

Writing Your First Package

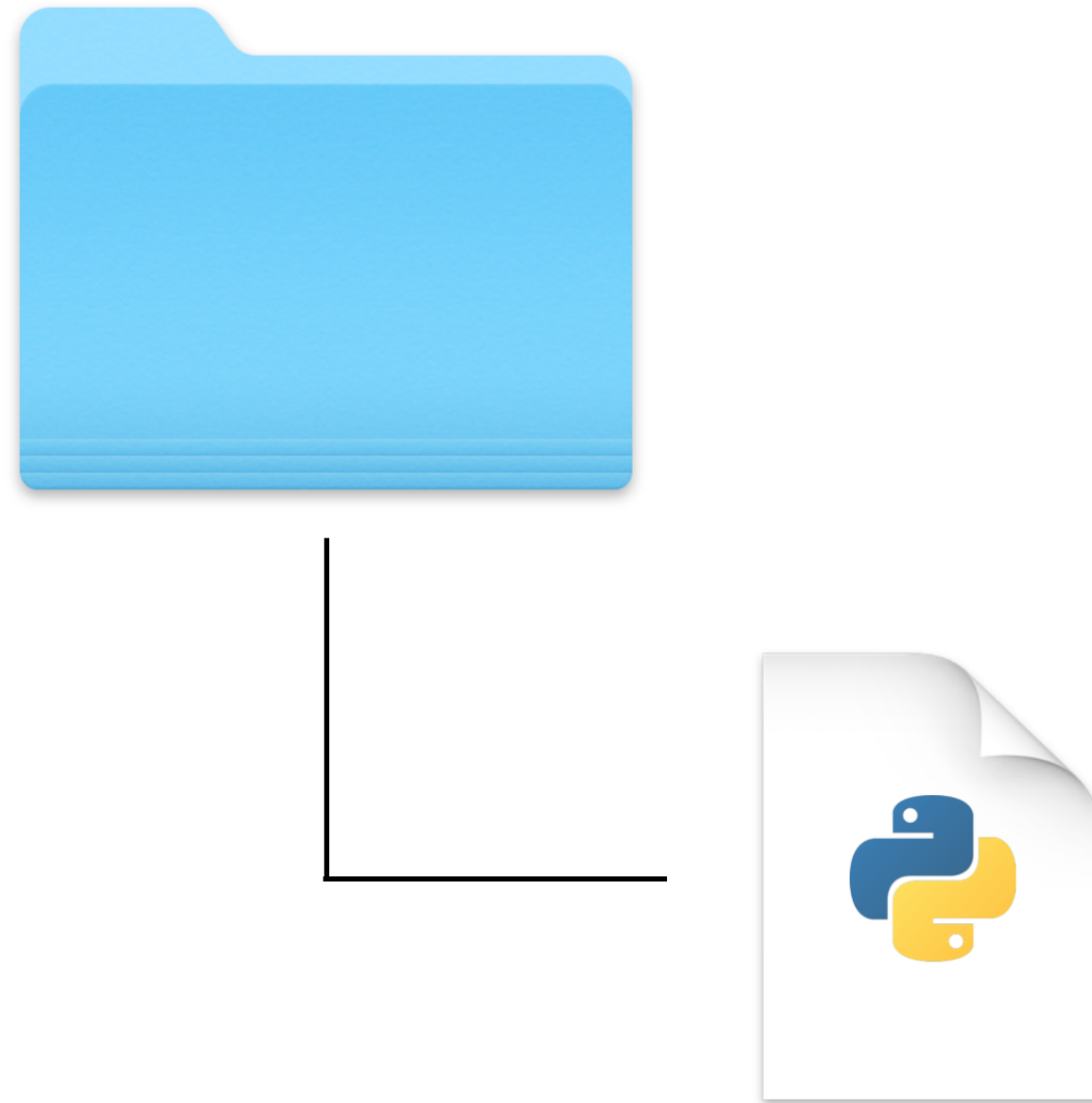
SOFTWARE ENGINEERING PRINCIPLES IN PYTHON



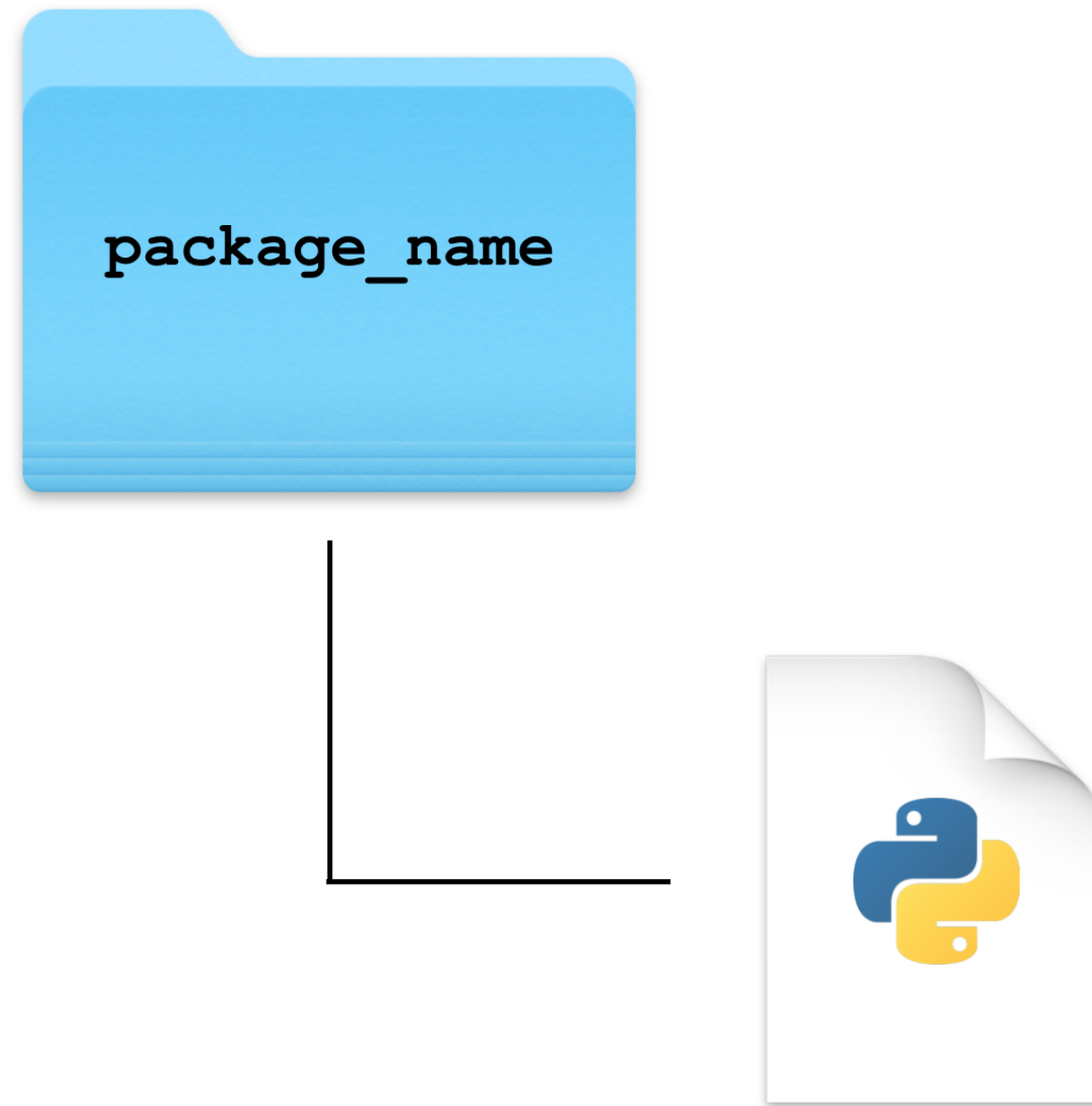
Adam Spannbauer

Machine Learning Engineer at Eastman

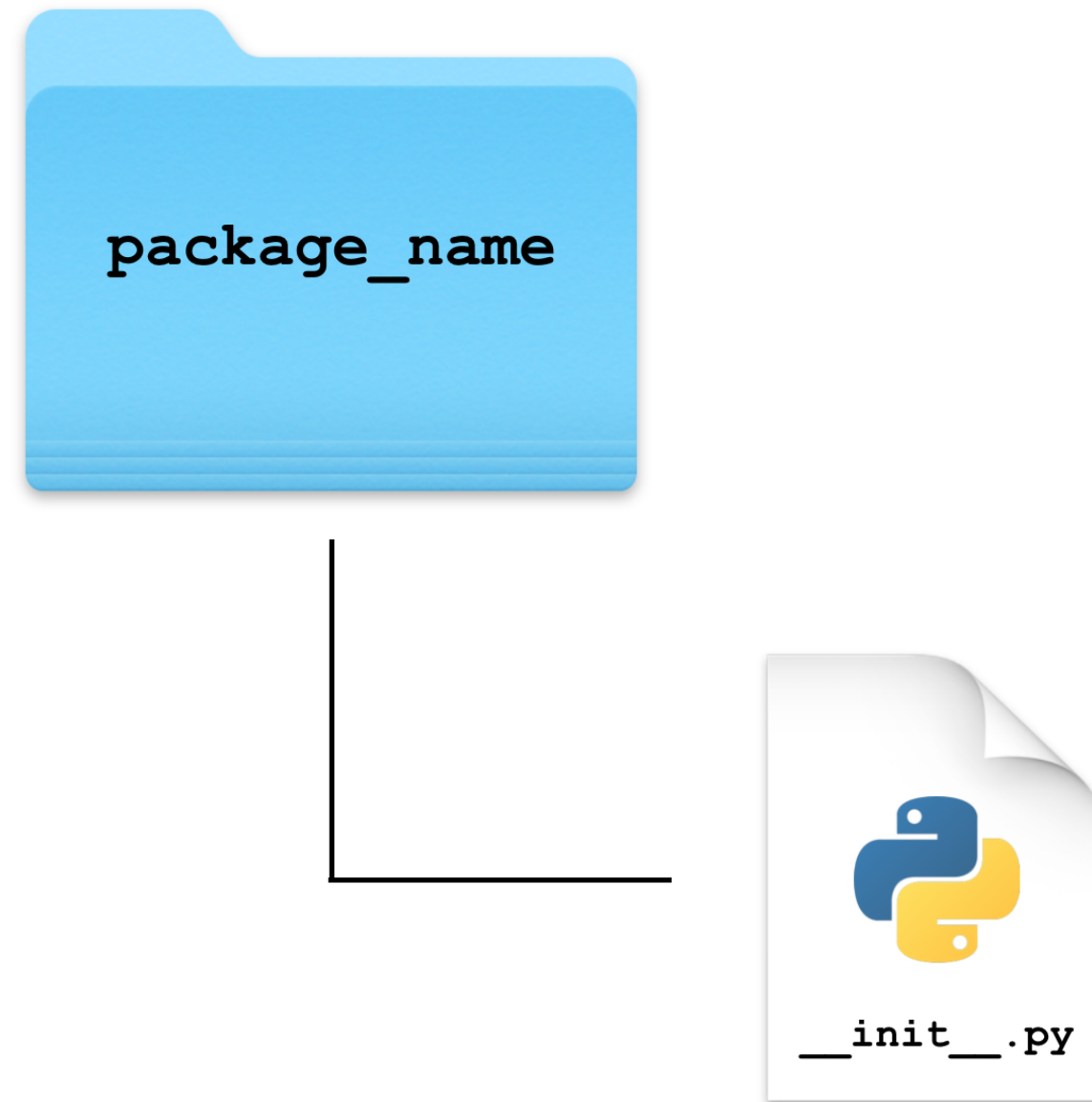
Package structure



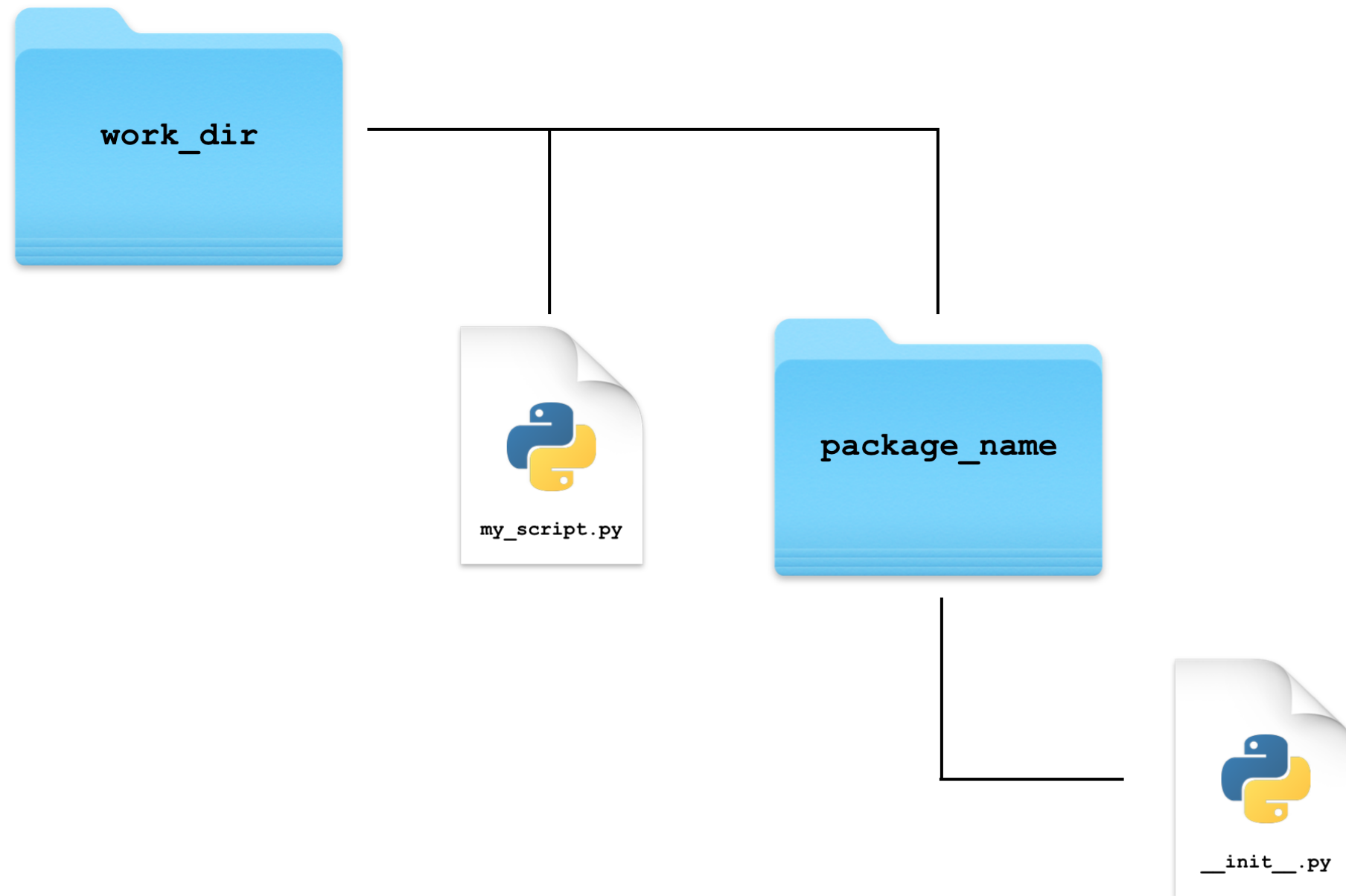
Package structure



Package structure



Importing a local package



Importing a local package

```
import my_package  
help(my_package)
```

```
Help on package my_package:
```

```
NAME
```

```
    my_package
```

```
PACKAGE CONTENTS
```

```
FILE
```

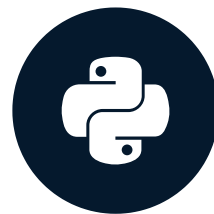
```
    ~/work_dir/my_package/__init__.py
```

Let's Practice

SOFTWARE ENGINEERING PRINCIPLES IN PYTHON

Adding Functionality to Packages

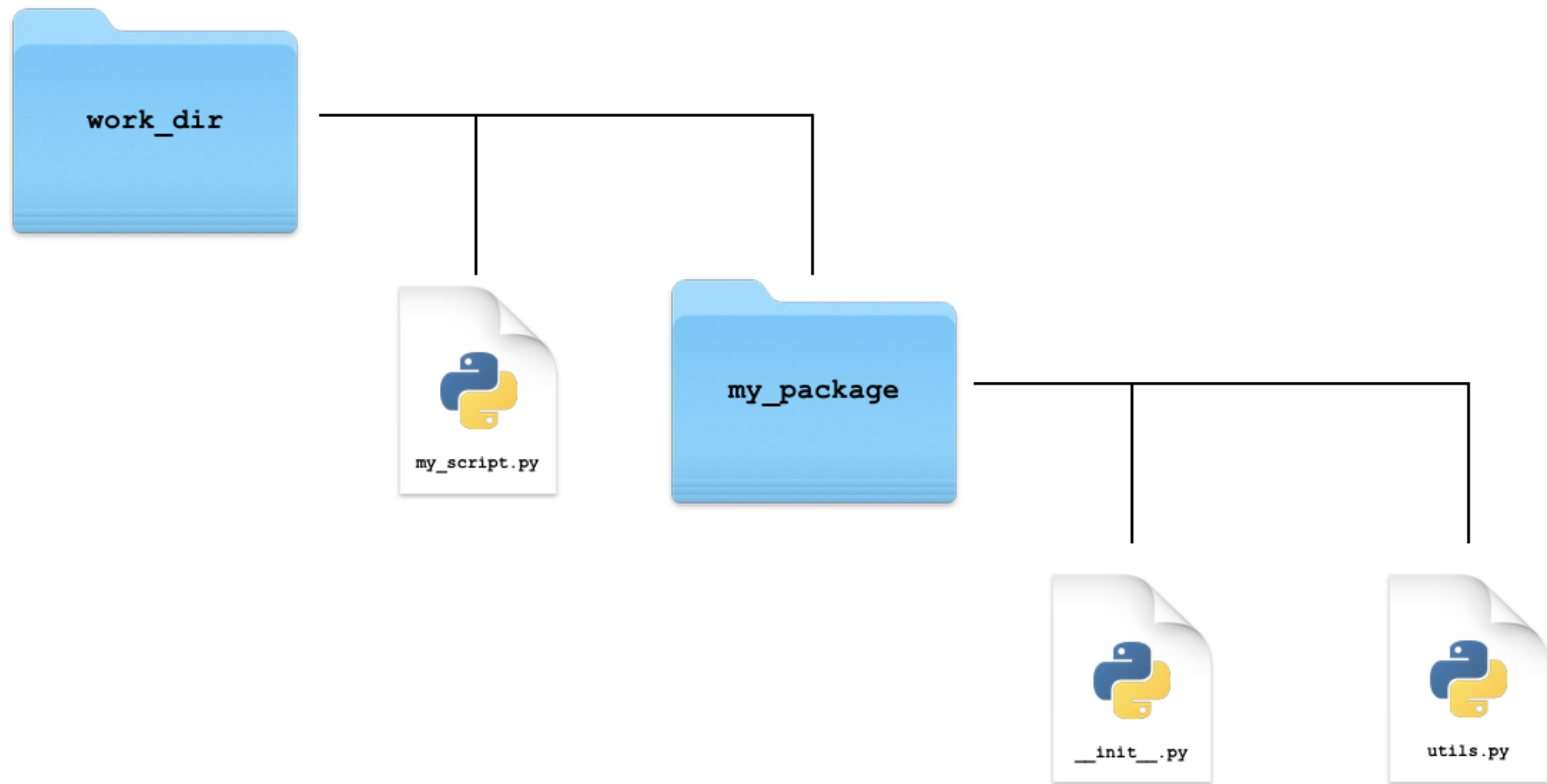
SOFTWARE ENGINEERING PRINCIPLES IN PYTHON



Adam Spannbauer

Machine Learning Engineer at Eastman

Package structure



Adding functionality

working in `work_dir/my_package/utils.py`

```
def we_need_to_talk(break_up=False):  
    """Helper for communicating with significant other"""  
    if break_up:  
        print("It's not you, it's me...")  
    else:  
        print('I <3 U!')
```

working in `work_dir/my_script.py`

```
# Import utils submodule  
import my_package.utils  
  
# Decide to start seeing other people  
my_package.utils.we_need_to_talk(break_up=True)
```

```
It's not you, it's me...
```

Importing functionality with `__init__.py`

working in `work_dir/my_package/__init__.py`

```
from .utils import we_need_to_talk
```

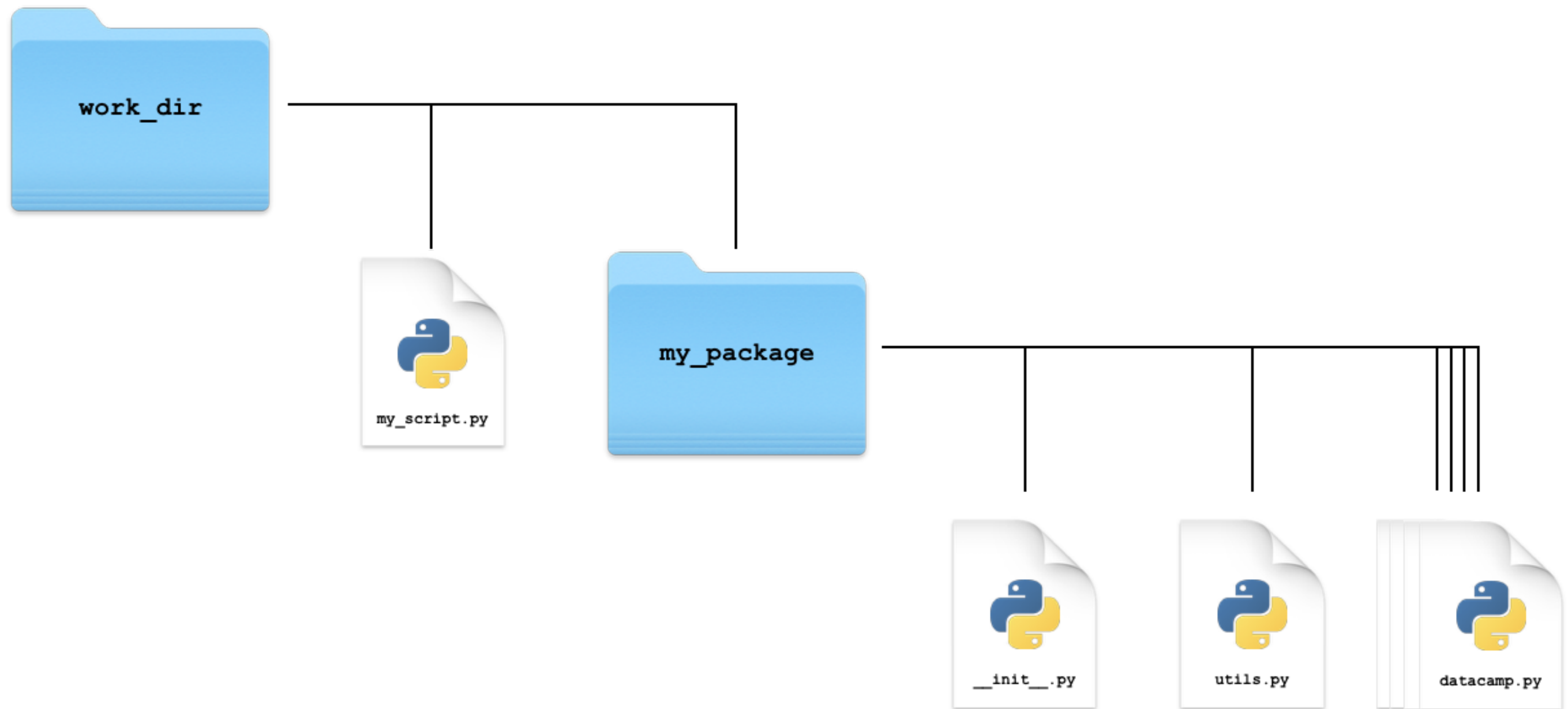
working in `work_dir/my_script.py`

```
# Import custom package
import my_package

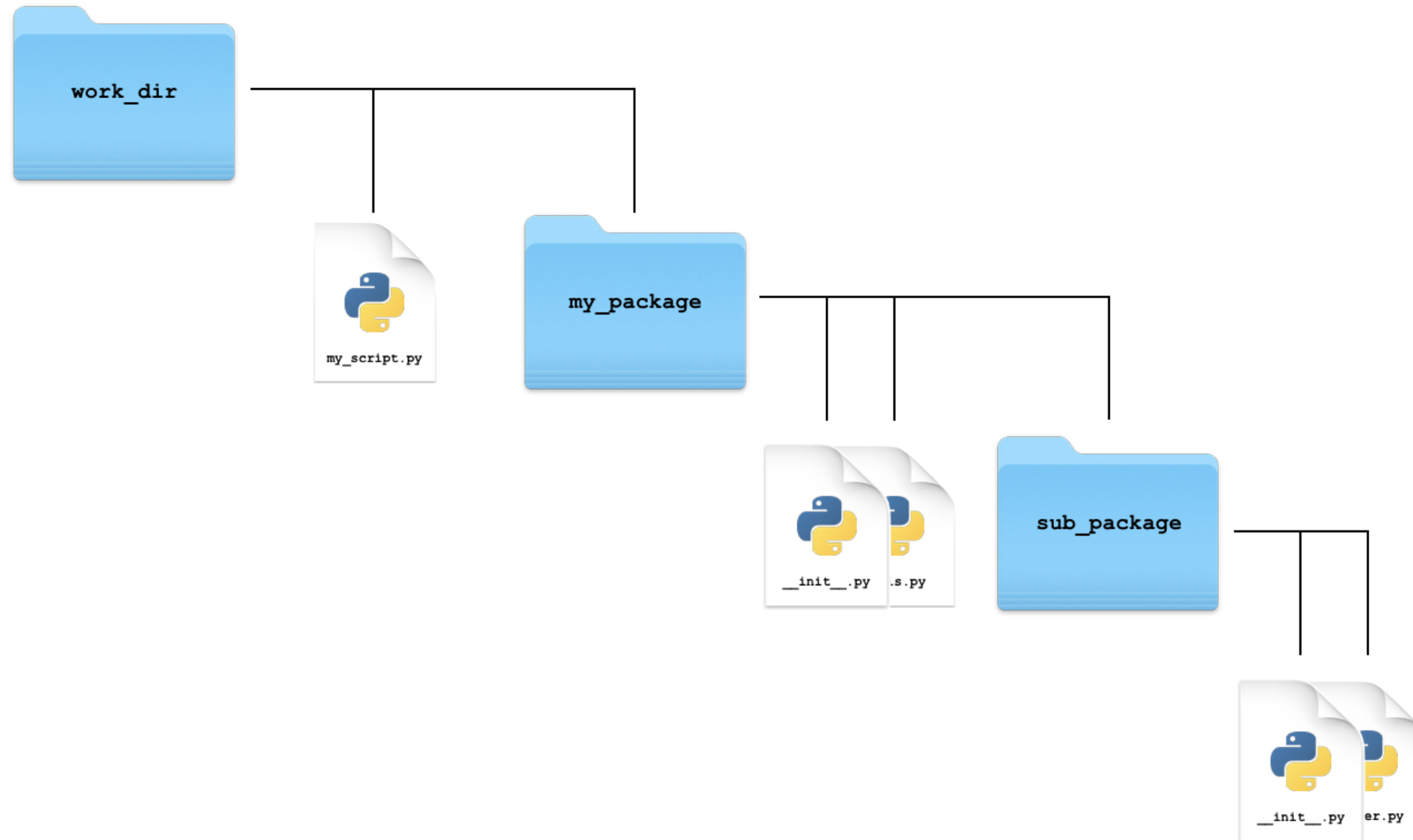
# Realize you're with your soulmate
my_package.we_need_to_talk(break_up=False)
```

I <3 U!

Extending package structure



Extending package structure

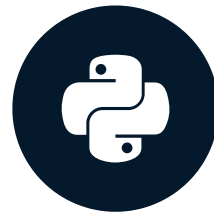


Let's Practice

SOFTWARE ENGINEERING PRINCIPLES IN PYTHON

Making your package portable

SOFTWARE ENGINEERING PRINCIPLES IN PYTHON



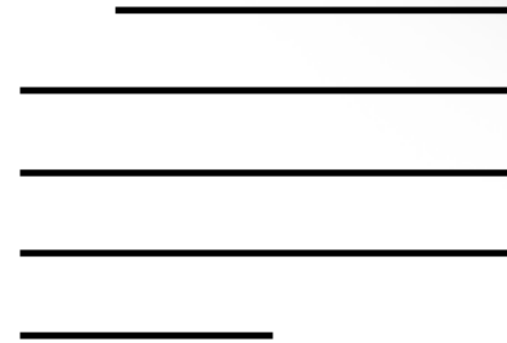
Adam Spannbauer

Machine Learning Engineer at Eastman

Steps to portability

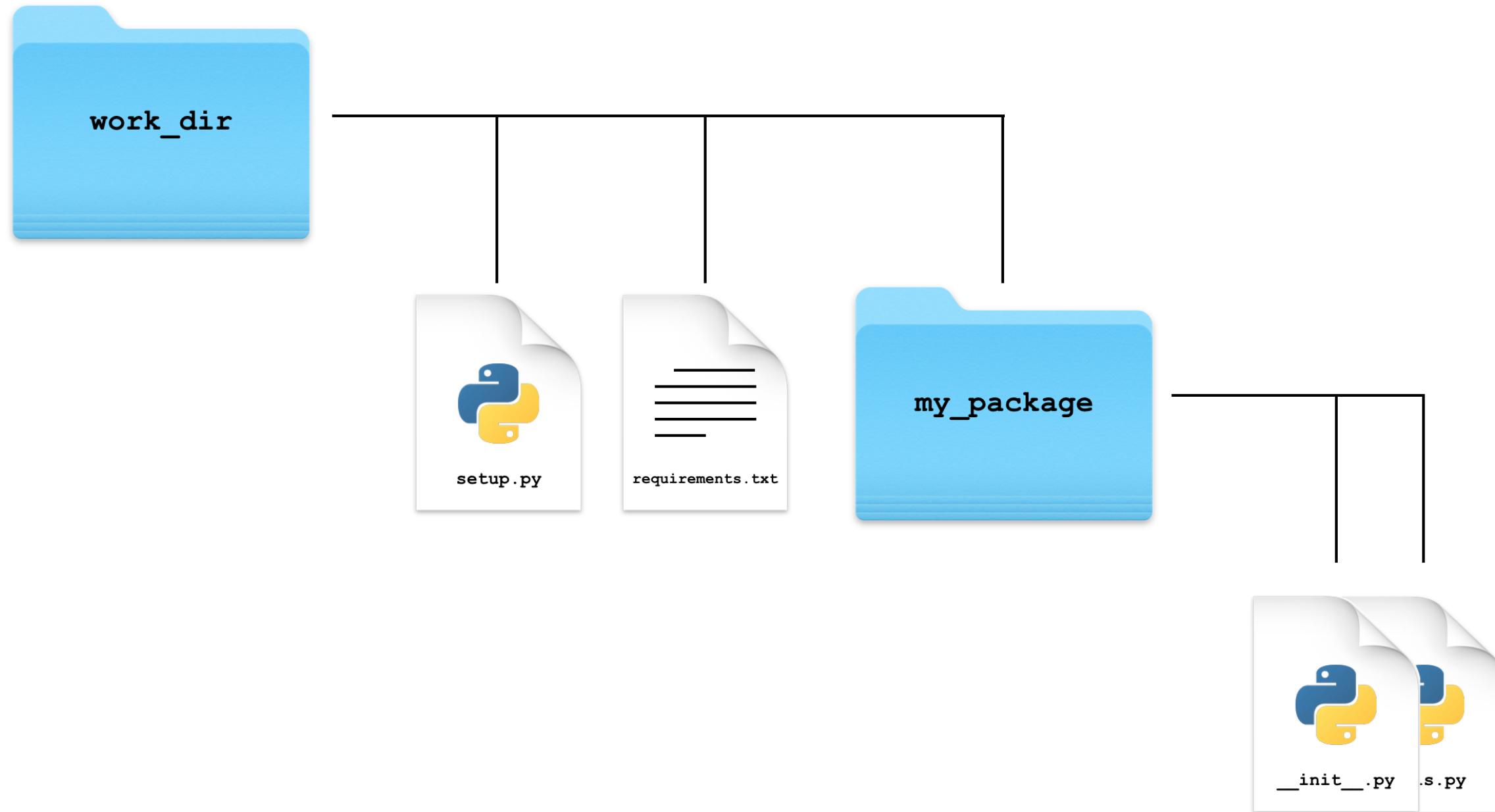


`setup.py`



`requirements.txt`

Portable package structure



Contents of requirements.txt

working in `work_dir/requirements.txt`

```
# Needed packages/versions  
matplotlib  
numpy==1.15.4  
pycodestyle>=2.4.0
```

working with `terminal`

```
datacamp@server:~$ pip install -r requirements.txt
```

Contents of setup.py

```
from setuptools import setup

setup(name='my_package',
      version='0.0.1',
      description='An example package for DataCamp.',
      author='Adam Spannauer',
      author_email='spannbaueradam@gmail.com',
      packages=['my_package'],
      install_requires=['matplotlib',
                       'numpy==1.15.4',
                       'pycodestyle>=2.4.0'])
```

install_requires vs requirements.txt

working in `work_dir/requirements.txt`

```
# Specify where to install requirements from
--index-url https://pypi.python.org/simple/

# Needed packages/versions
matplotlib
numpy==1.15.4
pycodestyle>=2.4.0
```

Documentation: [install_requires vs requirements files](#)

pip installing your package

```
datacamp@server:~/work_dir $ pip install .
```

```
Building wheels for collected packages: my-package  
  Running setup.py bdist_wheel for my-package ... done  
Successfully built my-package  
Installing collected packages: my-package  
Successfully installed my-package-0.0.1
```

Let's Practice

SOFTWARE ENGINEERING PRINCIPLES IN PYTHON