Sciences (FAES)

# Persistence in a pipeline

- Save files throughout a data analysis pipeline
  - Processed data
  - Plots
  - Trained models

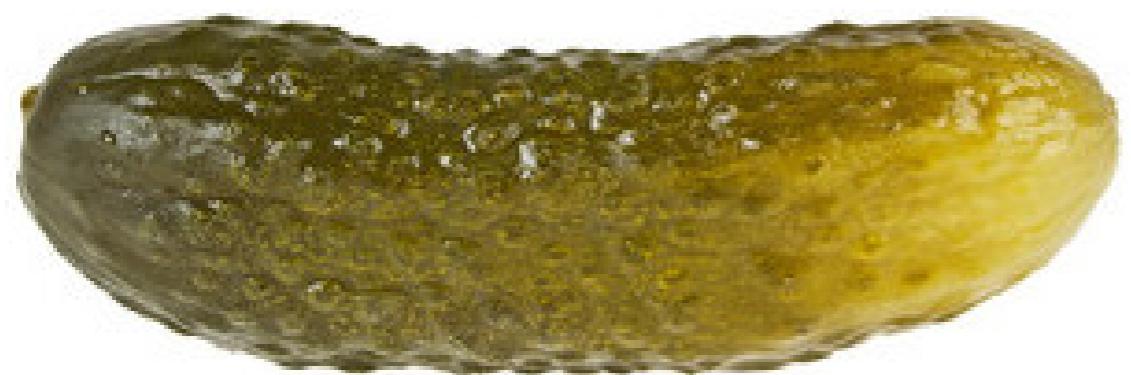


- Each file should be
  - A target in the project's `Makefile`
  - The sole responsibility of a script ( `plot1.py -> plot1.png` )

# Save data with pandas

Format	Writer
CSV	to_csv()
JSON	to_json()
Excel	to_excel()
Pickle	to_pickle()

```
# Make simple dataframe  
df = pd.DataFrame({  
    'Evens': range(0, 5, 2),  
    'Odds': range(1, 6, 2)  
})  
  
# Pickle dataframe  
df.to_pickle('numbers.pkl')
```



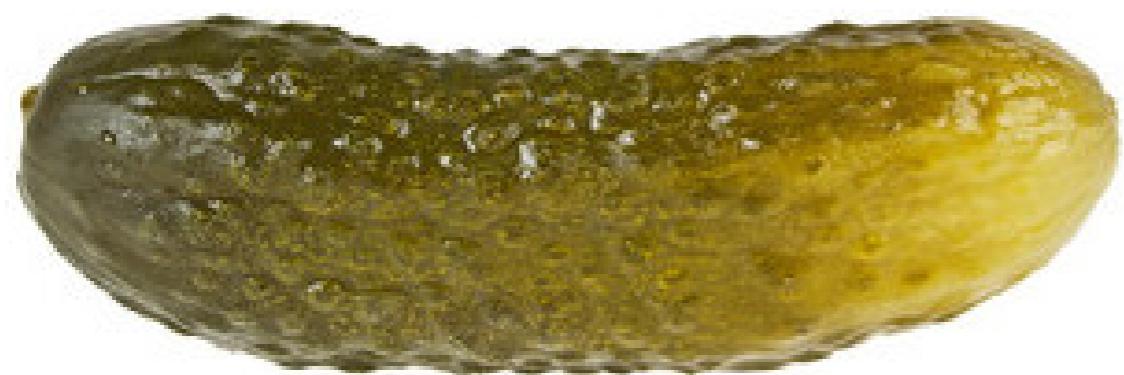
# Read data with pandas

Format	Writer	Reader
CSV	to_csv()	read_csv()
JSON	to_json()	read_json()
Excel	to_excel()	read_excel()
Pickle	to_pickle()	read_pickle()

```
# Unpickle dataframe  
pd.read_pickle('numbers.pkl')
```

Evens   Odds

0	0	1
1	2	3
2	4	5



# Why pickle?

- Built into `pandas`
  - `df.to_pickle('df.pkl')`
- Standard library includes `pickle` module
- `joblib` is optimized for `scikit-learn`
- Fast and memory efficient
- Works for many types of Python objects
- Supports on-the-fly (de)compression
  - `df.to_pickle('df.pkl.xz')`



# Pickle models

```
import joblib
from sklearn import datasets, model_selection, neighbors

diabetes = datasets.load_diabetes()

x_train, x_test, y_train, y_test = model_selection.train_test_split(
    diabetes.data, diabetes.target, test_size=0.33, random_state=42)

knn = neighbors.KNeighborsRegressor().fit(x_train, y_train)

# Pickle the k-nearest neighbors model with joblib
joblib.dump(knn, 'knn.pkl')
```

# Why package code?

- Everything in one place
  - code
  - documentation
  - data files
- Easy to share
  - Install from Python Package Index (PyPI):  
`pip install mypkg`
  - Install from git repository:  
`pip install git+REPO_URL`
  - Install local package: `pip install .`



# Package pickled objects

```
setuptools.setup(  
    name="PACKAGE_NAME",  
    version="MAJOR.MINOR.PATCH",  
    description="A minimal example of packaging pickled data and models.",  
    packages=setuptools.find_packages("src"),  
    package_dir={"": "src"},  
    # Include files in the data and models directories  
    package_data={"": ["data/*", "models/*"]},  
)
```

# Install local package

```
.venv:
```

```
    python -m venv .venv
```

```
.venv/bin/activate: requirements.txt
```

```
    .venv/bin/python3 -m pip install --requirement requirements.txt
```

```
    touch .venv/bin/activate
```

```
requirements.txt :
```

```
--editable .
```

# Automated release

- Create a Python Package Index (PyPI) account:  
<https://pypi.org/account/register/>

```
clean:  
    rm -rf dist/  
dist: clean  
    python setup.py sdist bdist_wheel  
release: dist  
    twine upload dist/*
```

```
$ make release
```

```
...  
Uploading distributions to https://...  
Uploading ...-0.0.8-py3-none-any.whl  
100%|??????????| 22.7k/22.7k...  
Uploading ...-0.0.8.tar.gz  
100%|??????????| 32.3k/32.3k...
```

# Access package data

```
import pkgutil
from pickle import loads
loads(pkgutil.get_data(
    'PACKAGE_NAME',
    'data/numbers.pkl'
))
```

```
??? setup.py
??? src
    ??? PACKAGE_NAME
        ??? __init__.py
        ??? data
    ?  ??? data.pkl
??? MODULE.py
```

	Evens	Odds
0	0	1
1	2	3
2	4	5

# Access package model

```
import pkg_resources
from joblib import load
load(pkg_resources.resource_filename(
    'PACKAGE_NAME',
    'models/knn.pkl'
))
```

```
KNeighborsRegressor(algorithm='auto',
leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=None,
n_neighbors=5, p=2, weights='uniform')
```

```
??? setup.py
??? src
    ??? PACKAGE_NAME
        ??? __init__.py
        ??? models
            ?   ??? knn.pkl
    ??? MODULE.py
```

# Let's practice pickling!

CREATING ROBUST PYTHON WORKFLOWS