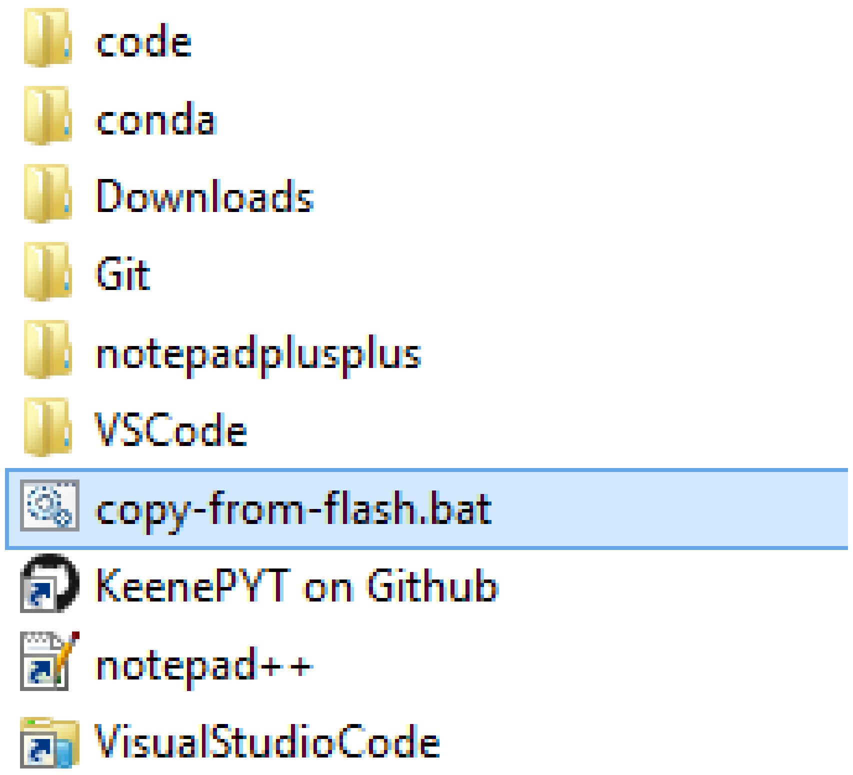
Help us get started.

1. Get a flash drive.
2. Run **copy-from-flash.bat**.

This will create **C:\Projects\NEARC**.

The same code is available at **https://github.com/jswise/keenepyt**.



code
conda
Downloads
Git
notepadplusplus
VSCode
copy-from-flash.bat
KeenePYT on Github
notepad++
VisualStudioCode

Distribute Your Python Tools to ArcGIS Pro Users

Jason Wise, Earth Data Scientist, Terracon

Distribute Your Python Tools to ArcGIS Pro Users

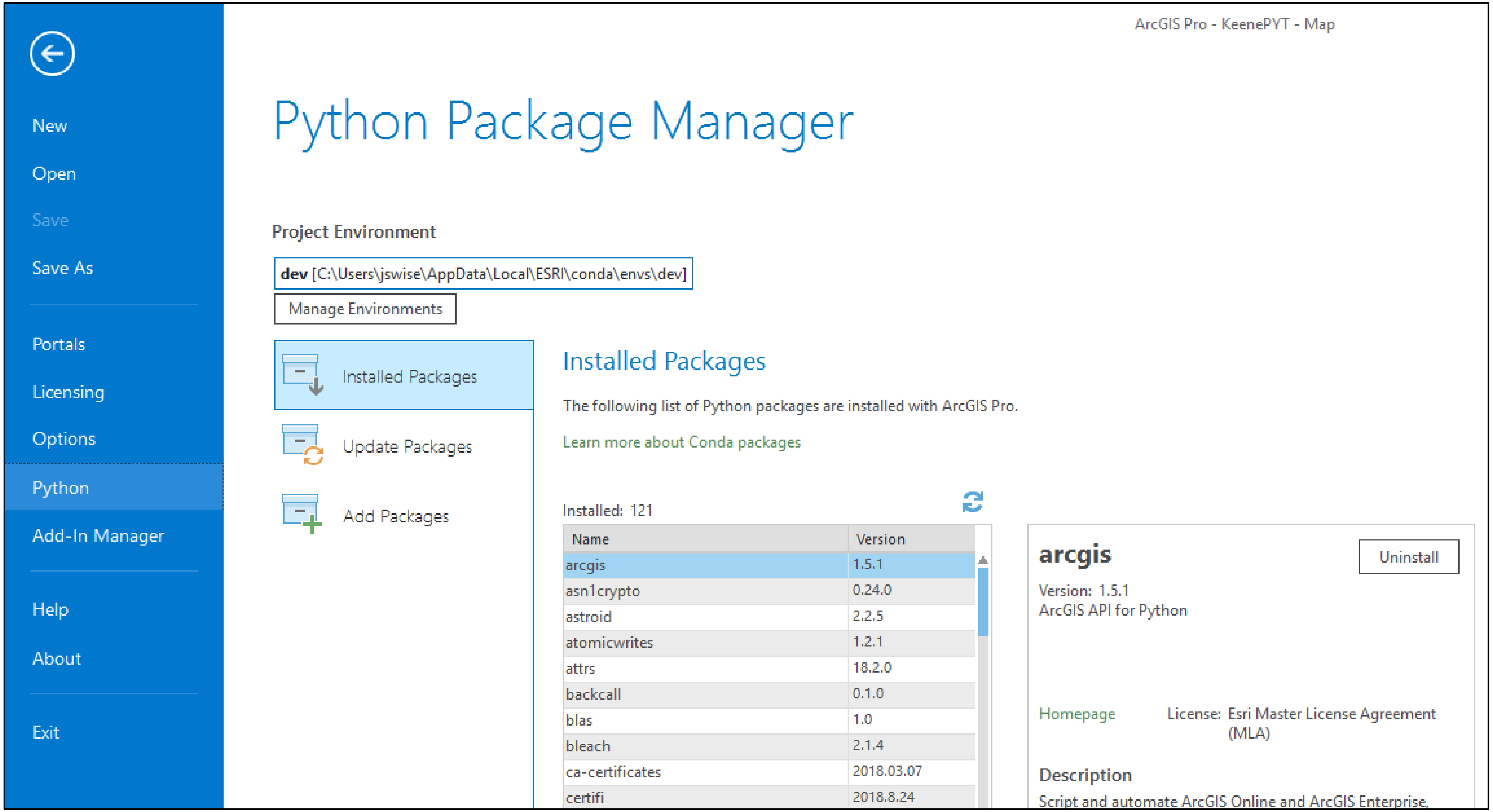
Jason Wise, Earth Data Scientist

Terracon

Northeast Arc User Group (NEARC)
Spring 2019

The Python Package Manager

It manages “environments” and “packages.” We’ll talk about what those are.



Python Package Manager

Project Environment

dev [C:\Users\jswise\AppData\Local\ESRI\conda\envs\dev]

Installed Packages

The following list of Python packages are installed with ArcGIS Pro.

Learn more about Conda packages

Installed: 121

Name	Version
arcgis	1.5.1
asn1crypto	0.24.0
astroid	2.2.5
atomicwrites	1.2.1
attrs	18.2.0
backcall	0.1.0
blas	1.0
bleach	2.1.4
ca-certificates	2018.03.07
certifi	2018.8.24

arcgis

Version: 1.5.1
ArcGIS API for Python

Homepage

License: Esri Master License Agreement (MLA)

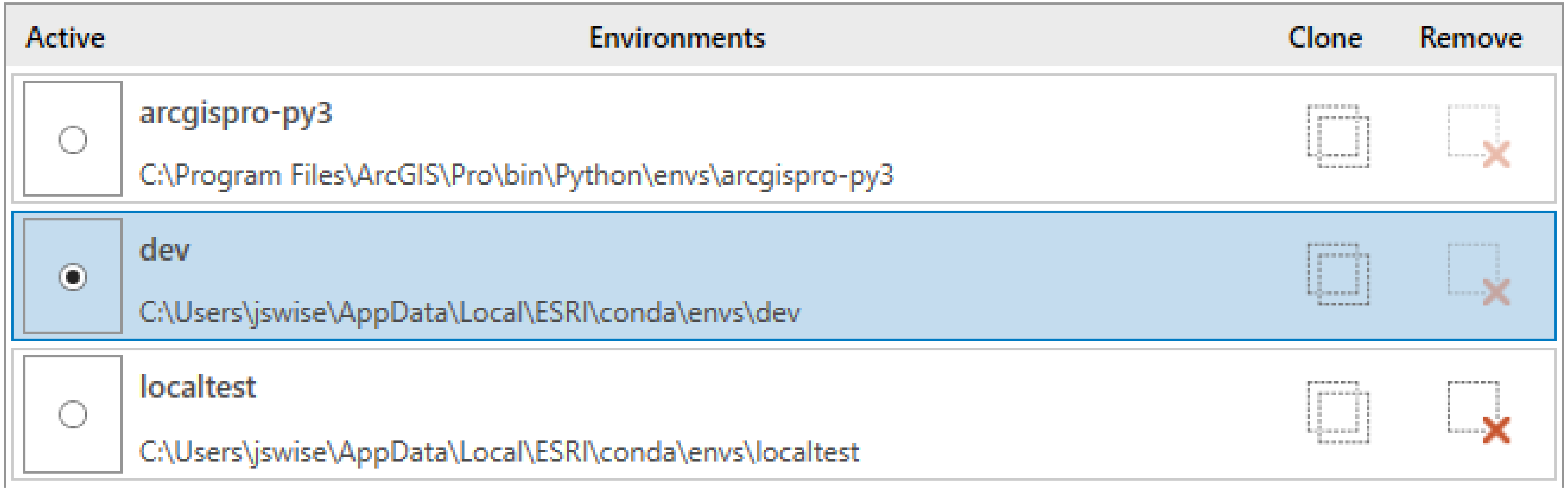
Description

Script and automate ArcGIS Online and ArcGIS Enterprise,

Create dev & test environments.

In **C:\Projects\NEARC\code\keenepyt\batch**, run **create-dev-envs.bat.**

This will clone your default Python environment to two new environments, *dev* and *localtest*. It takes a while to run.



Active	Environments	Clone	Remove
○	arcgispro-py3 C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3		
●	dev C:\Users\jswise\AppData\Local\ESRI\conda\envs\dev		
○	localtest C:\Users\jswise\AppData\Local\ESRI\conda\envs\localtest		

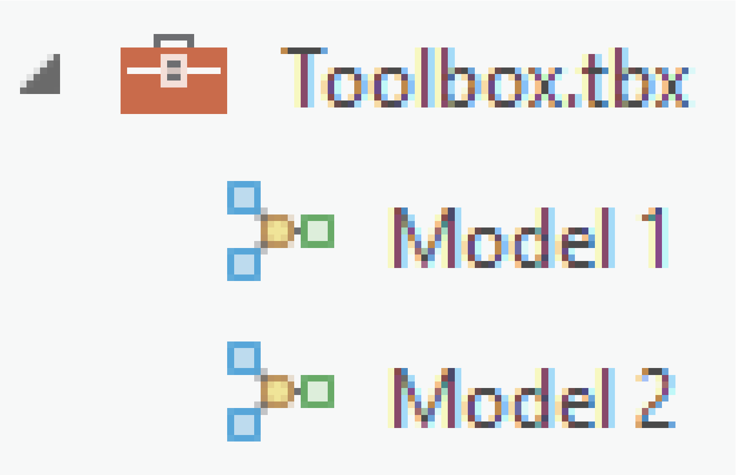
The goals:

1. Give a geoprocessing toolbox to other users.
2. Easily update the toolbox.

With traditional methods (e.g. passing out .TBX files and Python scripts), updating is the hard part.

The Toolbox Menagerie

A traditional toolbox is a binary file.

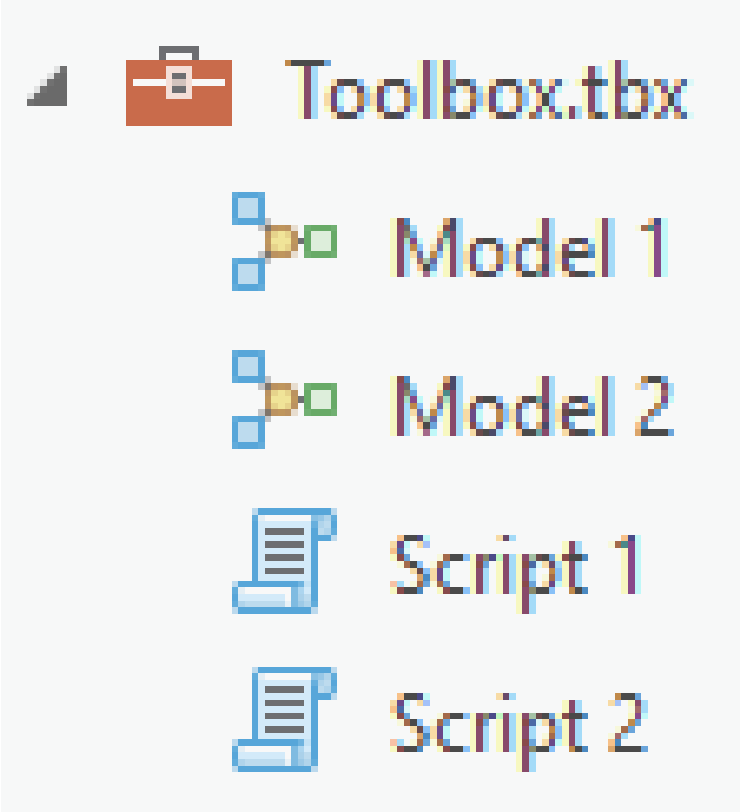


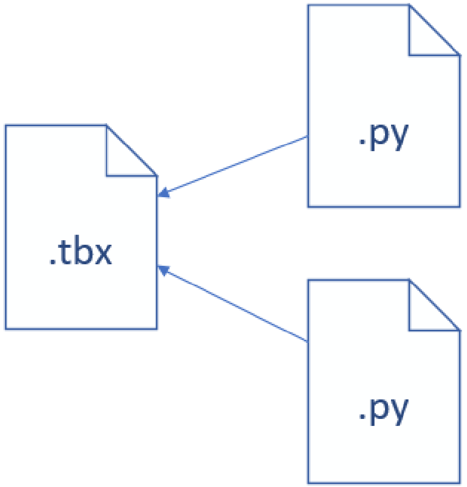
Toolbox.tbx

- Model 1
- Model 2

The Toolbox Menagerie

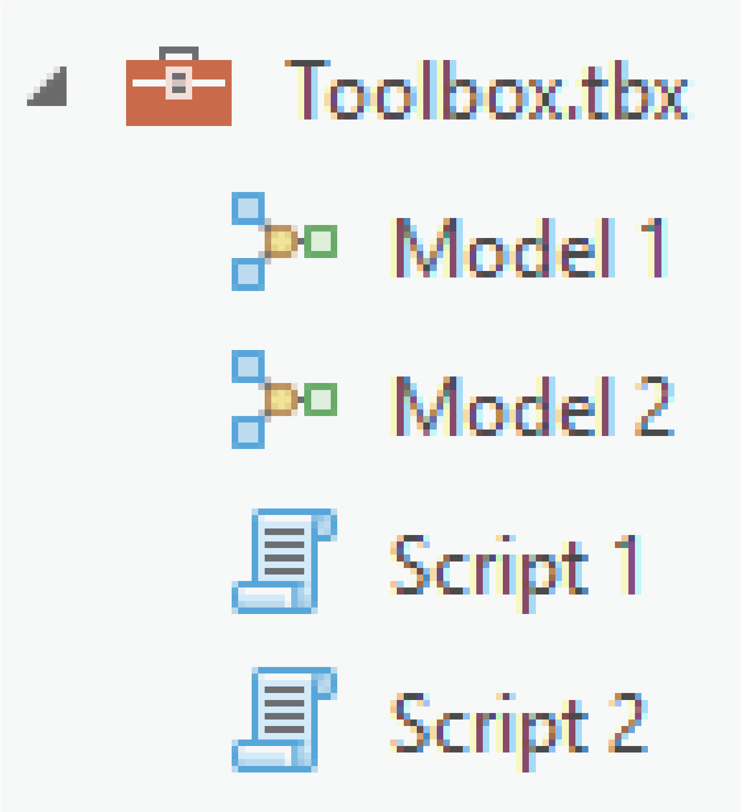
Each script is a Python module (file).





The Toolbox Menagerie

Each Python module can import other modules & packages.



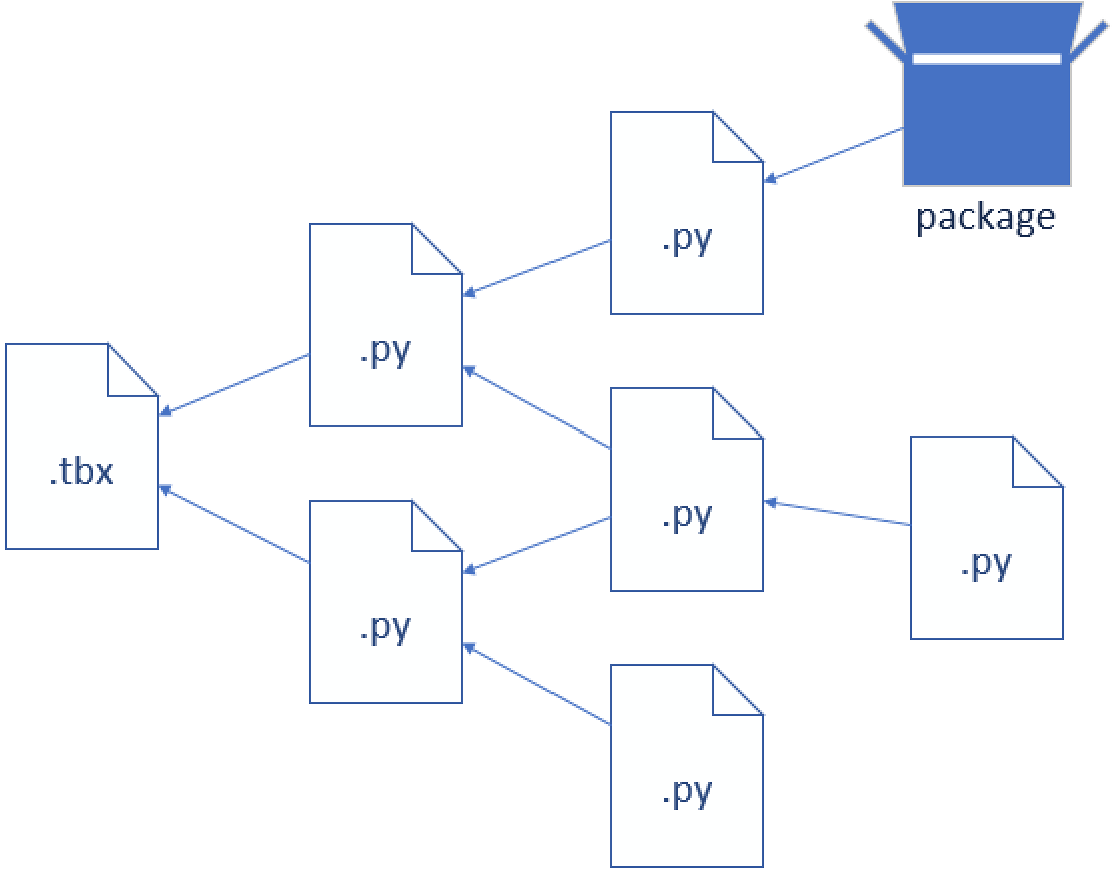
Toolbox.tbx

Model 1

Model 2

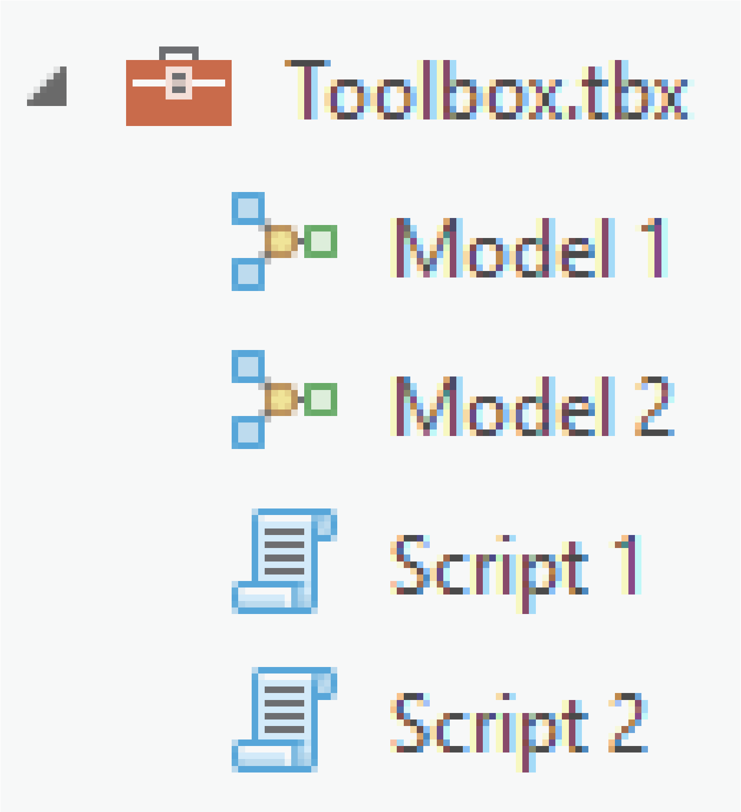
Script 1

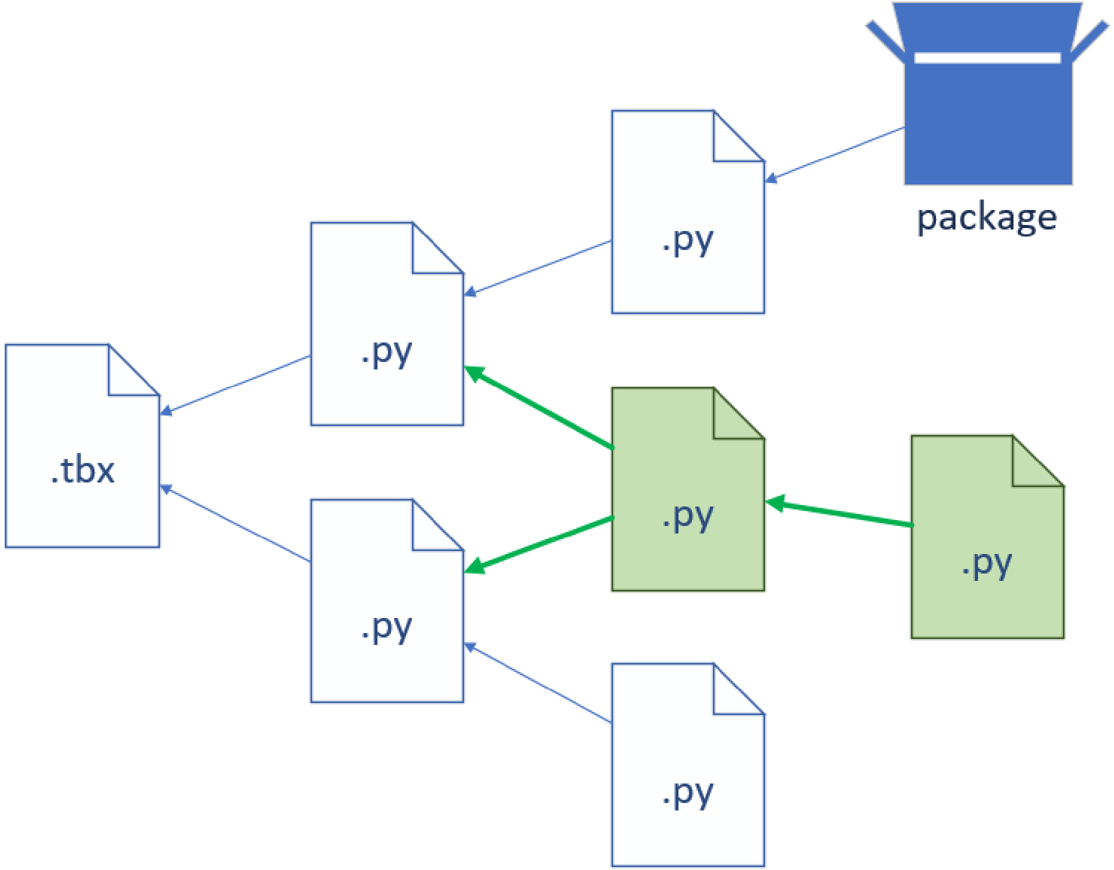
Script 2



The Toolbox Menagerie

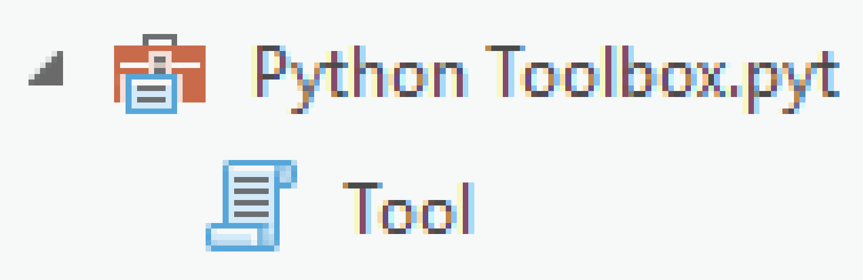
Reusing code is good!





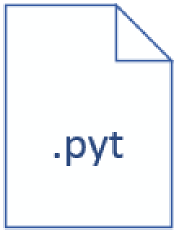
The Toolbox Menagerie

A Python toolbox is a Python file with a goofy extension.
It can be self-contained…



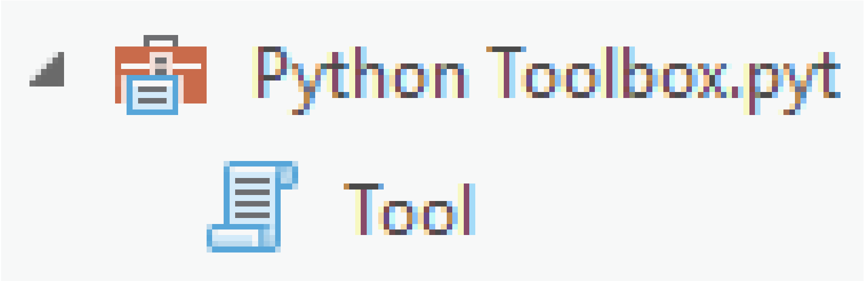
Python Toolbox.pyt

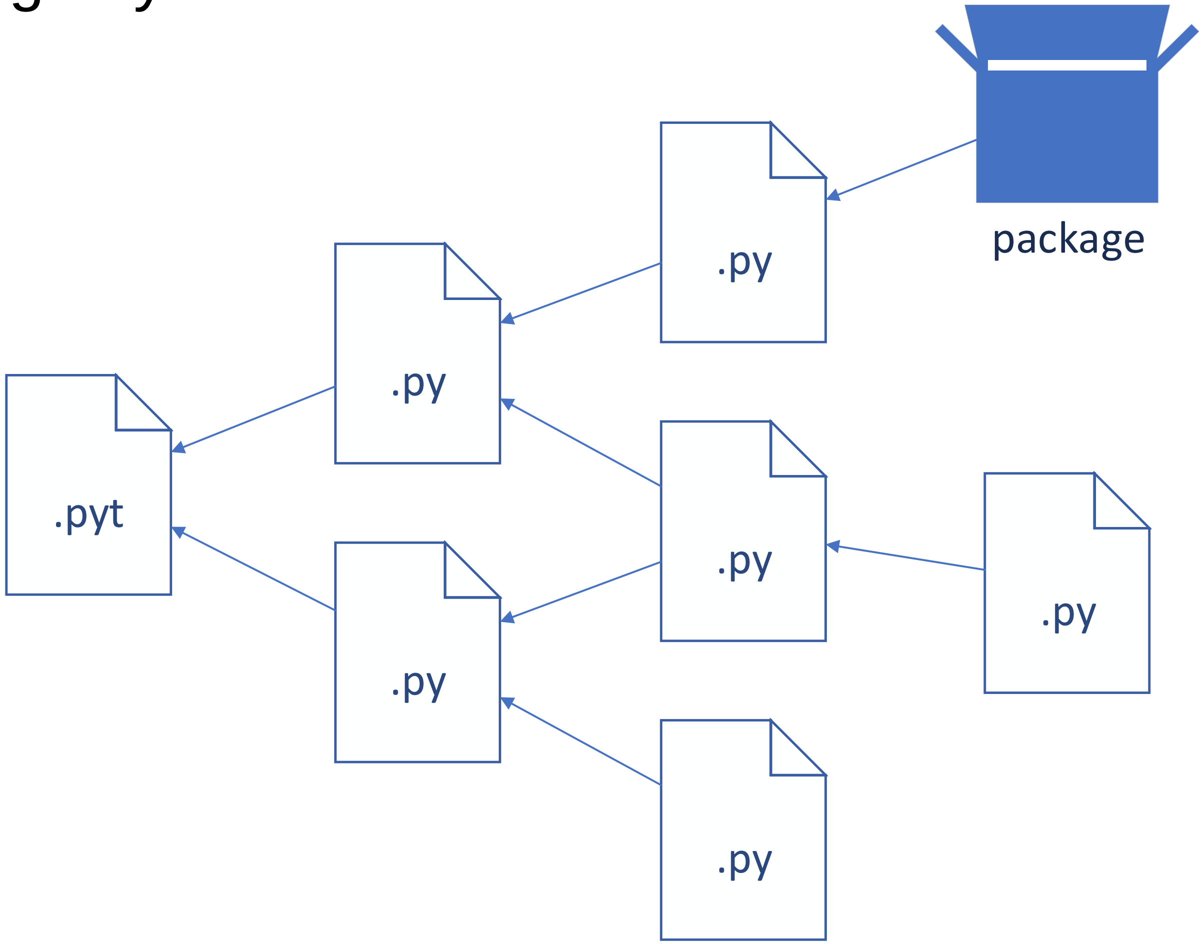
Tool



The Toolbox Menagerie

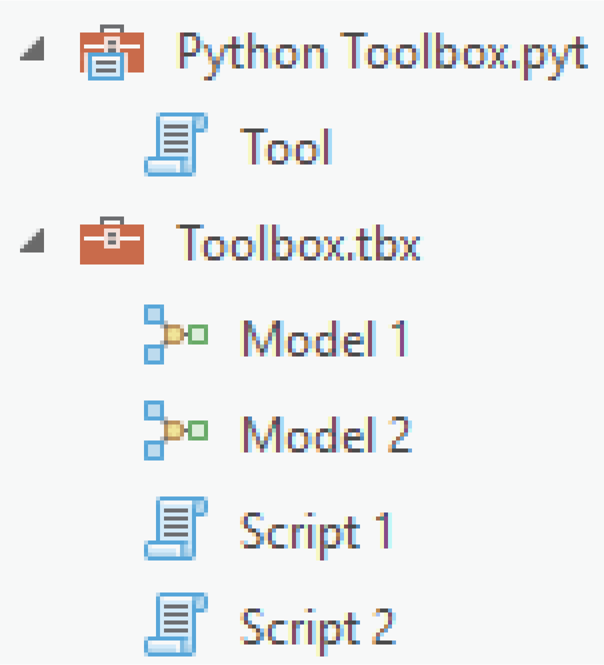
A Python toolbox is a Python file with a goofy extension.
It can be self-contained…
…or import other modules & packages.



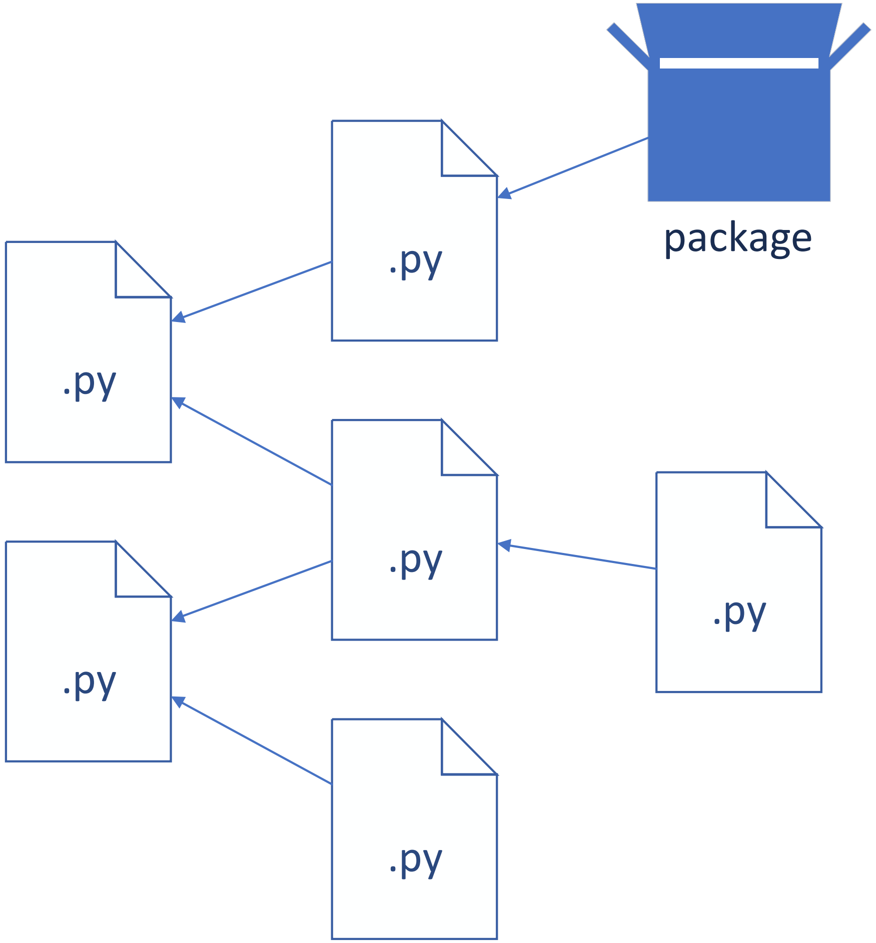


The Toolbox Menagerie

Distributing these to users can be a pain.
Updating them is worse.

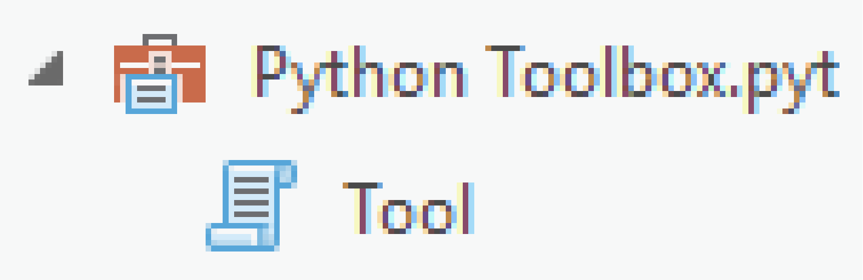


- Python Toolbox.pyt
  - Tool
- Toolbox.tbx
  - Model 1
  - Model 2
  - Script 1
  - Script 2



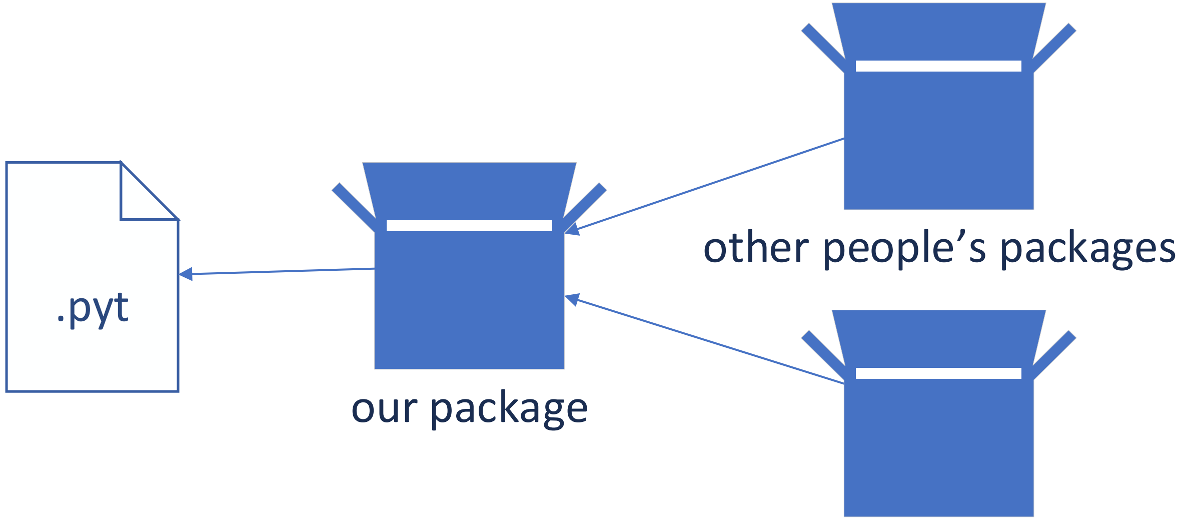
Our approach

We'll put all our code in a package, then import it into a Python toolbox.



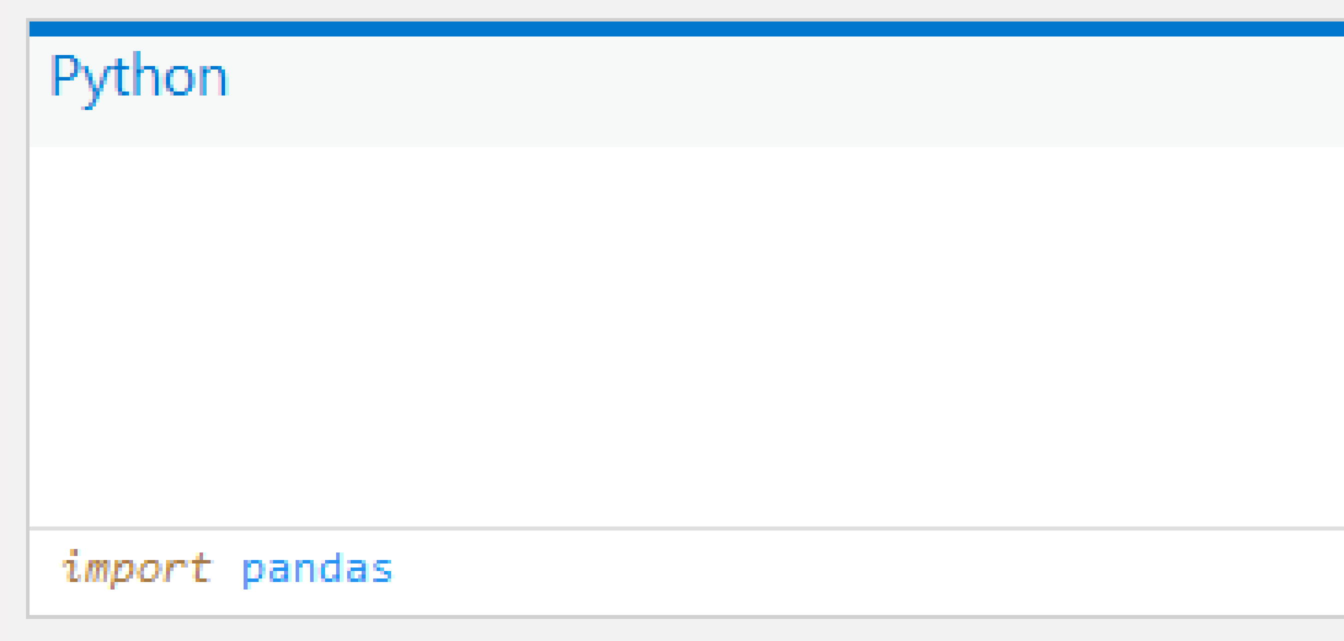
Python Toolbox.pyt

Tool



Python packages in Pro

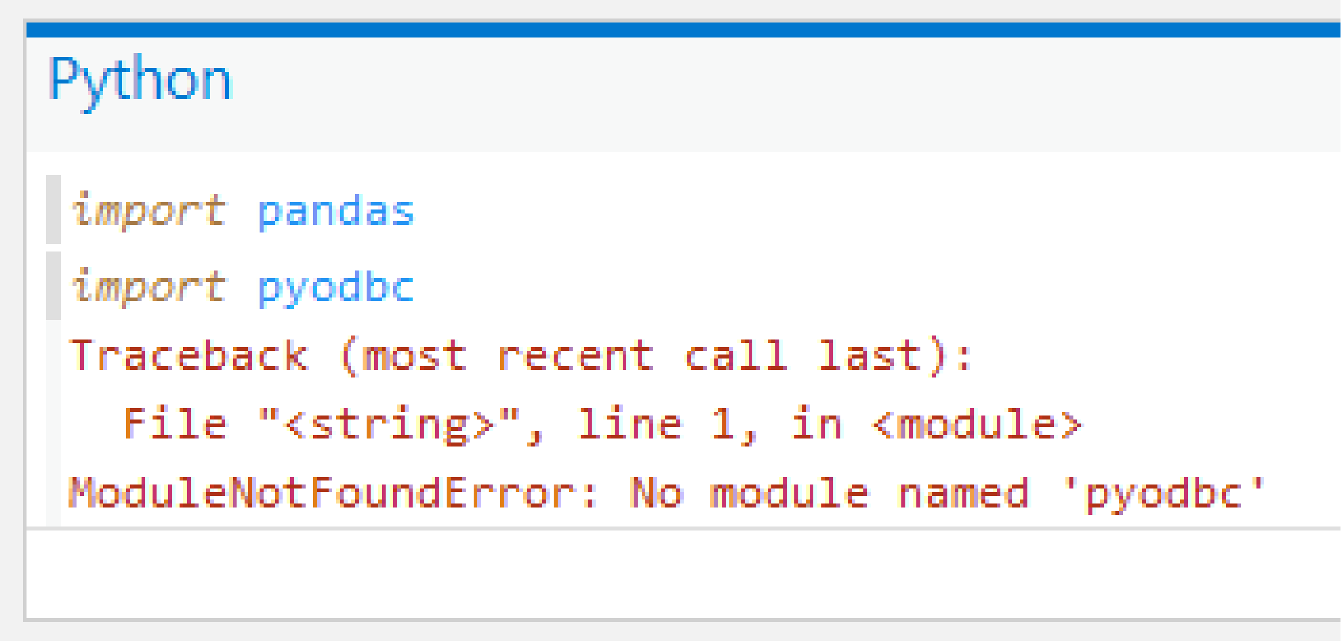
ArcGIS Pro comes with lots of handy Python packages that you can import into your code…



```
import pandas
```

Python packages in Pro

ArcGIS Pro comes with lots of handy Python packages that you can import into your code…
…but not all of them.

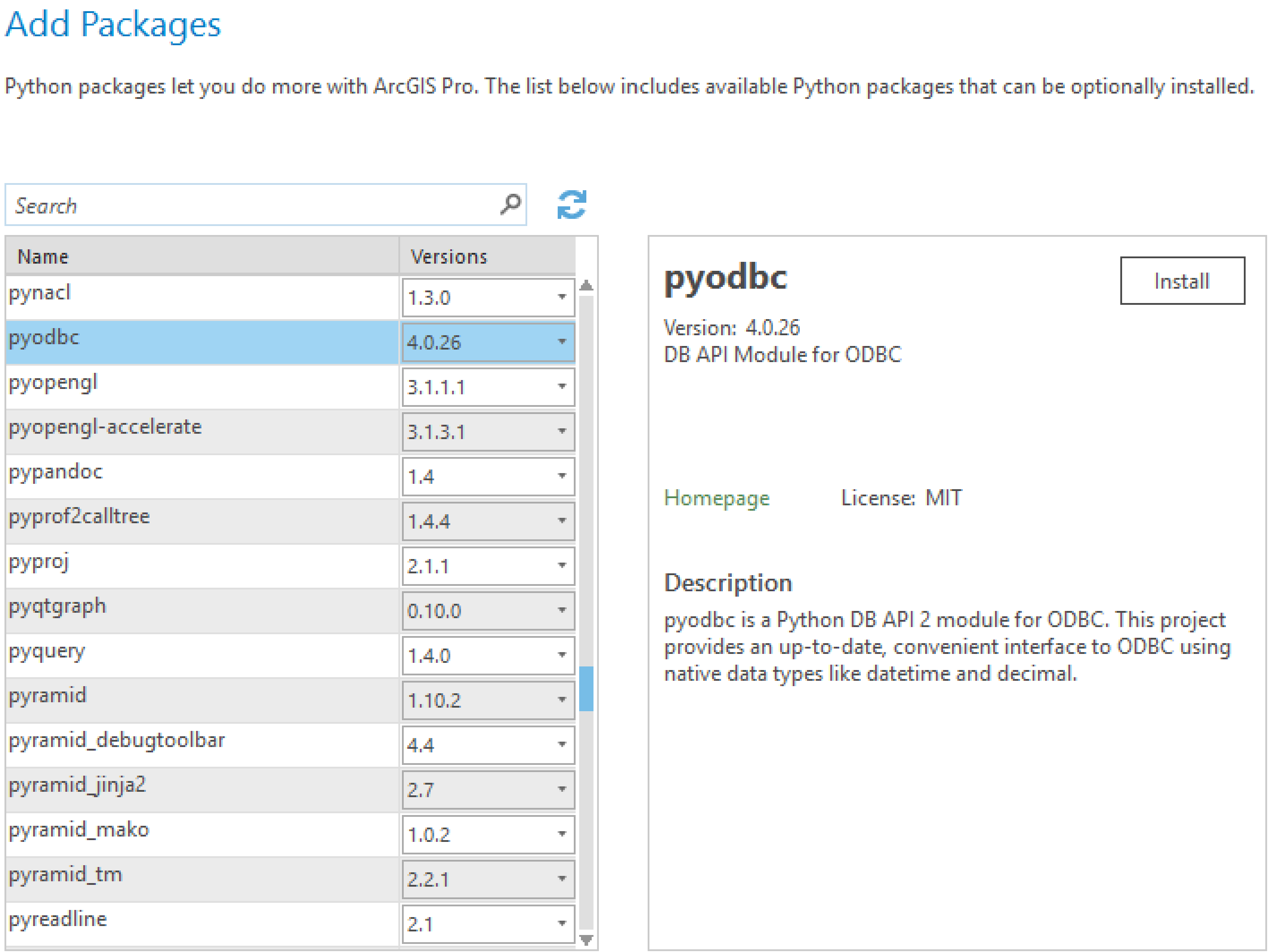


```
import pandas
import pyodbc
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ModuleNotFoundError: No module named 'pyodbc'
```

Terracon

Python packages in Pro

The Python Package Manager makes it easy to get more of them.



Add Packages

Python packages let you do more with ArcGIS Pro. The list below includes available Python packages that can be optionally installed.

Name	Versions
pynacl	1.3.0
pyodbc	4.0.26
pyopengl	3.1.1.1
pyopengl-accelerate	3.1.3.1
pypandoc	1.4
pyprof2calltree	1.4.4
pyproj	2.1.1
pyqtgraph	0.10.0
pyquery	1.4.0
pyramid	1.10.2
pyramid_debugtoolbar	4.4
pyramid_jinja2	2.7
pyramid_mako	1.0.2
pyramid_tm	2.2.1
pyreadline	2.1

pyodbc

Version: 4.0.26
DB API Module for ODBC

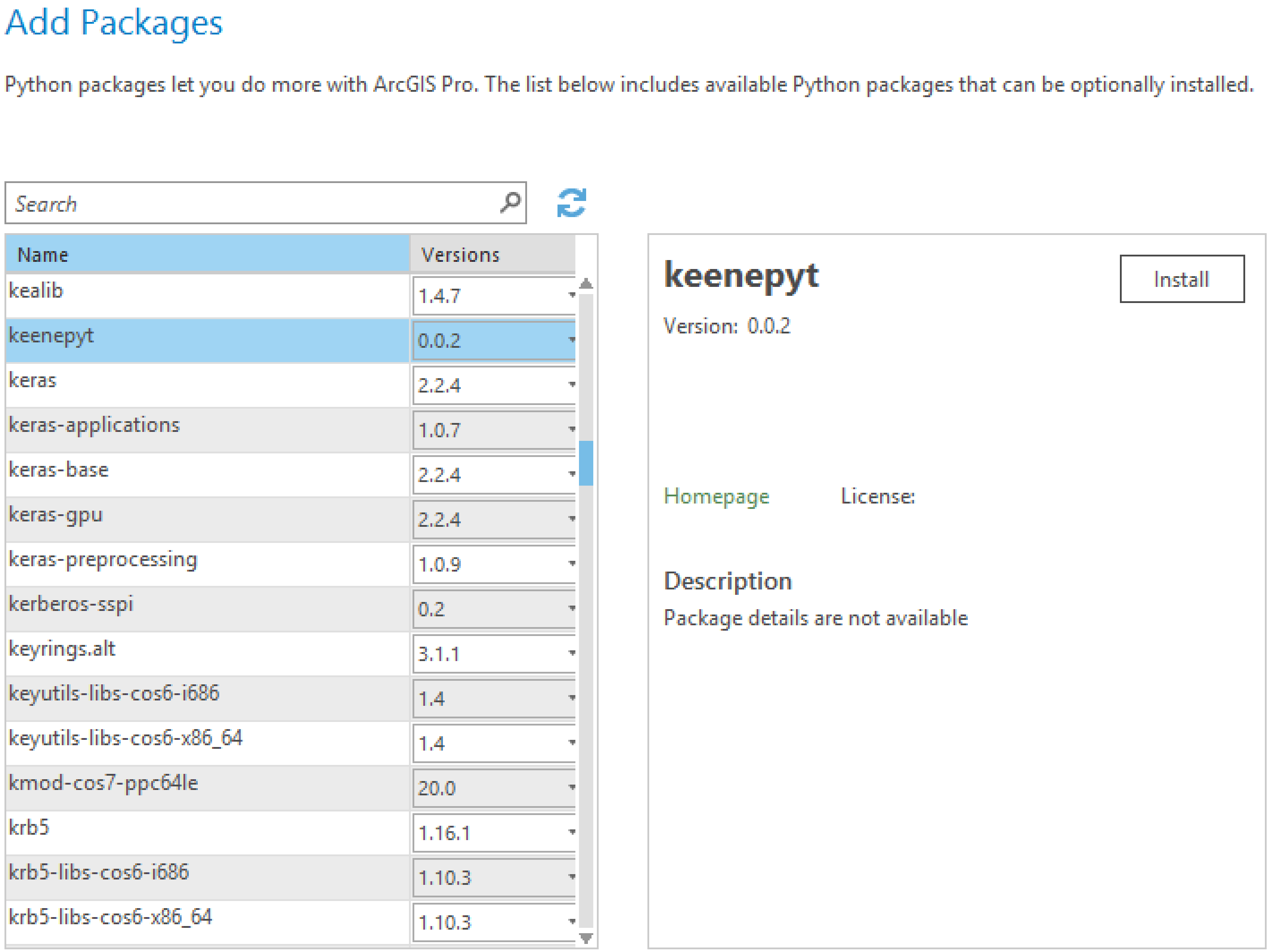
Homepage License: MIT

Description

pyodbc is a Python DB API 2 module for ODBC. This project provides an up-to-date, convenient interface to ODBC using native data types like datetime and decimal.

Python packages in Pro

How about making your own package?



Add Packages

Python packages let you do more with ArcGIS Pro. The list below includes available Python packages that can be optionally installed.

Name	Versions
kealib	1.4.7
keenepyt	0.0.2
keras	2.2.4
keras-applications	1.0.7
keras-base	2.2.4
keras-gpu	2.2.4
keras-preprocessing	1.0.9
kerberos-sspi	0.2
keyrings.alt	3.1.1
keyutils-libs-cos6-i686	1.4
keyutils-libs-cos6-x86_64	1.4
kmod-cos7-ppc64le	20.0
krb5	1.16.1
krb5-libs-cos6-i686	1.10.3
krb5-libs-cos6-x86_64	1.10.3

keenepyt

Version: 0.0.2

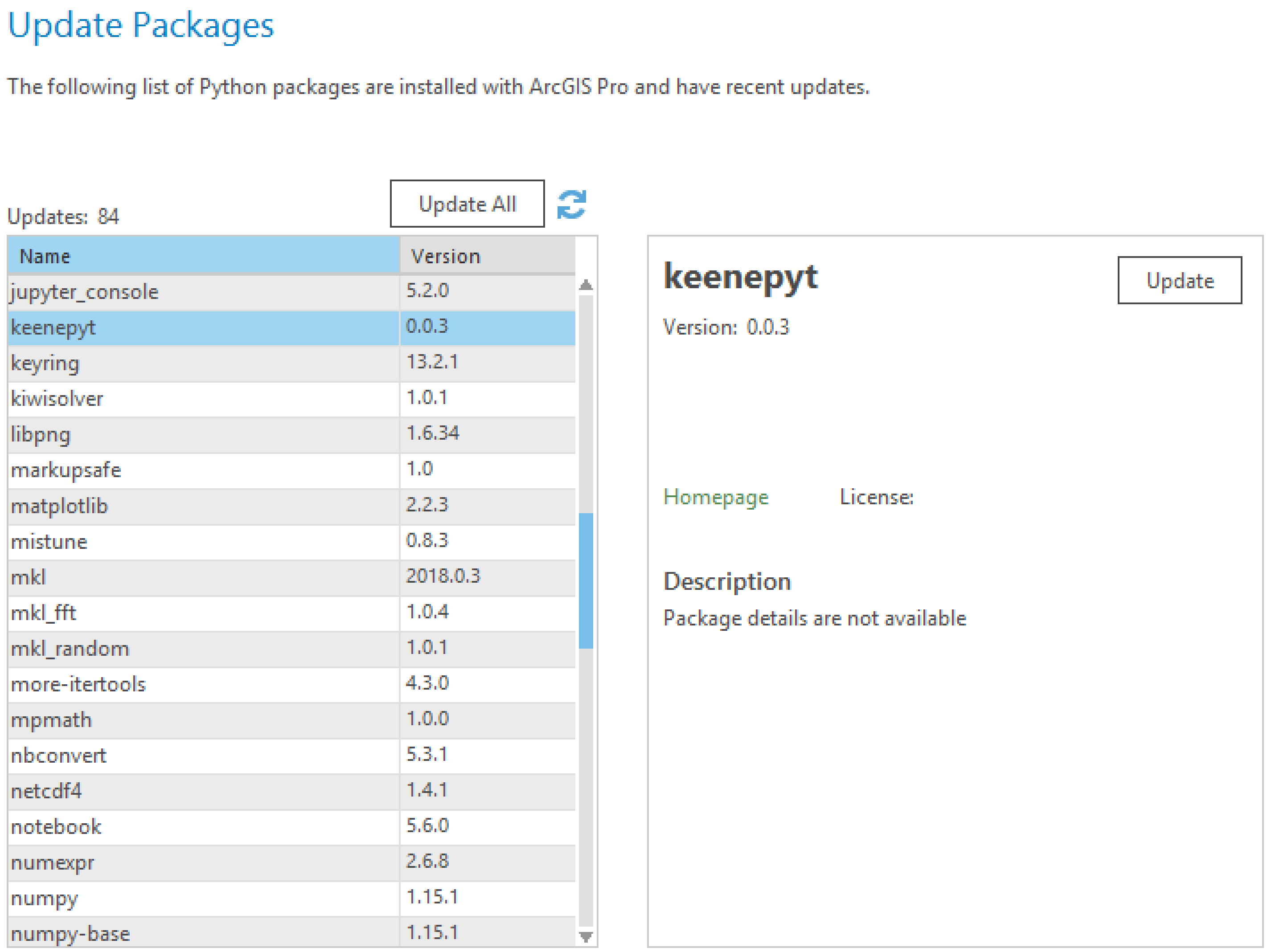
Homepage License:

Description

Package details are not available

Our approach

Updating will be easy.



Update Packages

The following list of Python packages are installed with ArcGIS Pro and have recent updates.

Updates: 84

Name	Version
jupyter_console	5.2.0
keenepyt	0.0.3
keyring	13.2.1
kiwisolver	1.0.1
libpng	1.6.34
markupsafe	1.0
matplotlib	2.2.3
mistune	0.8.3
mkl	2018.0.3
mkl_fft	1.0.4
mkl_random	1.0.1
more-itertools	4.3.0
mpmath	1.0.0
nbconvert	5.3.1
netcdf4	1.4.1
notebook	5.6.0
numexpr	2.6.8
numpy	1.15.1
numpy-base	1.15.1

keenepyt

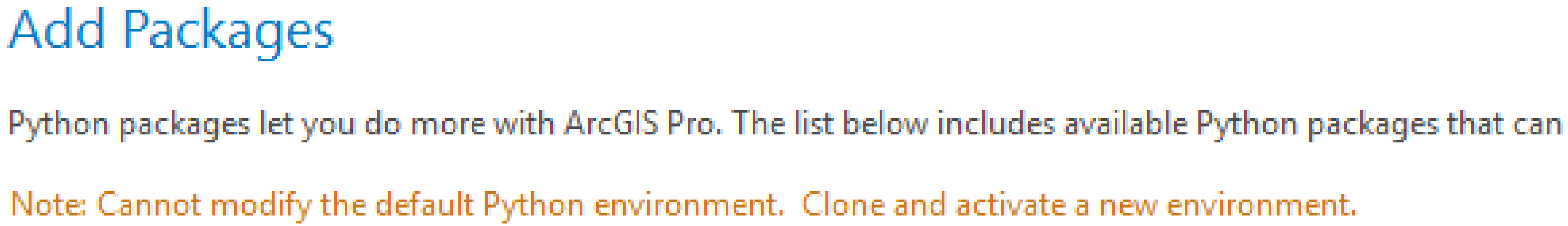
Version: 0.0.3

Description

Package details are not available

Our approach

Esri doesn’t want us to modify the default environment, so we cloned it.



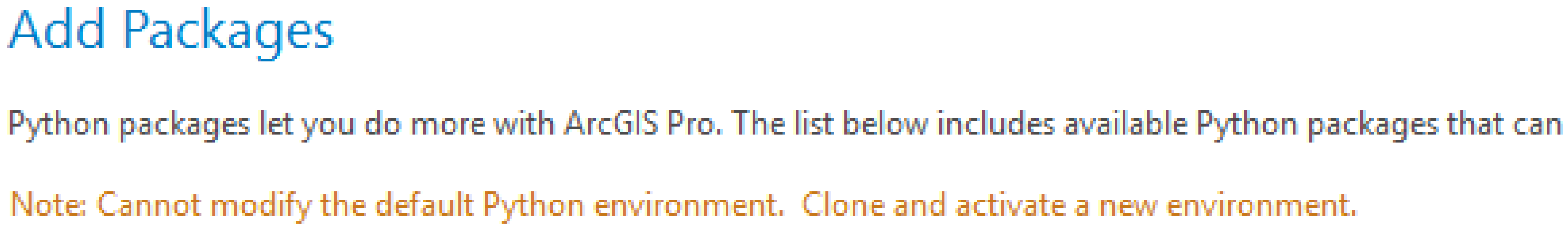
Add Packages

Python packages let you do more with ArcGIS Pro. The list below includes available Python packages that can

Note: Cannot modify the default Python environment. Clone and activate a new environment.

Our approach

Esri doesn’t want us to modify the default environment, so we cloned it.
Think of an “environment” as a Python installation.



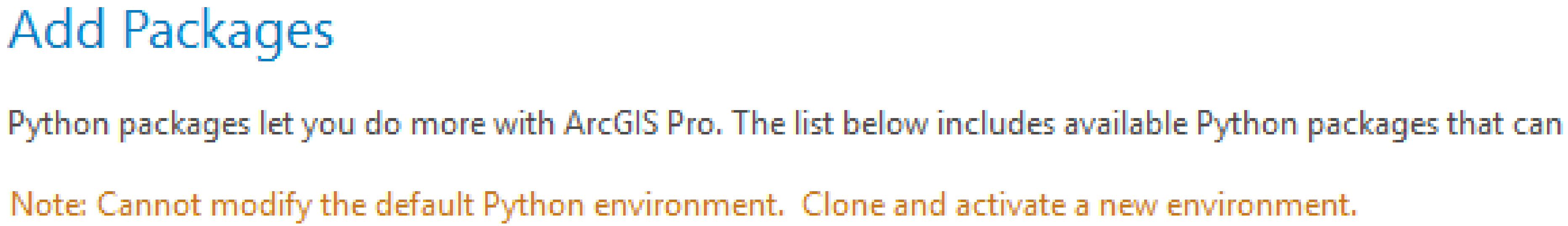
Add Packages

Python packages let you do more with ArcGIS Pro. The list below includes available Python packages that can

Note: Cannot modify the default Python environment. Clone and activate a new environment.

Our approach

Esri doesn’t want us to modify the default environment, so we cloned it.



Add Packages

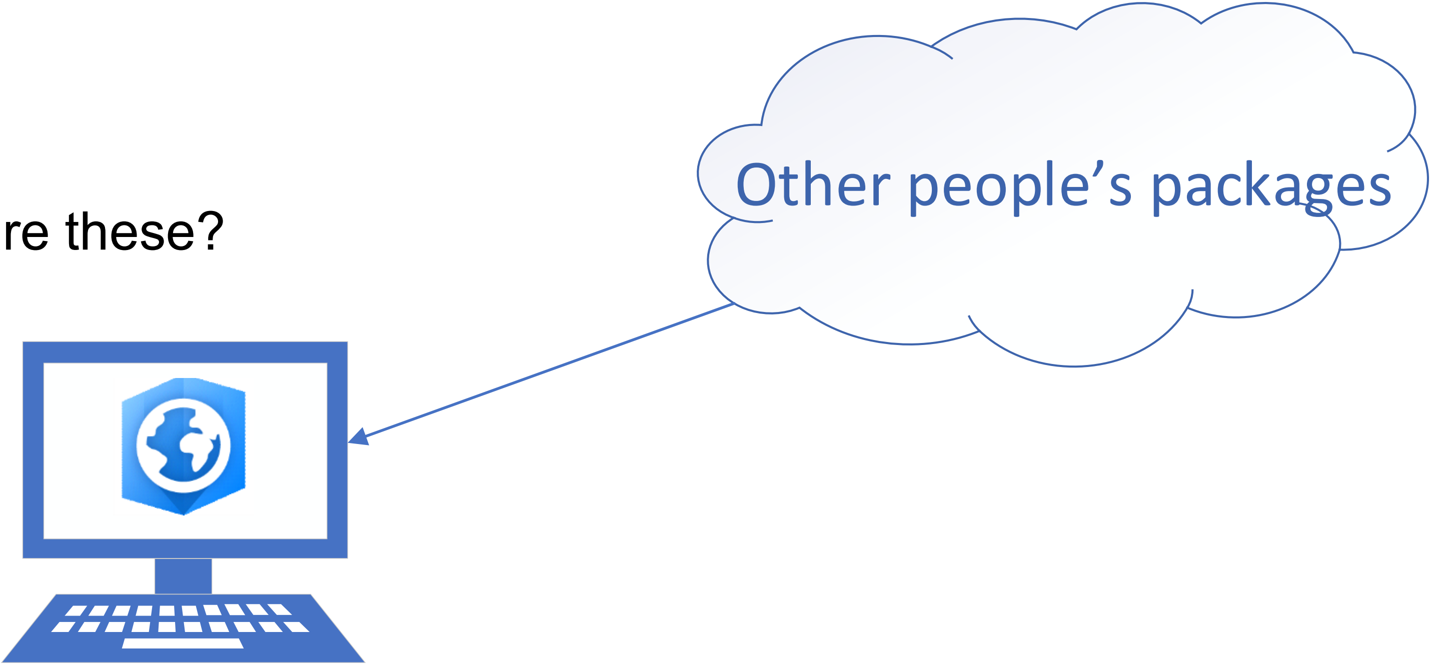
Python packages let you do more with ArcGIS Pro. The list below includes available Python packages that can

Note: Cannot modify the default Python environment. Clone and activate a new environment.

Note: We could get around this restriction by doing it outside of Pro, but we won’t.

How Pro gets packages

What kind of “packages” are these?

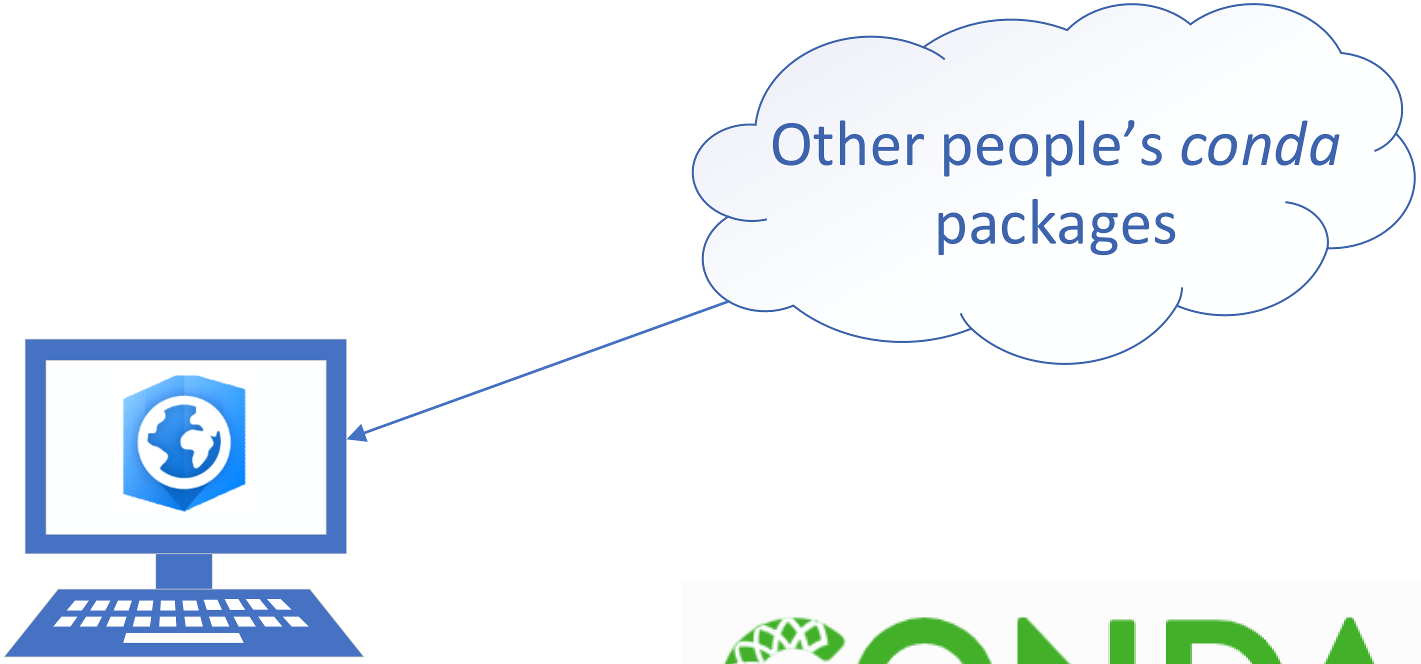


Other people’s packages

ArcGIS Pro

How Pro gets packages

They’re conda packages.



conda channel

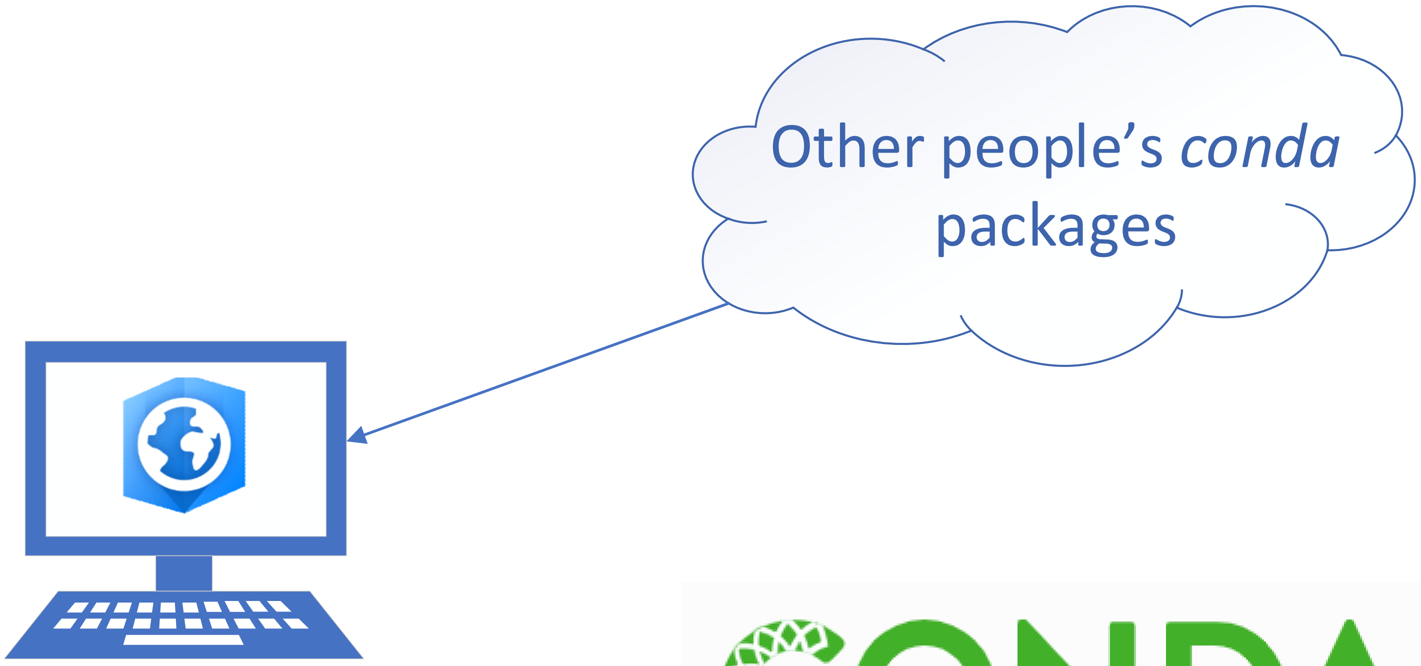
Other people’s *conda* packages

ArcGIS Pro
(conda)

How Pro gets packages

They’re conda packages.

Conda is open-source software for managing environments and packages.



conda channel

Other people’s *conda* packages

ArcGIS Pro
(conda)

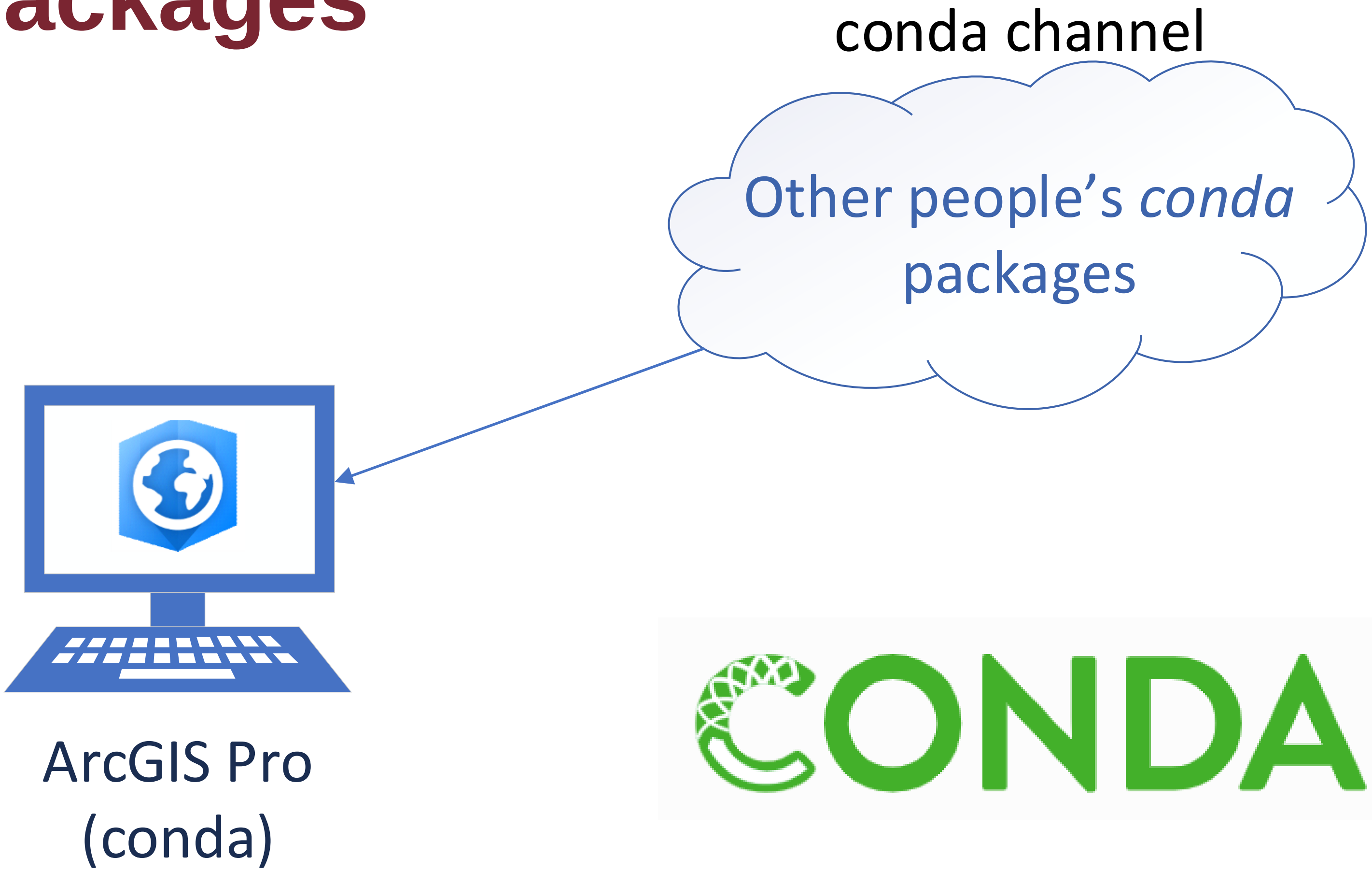
CONDA

How Pro gets packages

They’re conda packages.

Conda is open-source software for managing environments and packages.

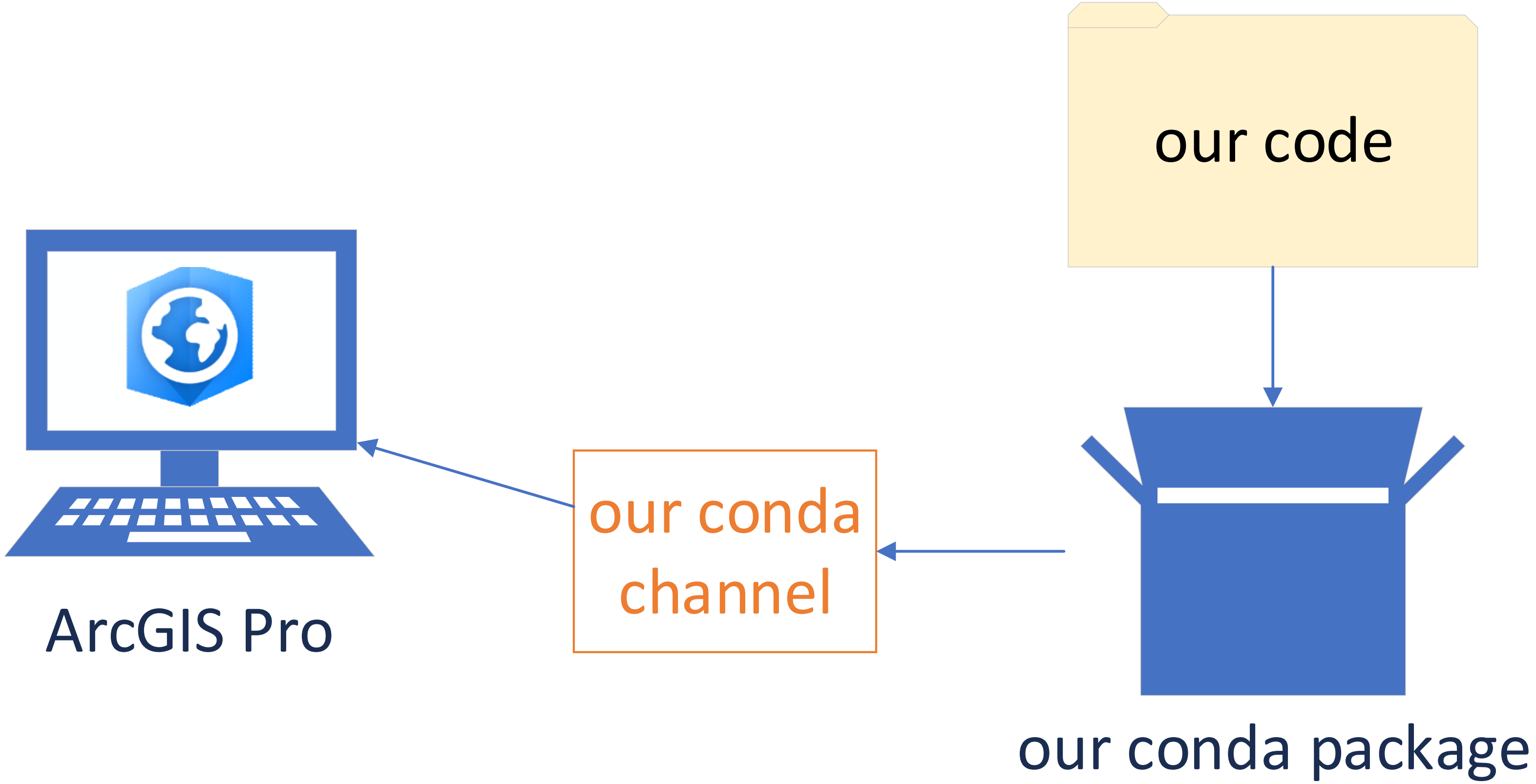
The “c” is lowercase.



CONDA

How Pro gets packages

We'll make our own conda package and channel.



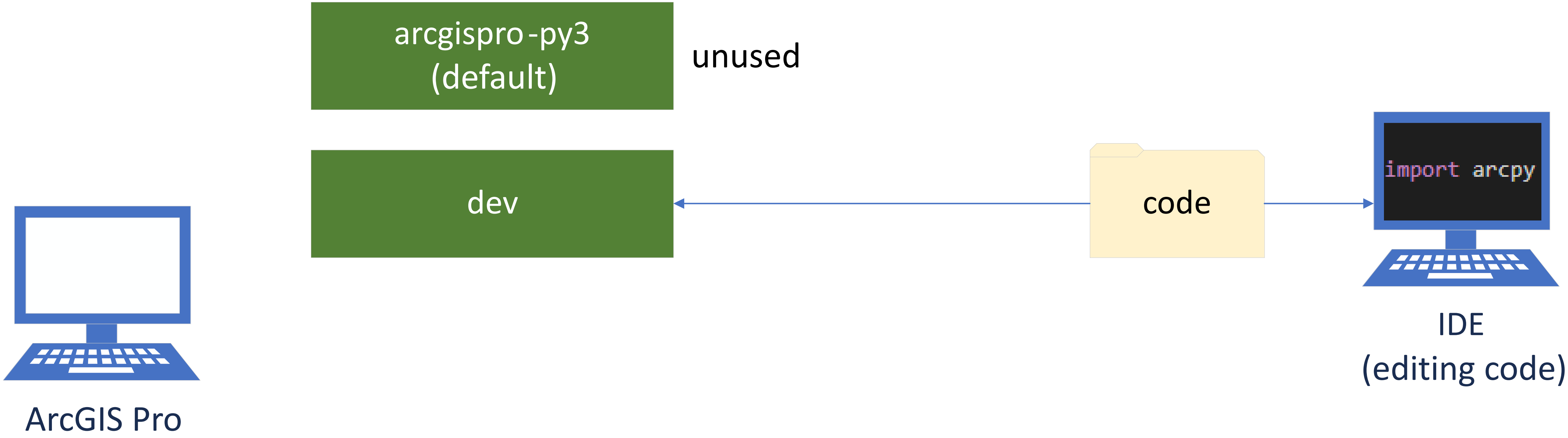
Our environments

arcgispro-py3
(default)

unused

ArcGIS Pro

Our environments



arcgispro-py3
(default)

unused

dev

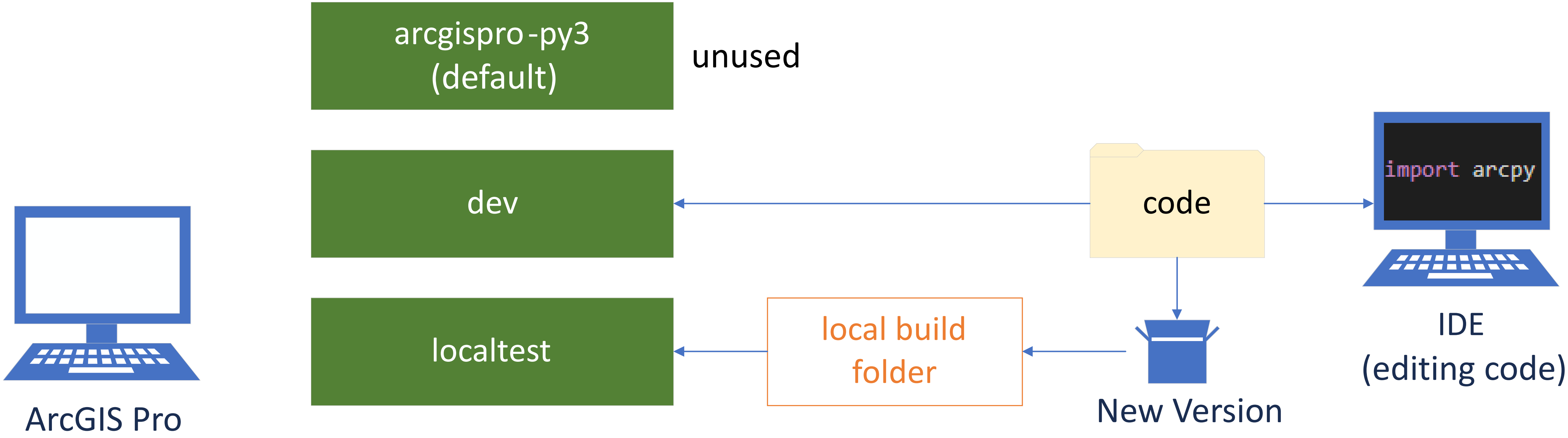
code

```
import arcpy
```

IDE
(editing code)

ArcGIS Pro

Our environments



arcgispro-py3 (default)

unused

dev

localtest

local build folder

code

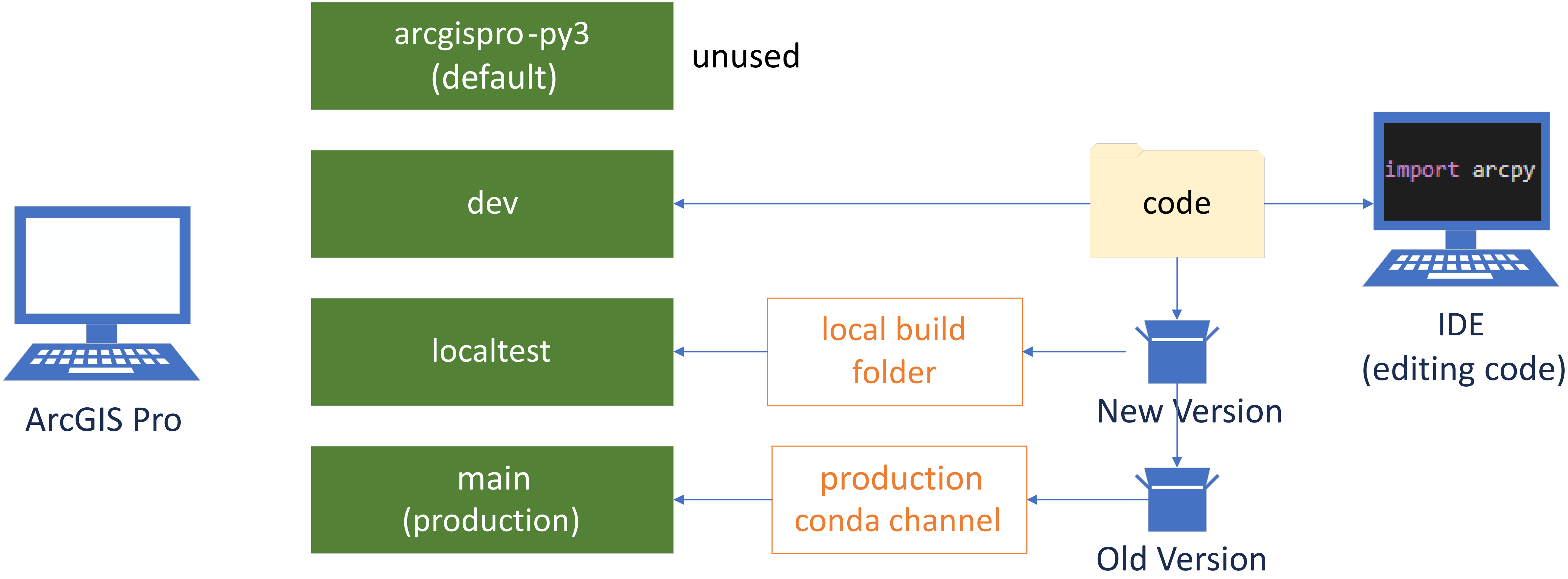
```
import arcpy
```

IDE (editing code)

New Version

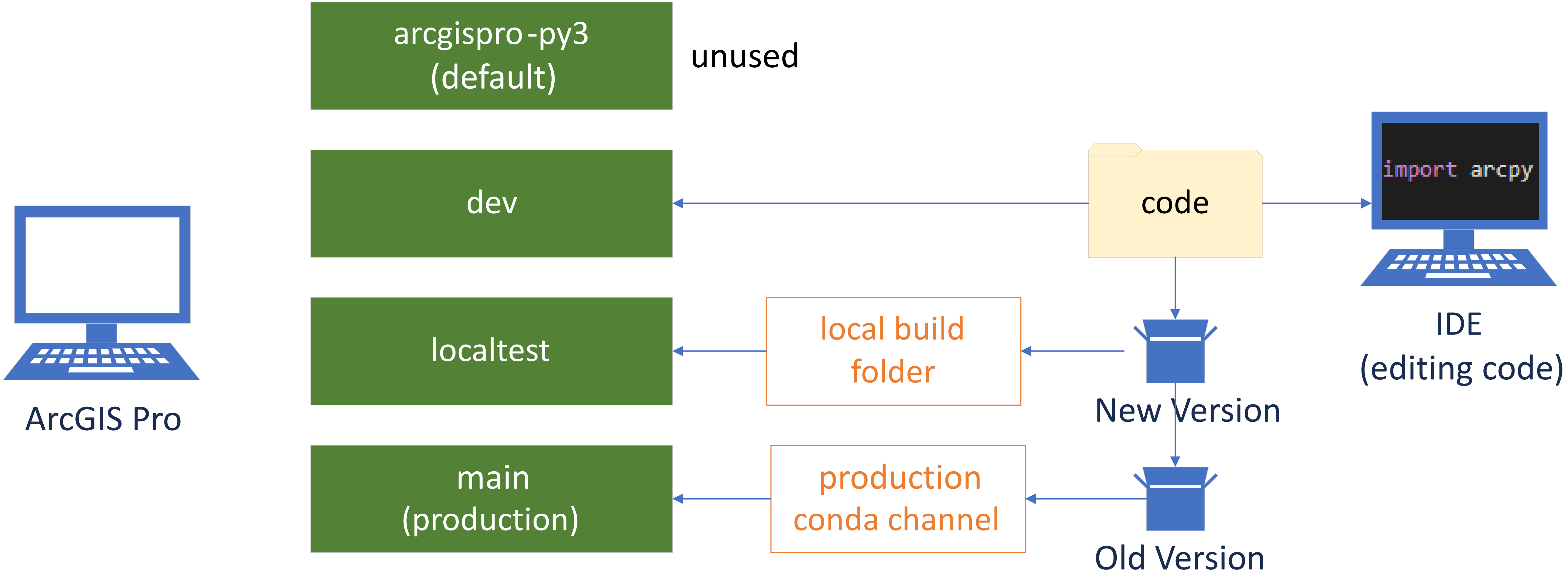
ArcGIS Pro

Our environments



```
import arcpy
```

Our environments



arcgispro-py3
(default)

unused

dev

localtest

main
(production)

ArcGIS Pro

code

import arcpy

IDE
(editing code)

local build
folder

production
conda channel

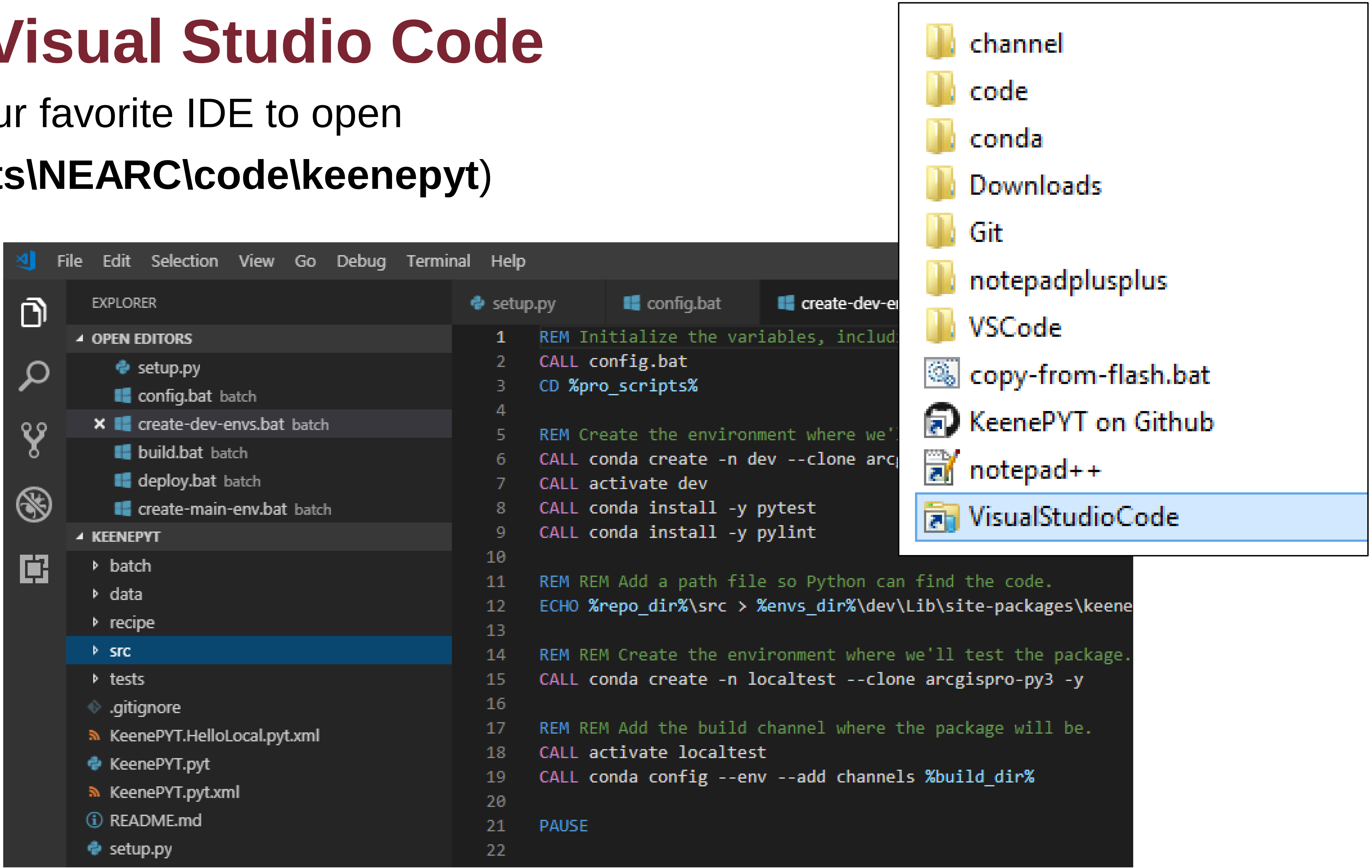
New Version

Old Version

+build environment (separate conda installation)

Open Visual Studio Code

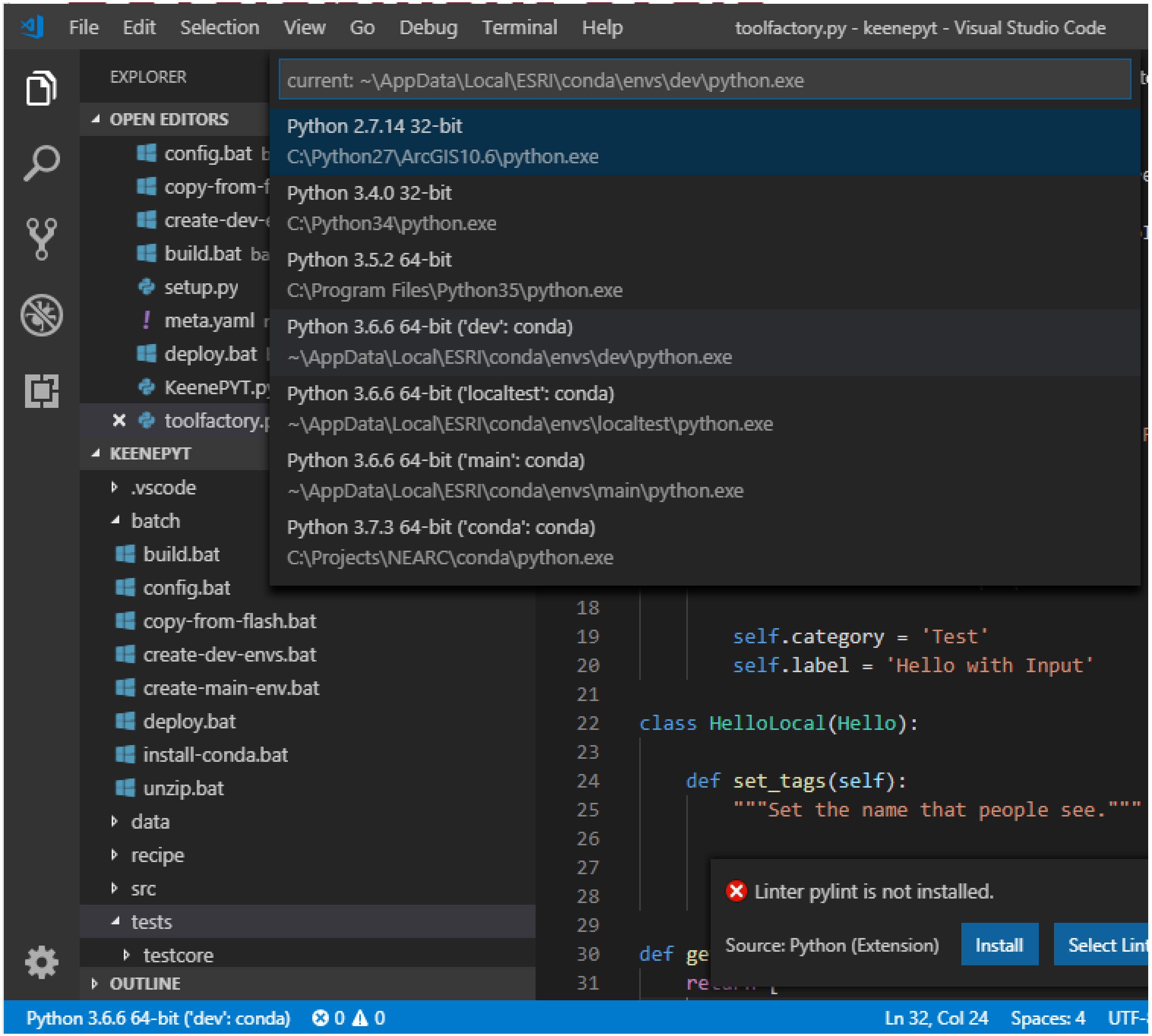
(or use your favorite IDE to open **C:\Projects\NEARC\code\keenepyt**)



```
REM Initialize the variables, includ
CALL config.bat
CD %pro_scripts%

REM Create the environment where we'
CALL conda create -n dev --clone arc
CALL activate dev
CALL conda install -y pytest
CALL conda install -y pylint

REM REM Add a path file so Python can find the code.
ECHO %repo_dir%\src > %envs_dir%\dev\Lib\site-packages\keene

REM REM Create the environment where we'll test the package.
CALL conda create -n localtest --clone arcgispro-py3 -y

REM REM Add the build channel where the package will be.
CALL activate localtest
CALL conda config --env --add channels %build_dir%

PAUSE

```

Set the Python interpreter.

Click on “Python” in the lower-left corner, then choose “dev.”



current: ~\AppData\Local\ESRI\conda\envs\dev\python.exe

Python 2.7.14 32-bit
C:\Python27\ArcGIS10.6\python.exe

Python 3.4.0 32-bit
C:\Python34\python.exe

Python 3.5.2 64-bit
C:\Program Files\Python35\python.exe

Python 3.6.6 64-bit ('dev': conda)
~\AppData\Local\ESRI\conda\envs\dev\python.exe

Python 3.6.6 64-bit ('localtest': conda)
~\AppData\Local\ESRI\conda\envs\localtest\python.exe

Python 3.6.6 64-bit ('main': conda)
~\AppData\Local\ESRI\conda\envs\main\python.exe

Python 3.7.3 64-bit ('conda': conda)
C:\Projects\NEARC\conda\python.exe

```
        self.category = 'Test'
        self.label = 'Hello with Input'

class HelloLocal(Hello):

    def set_tags(self):
        """Set the name that people see."""



def ge
    re
```

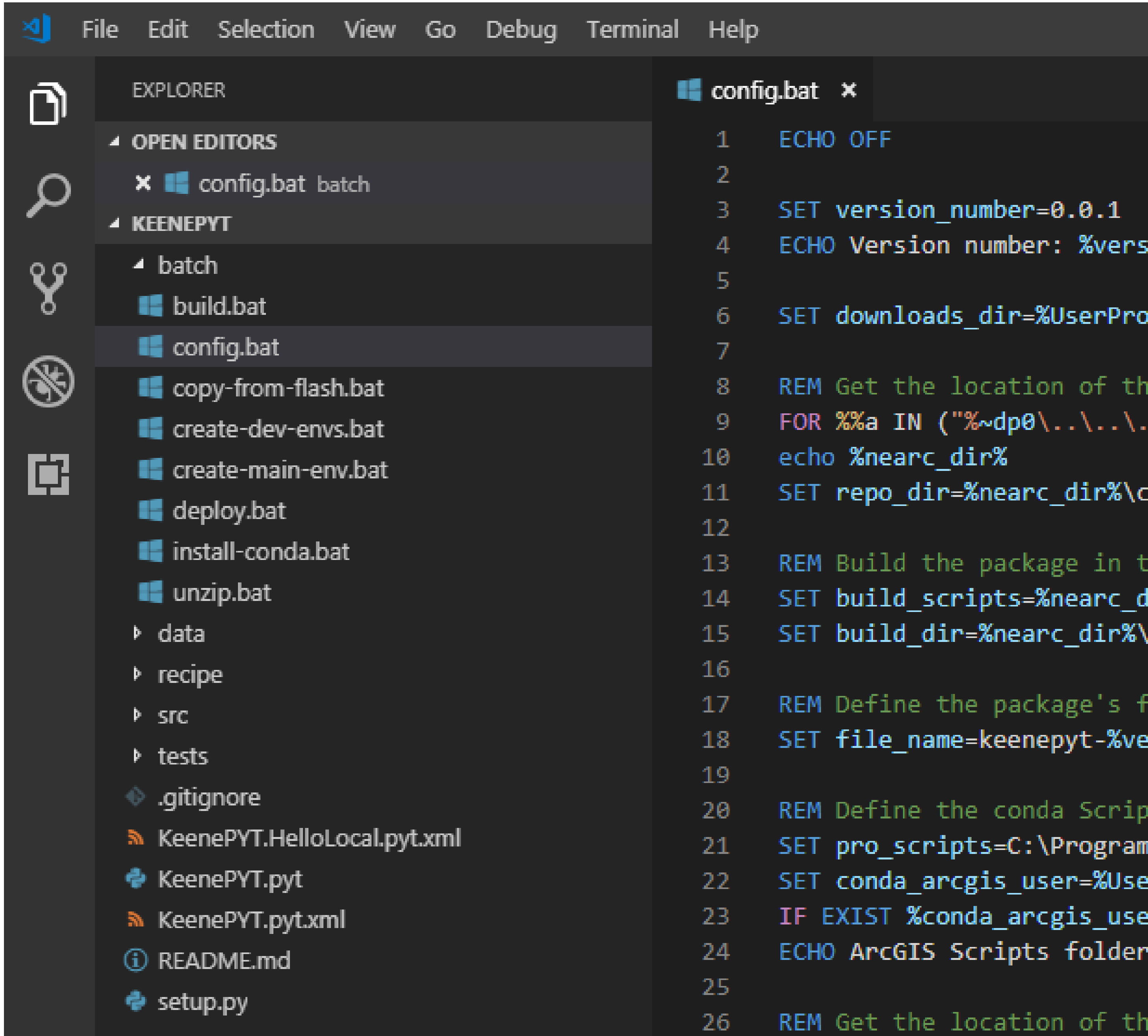
Linter pylint is not installed.

Source: Python (Extension)

Python 3.6.6 64-bit ('dev': conda)   ⊗ 0 ⚠ 0   Ln 32, Col 24   Spaces: 4   UTF-

View the batch files.

In your IDE, open **batch\config.bat**.

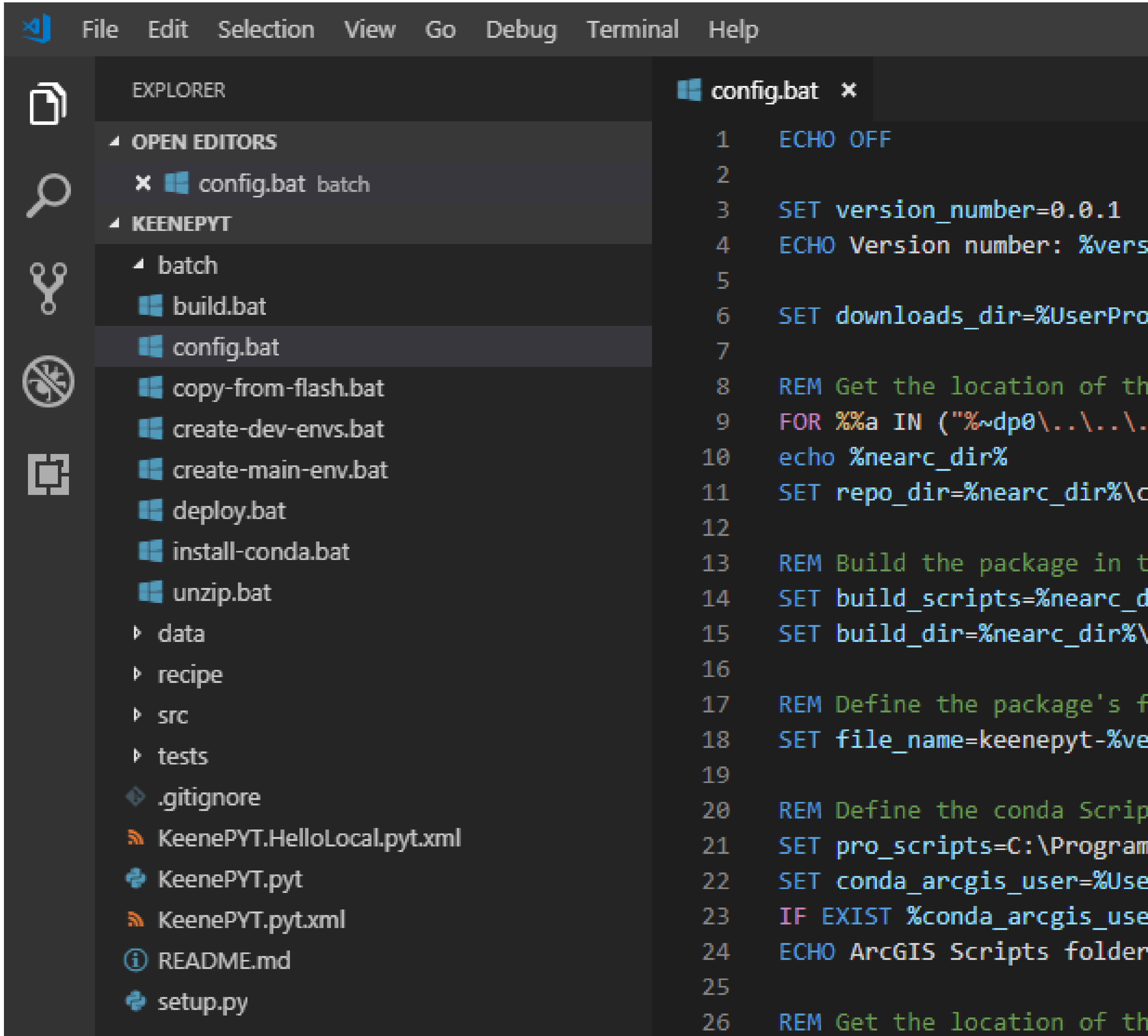


```
ECHO OFF

SET version_number=0.0.1
ECHO Version number: %vers

SET downloads_dir=%UserPro

REM Get the location of th
FOR %%a IN ("%~dp0\..\..\.
echo %nearc_dir%
SET repo_dir=%nearc_dir%\c

REM Build the package in t
SET build_scripts=%nearc_d
SET build_dir=%nearc_dir%\

REM Define the package's f
SET file_name=keenepyt-%ve

REM Define the conda Scrip
SET pro_scripts=C:\Program
SET conda_arcgis_user=%Use
IF EXIST %conda_arcgis_use
ECHO ArcGIS Scripts folder

REM Get the location of th
```

View the batch files.

In your IDE, open **batch\config.bat**.

This batch file sets a bunch of variables that other batch files will use, including the version number of our package.



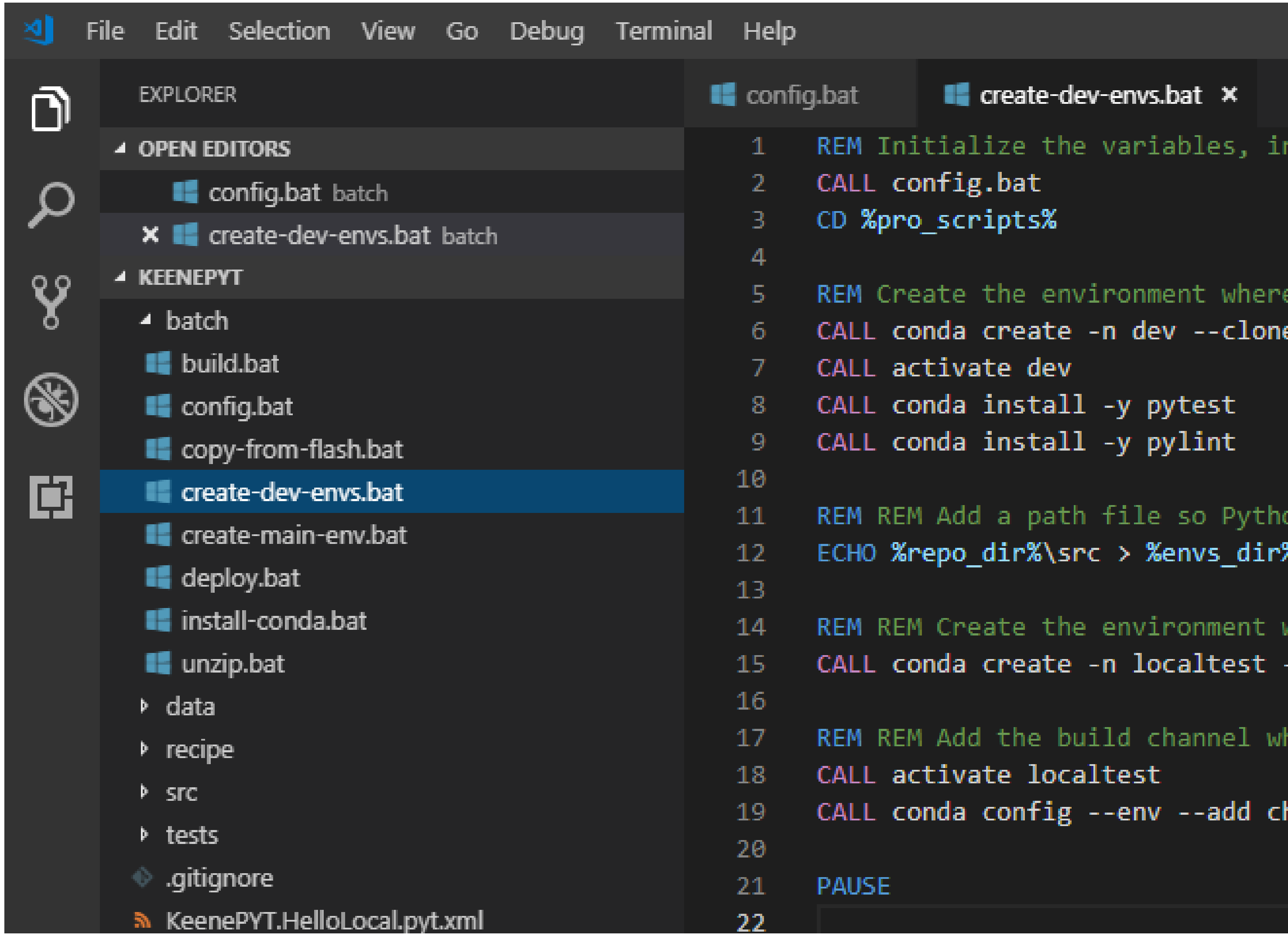
```
ECHO OFF

SET version_number=0.0.1
ECHO Version number: %vers

SET downloads_dir=%UserPro

REM Get the location of th
FOR %%a IN ("%~dp0\..\..\.
echo %nearc_dir%
SET repo_dir=%nearc_dir%\c

REM Build the package in t
SET build_scripts=%nearc_d
SET build_dir=%nearc_dir%\

REM Define the package's f
SET file_name=keenepyt-%ve

REM Define the conda Scrip
SET pro_scripts=C:\Program
SET conda_arcgis_user=%Use
IF EXIST %conda_arcgis_use
ECHO ArcGIS Scripts folder

REM Get the location of th
```

Terracon

View the batch files.

Open **create-dev-envs.bat**.

You ran this one earlier.

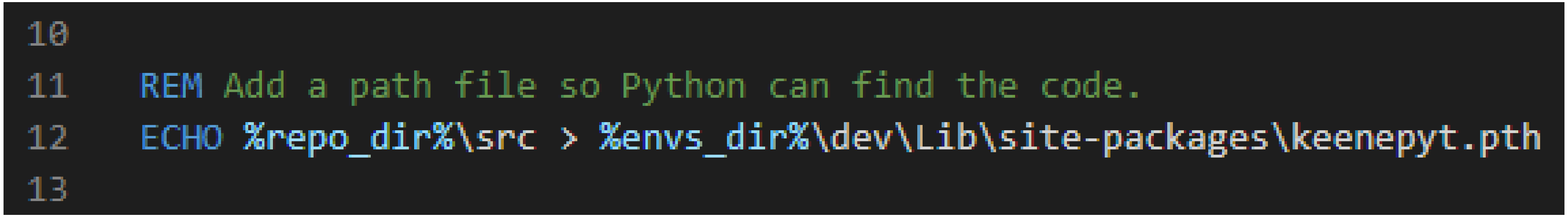


```
REM Initialize the variables, i
CALL config.bat
CD %pro_scripts%

REM Create the environment where
CALL conda create -n dev --clone
CALL activate dev
CALL conda install -y pytest
CALL conda install -y pylint

REM REM Add a path file so Pytho
ECHO %repo_dir%\src > %envs_dir

REM REM Create the environment w
CALL conda create -n localtest -

REM REM Add the build channel wh
CALL activate localtest
CALL conda config --env --add ch

PAUSE

```

View the batch files.

Open **create-dev-envs.bat**.

Note this line:

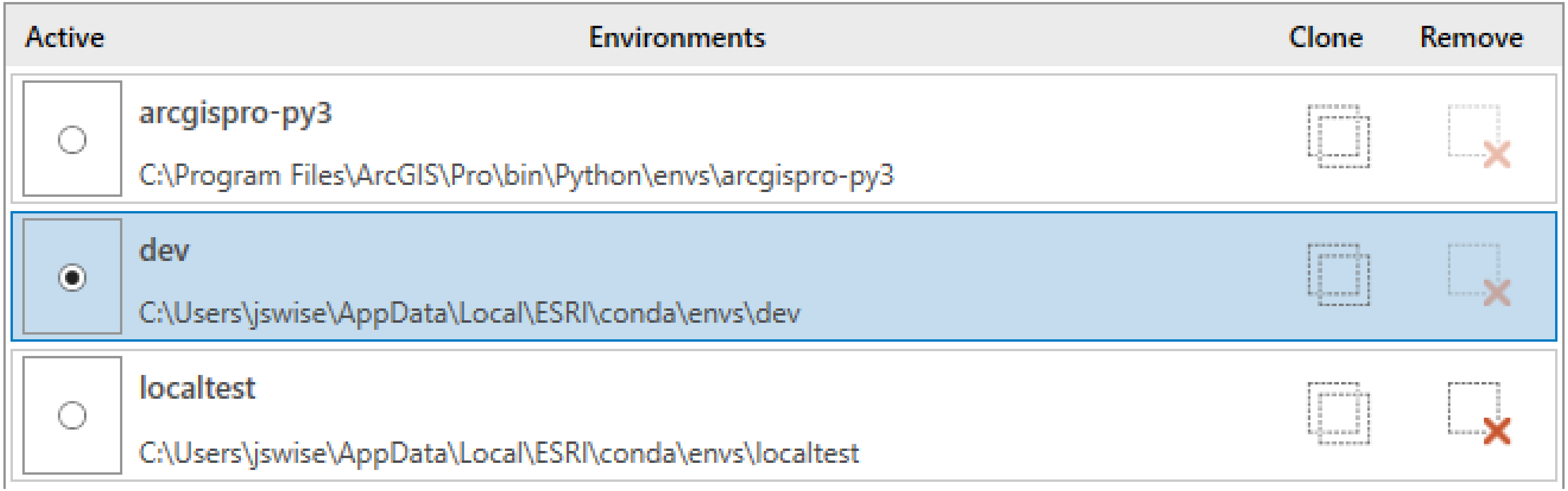


```
REM Add a path file so Python can find the code.
ECHO %repo_dir%\src > %envs_dir%\dev\Lib\site-packages\keenepyt.pth
```

It creates a path file (.pth) that contains the path to your code.

Activate the dev environment.

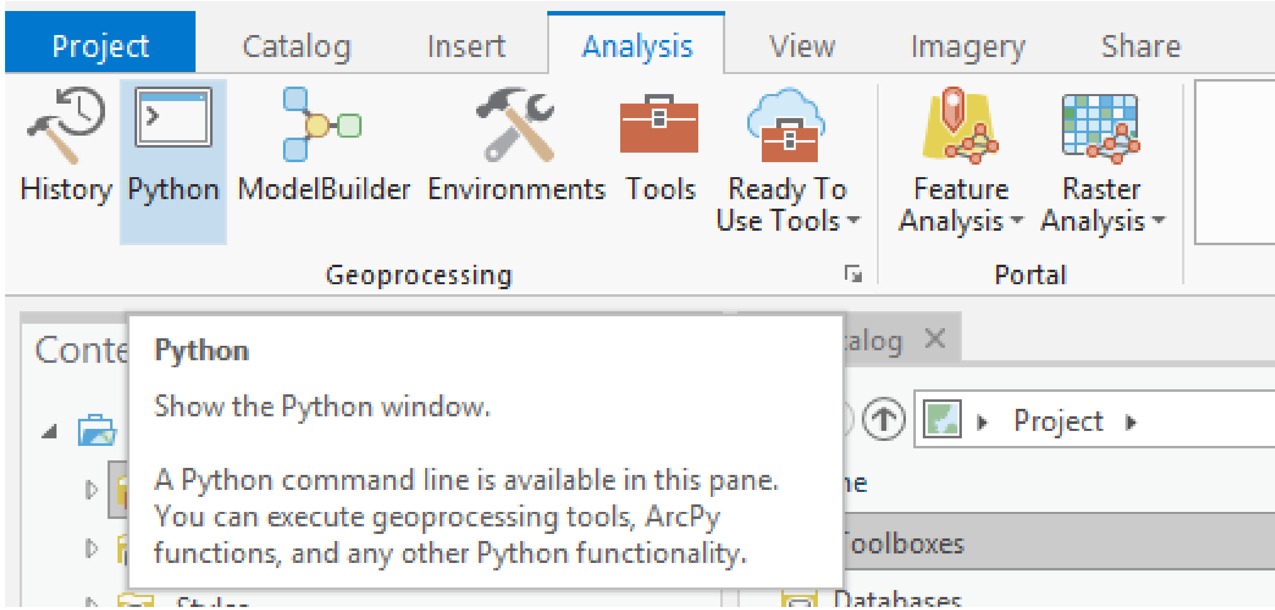
Open ArcGIS Pro, and use the Python Package Manager to activate dev.



Active	Environments	Clone	Remove
○	arcgispro-py3 C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3		
●	dev C:\Users\jswise\AppData\Local\ESRI\conda\envs\dev		
○	localtest C:\Users\jswise\AppData\Local\ESRI\conda\envs\localtest		

Try importing keenepyt.

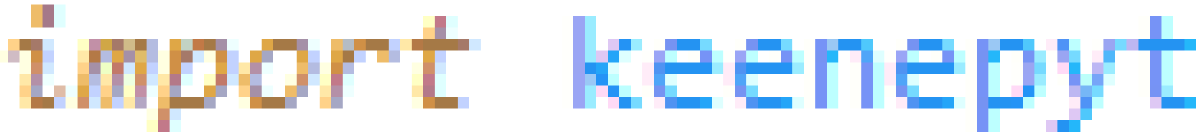
This should work in the dev environment.



Python

Show the Python window.

A Python command line is available in this pane. You can execute geoprocessing tools, ArcPy functions, and any other Python functionality.

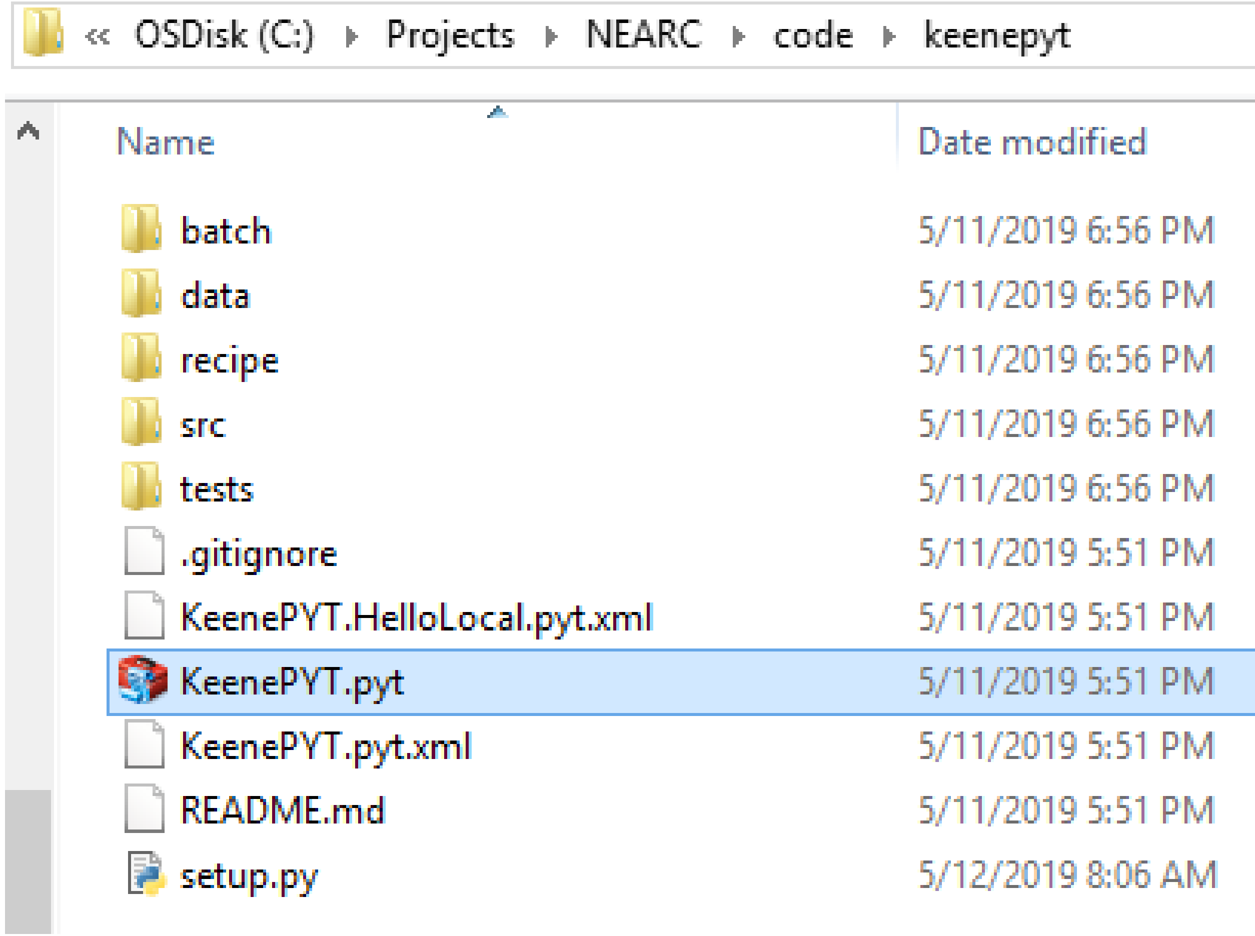


```
import keenepyt
```

Pretend to distribute the toolbox.

Find KeenePYT.pyt, and copy it to some other place.

This simulates giving your toolbox to somebody.

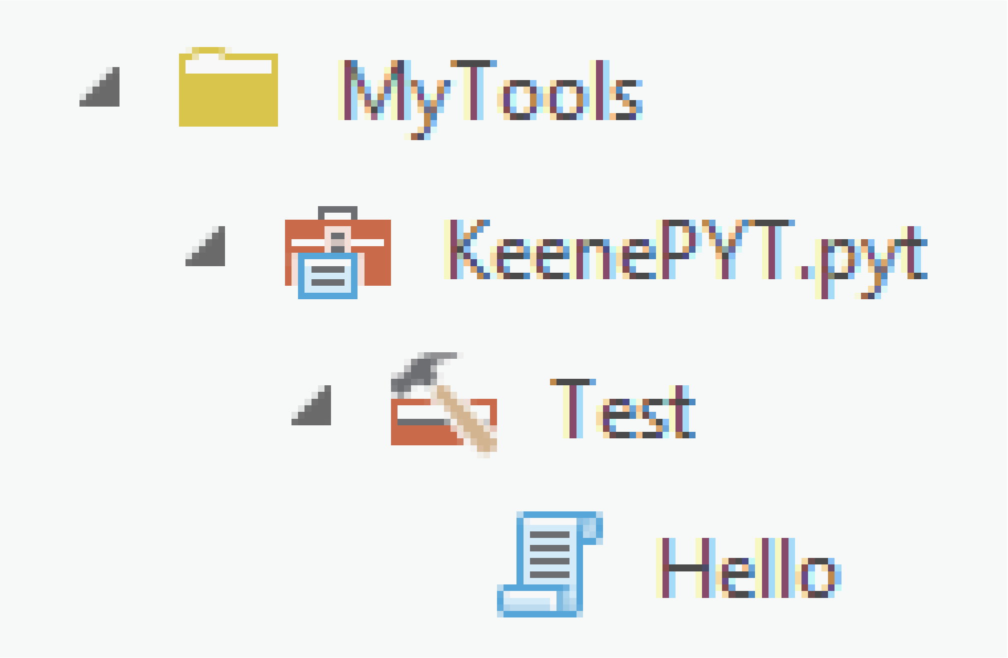


« OSDisk (C:) › Projects › NEARC › code › keenepyt

Name	Date modified
batch	5/11/2019 6:56 PM
data	5/11/2019 6:56 PM
recipe	5/11/2019 6:56 PM
src	5/11/2019 6:56 PM
tests	5/11/2019 6:56 PM
.gitignore	5/11/2019 5:51 PM
KeenePYT.HelloLocal.pyt.xml	5/11/2019 5:51 PM
KeenePYT.pyt	5/11/2019 5:51 PM
KeenePYT.pyt.xml	5/11/2019 5:51 PM
README.md	5/11/2019 5:51 PM
setup.py	5/12/2019 8:06 AM

Open the toolbox.

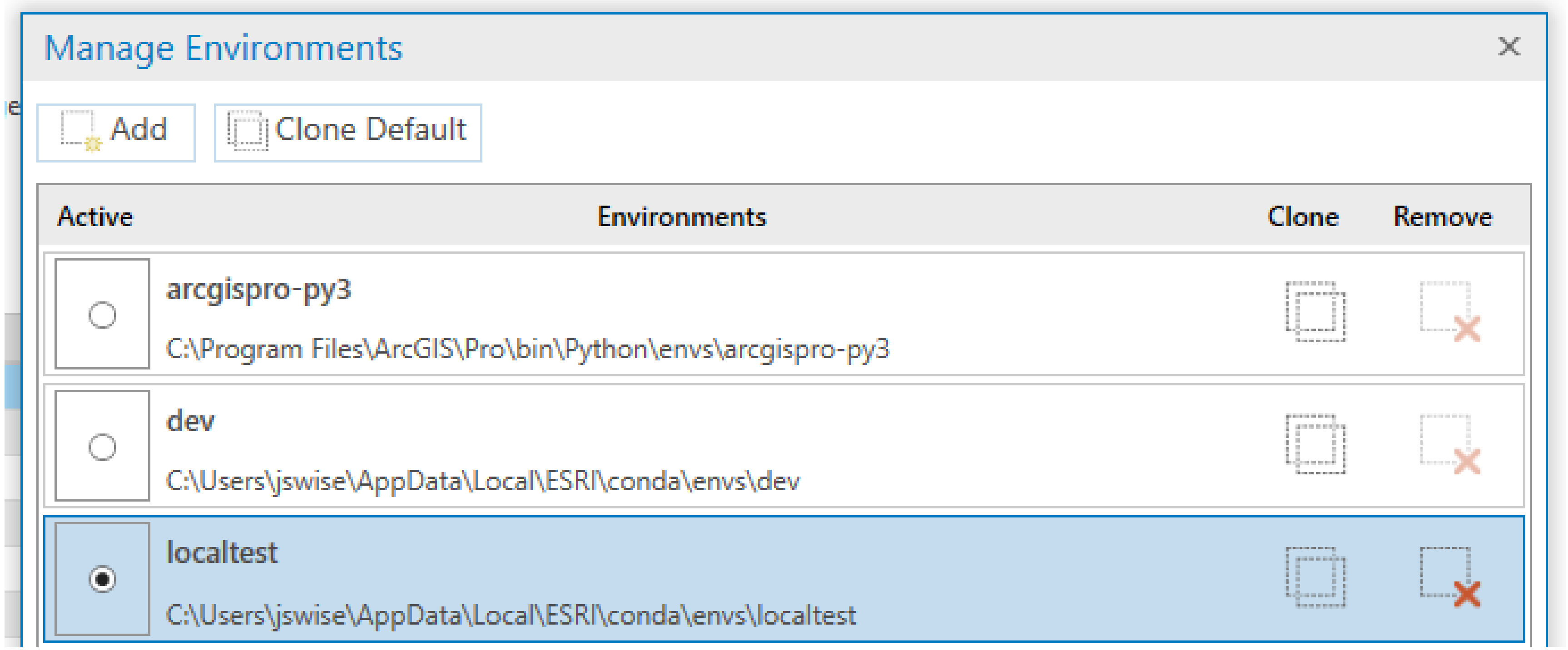
You should be able to open the toolbox in Pro.



Change environments.

Activate localtest in Pro.

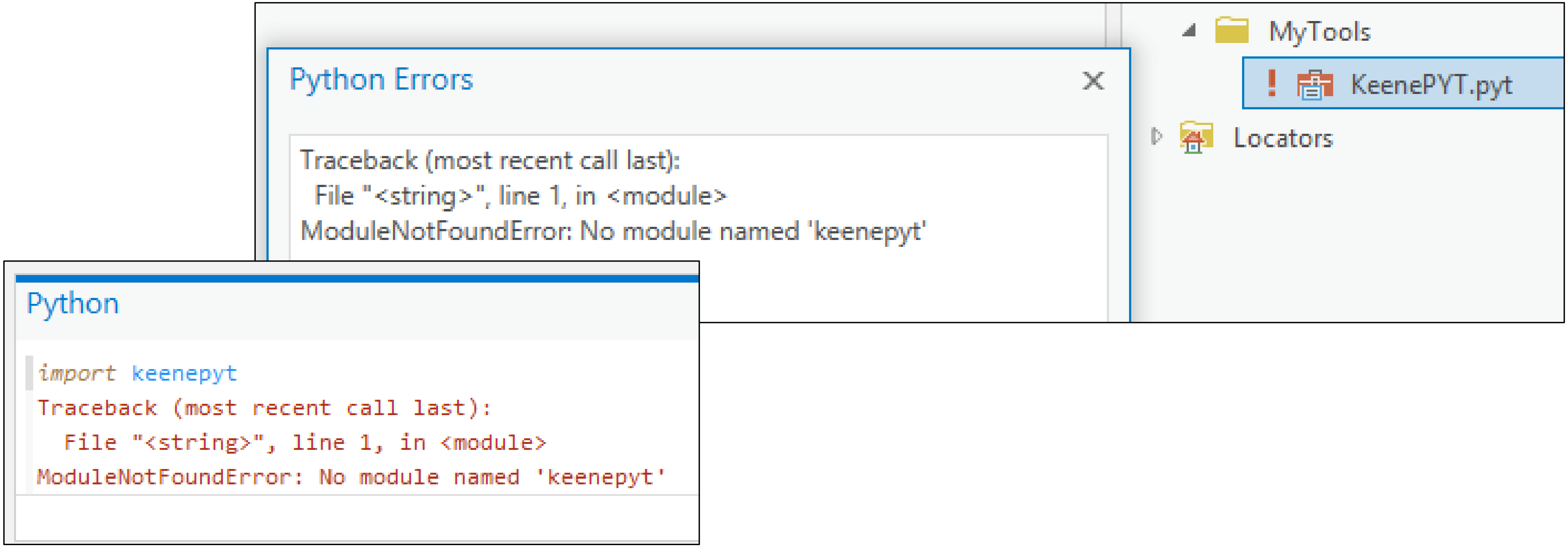
Restart Pro.



Active	Environments	Clone	Remove
○	arcgispro-py3 C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3		
○	dev C:\Users\jswise\AppData\Local\ESRI\conda\envs\dev		
●	localtest C:\Users\jswise\AppData\Local\ESRI\conda\envs\localtest		

Your toolbox is now broken.

The localtest environment doesn’t know about your code.



Python Errors

```
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ModuleNotFoundError: No module named 'keenepyt'
```

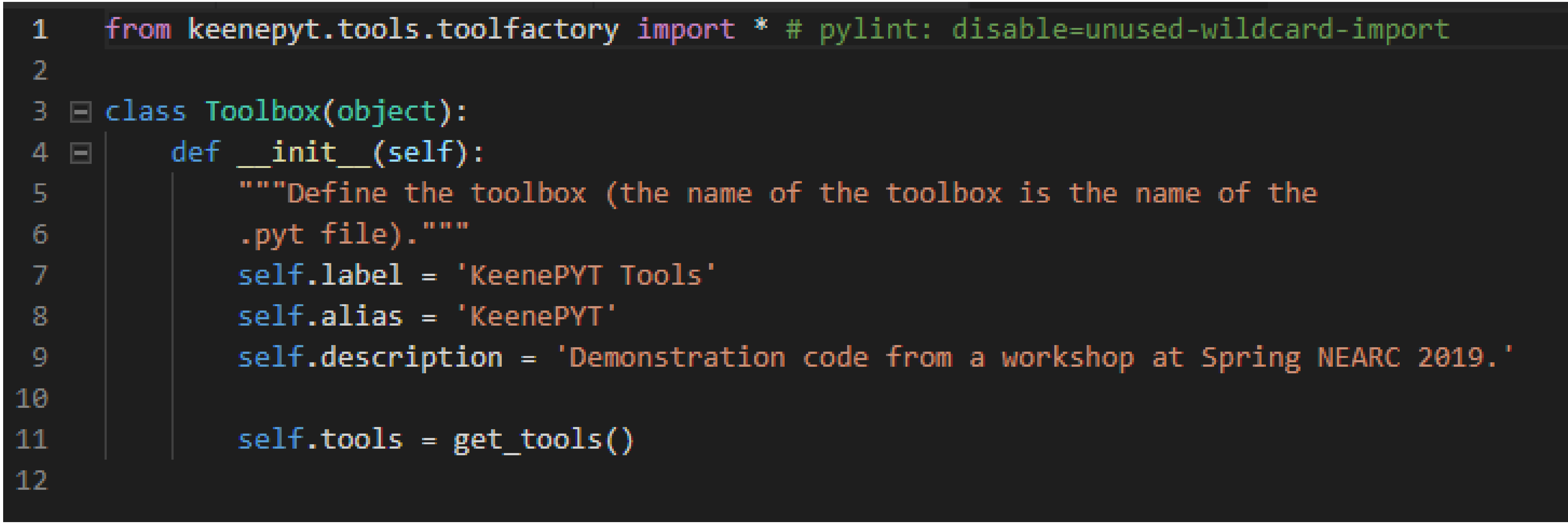
Python

```
import keenepyt
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ModuleNotFoundError: No module named 'keenepyt'
```

Look at the toolbox code.

It’s short.

It imports *keenepyt.tools.toolfactory* and runs the *get_tools* method.

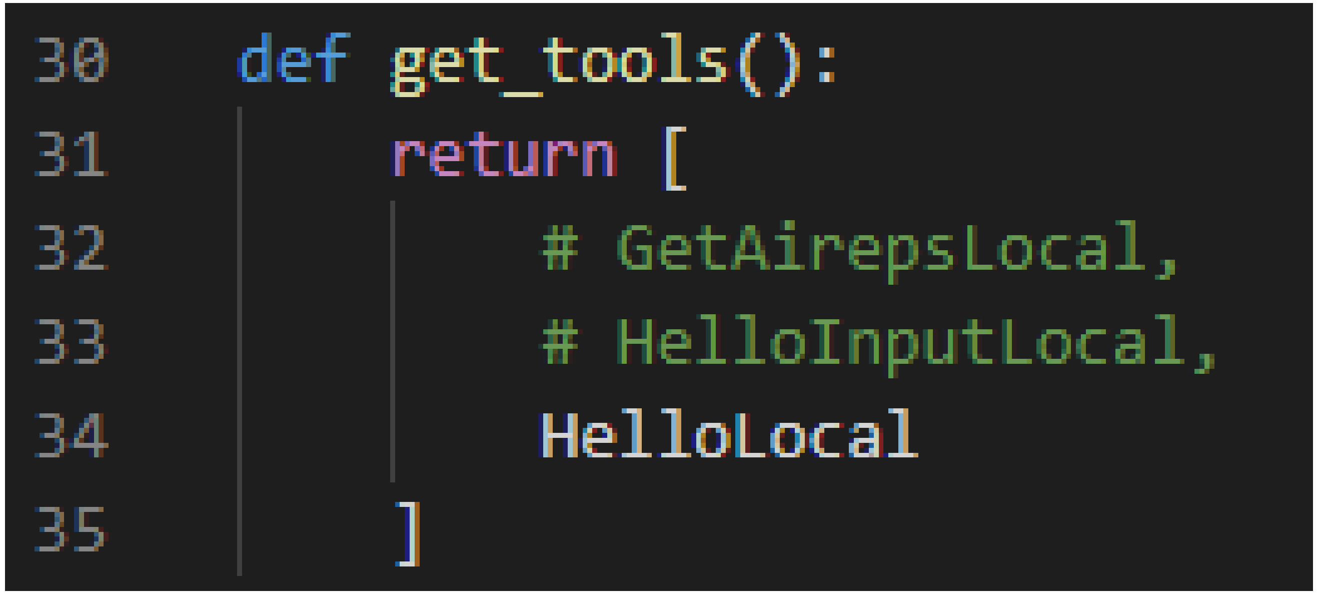


```python
from keenepyt.tools.toolfactory import * # pylint: disable=unused-wildcard-import

class Toolbox(object):
    def __init__(self):
        """Define the toolbox (the name of the toolbox is the name of the
        .pyt file)."""
        self.label = 'KeenePYT Tools'
        self.alias = 'KeenePYT'
        self.description = 'Demonstration code from a workshop at Spring NEARC 2019.'

        self.tools = get_tools()

```

Look at the toolbox code.

The *get_tools* method in *keenepyt.tools.toolfactory* provides a list of tool classes.

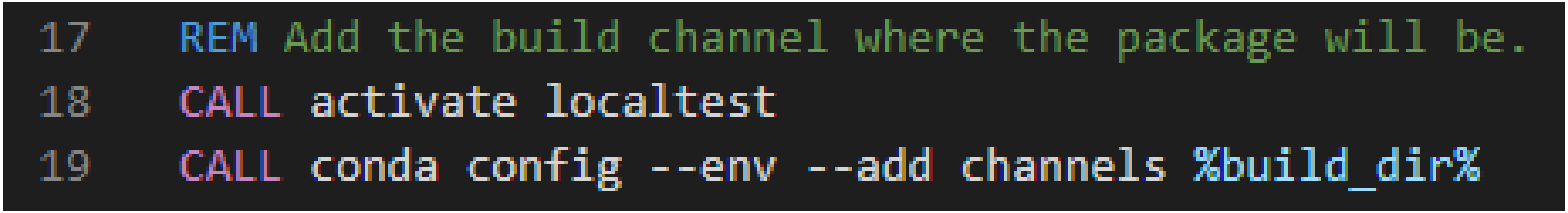
Some are commented out so we can have fun adding them later.



```python
def get_tools():
    return [
        # GetAirepsLocal,
        # HelloInputLocal,
        HelloLocal
    ]
```

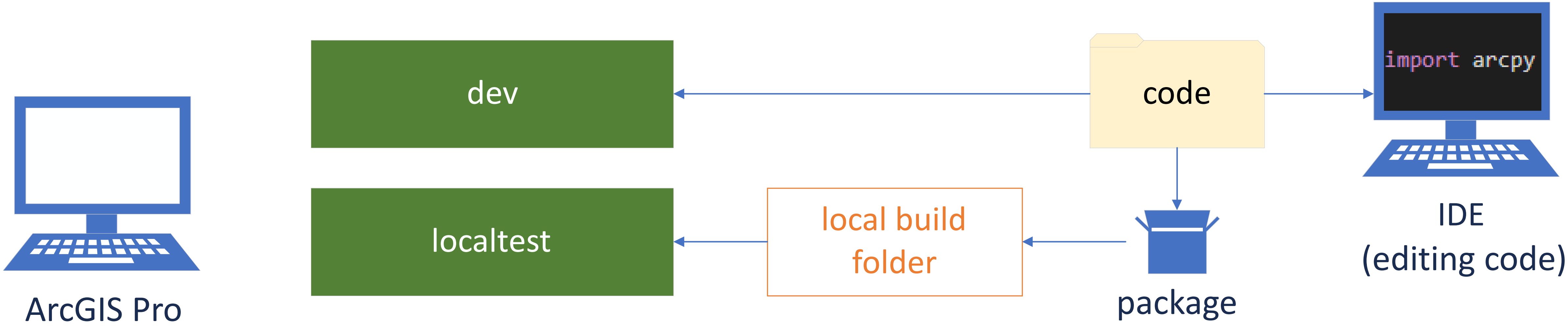
Why is it broken?

In **create-dev-envs.bat**, we told the localtest environment where to find a conda channel instead of the code. There’s no package in the channel yet.



```
REM Add the build channel where the package will be.
CALL activate localtest
CALL conda config --env --add channels %build_dir%
```

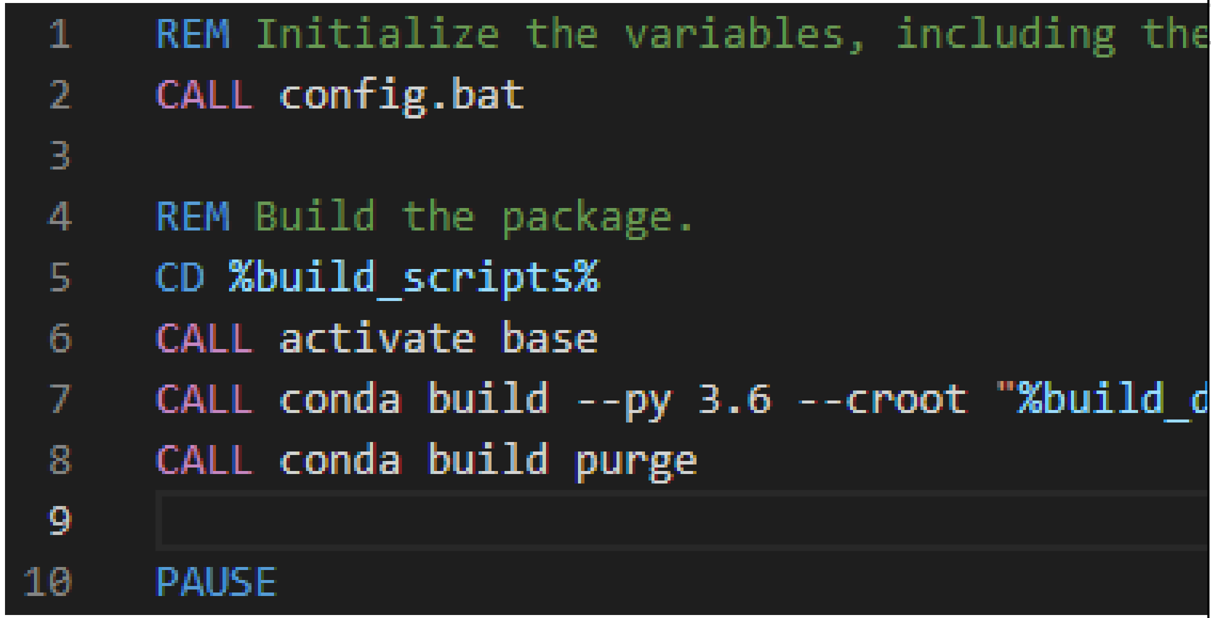
Now we need to build the package.



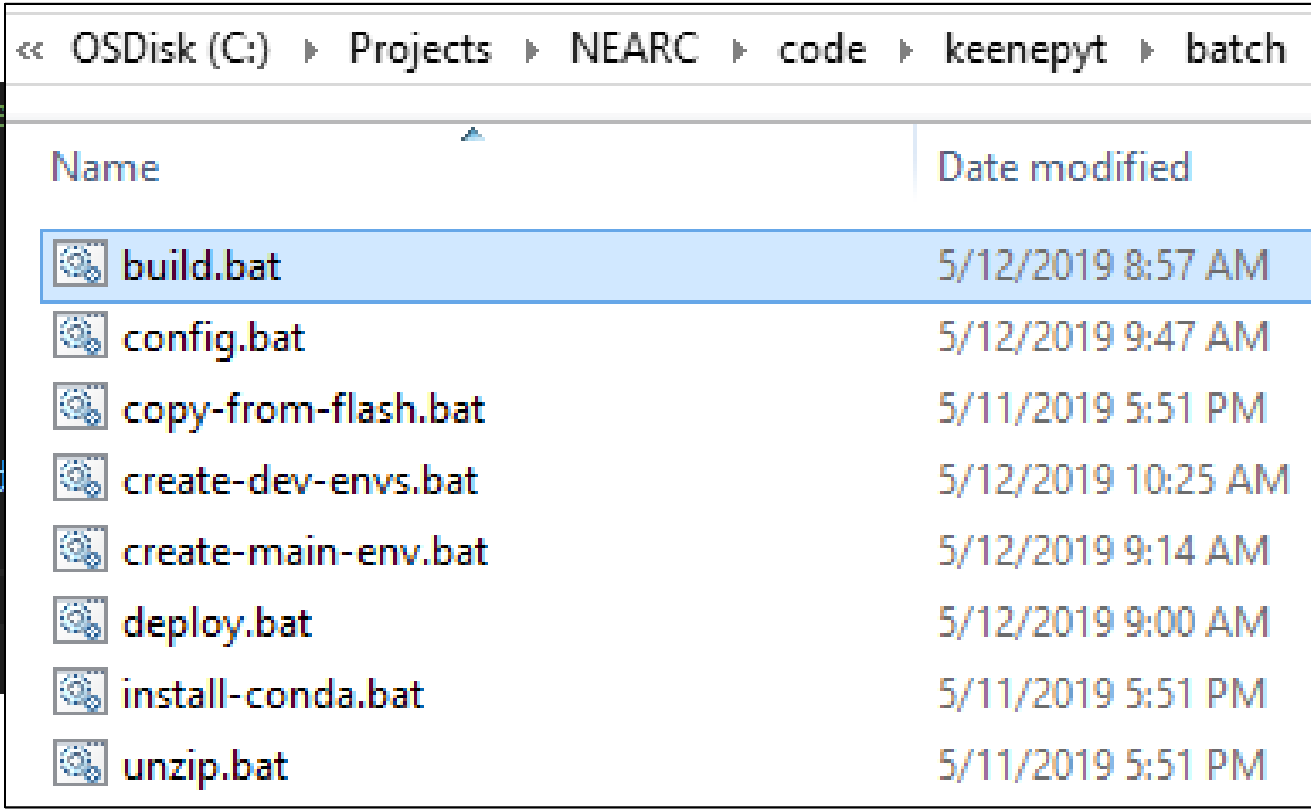
Run build.bat.

This creates a package in

C:\Projects\NEARC\conda\conda-bld\win-64.



```
REM Initialize the variables, including the
CALL config.bat

REM Build the package.
CD %build_scripts%
CALL activate base
CALL conda build --py 3.6 --croot "%build_d
CALL conda build purge

PAUSE
```



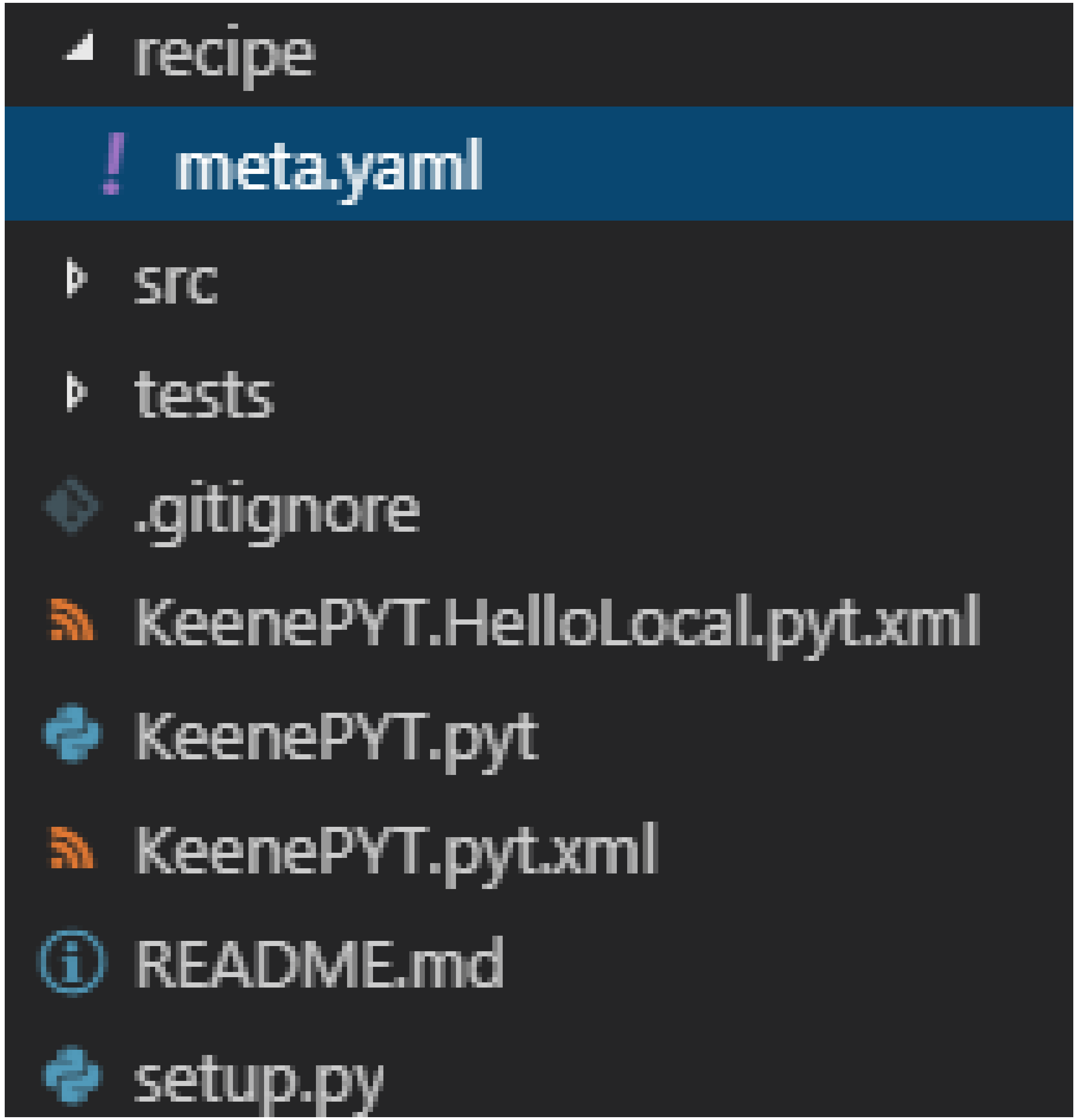
« OSDisk (C:) ▸ Projects ▸ NEARC ▸ code ▸ keenepyt ▸ batch

Name	Date modified
build.bat	5/12/2019 8:57 AM
config.bat	5/12/2019 9:47 AM
copy-from-flash.bat	5/11/2019 5:51 PM
create-dev-envs.bat	5/12/2019 10:25 AM
create-main-env.bat	5/12/2019 9:14 AM
deploy.bat	5/12/2019 9:00 AM
install-conda.bat	5/11/2019 5:51 PM
unzip.bat	5/11/2019 5:51 PM

Run build.bat.

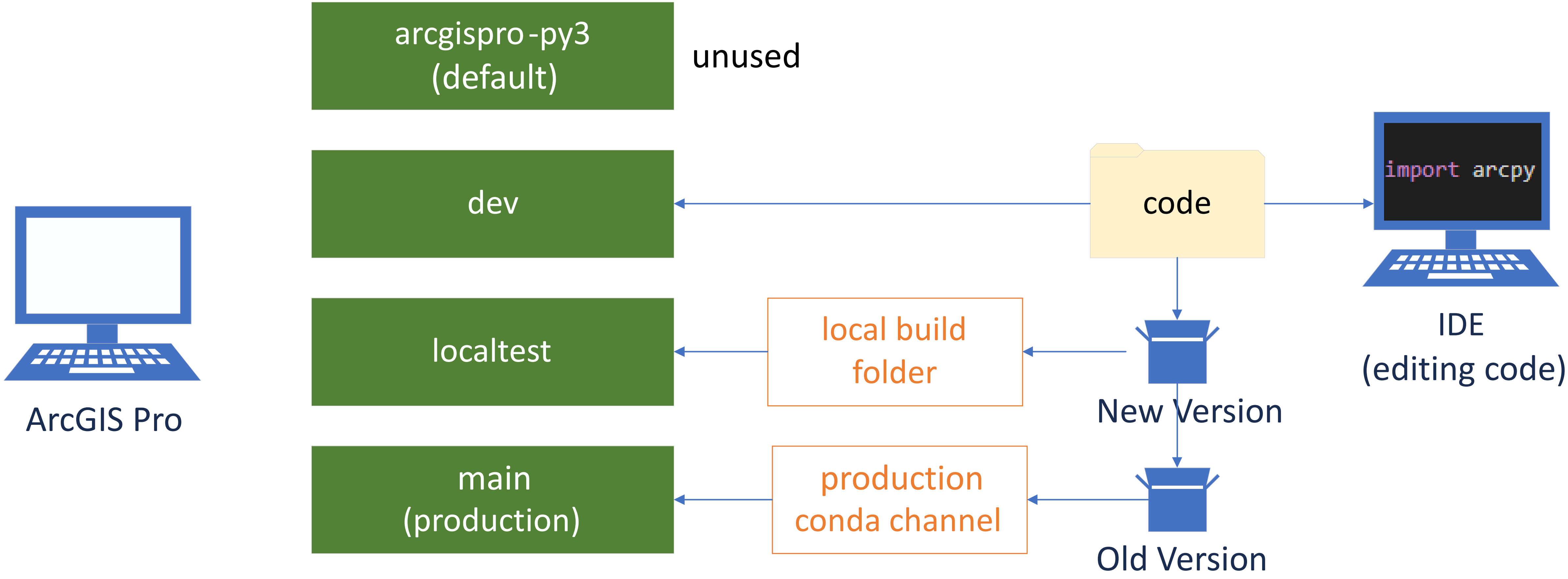
The build process depends on **recipe\meta.yaml** and **setup.py**.

We’re using a separate conda installation, because we can’t do it in ArcGIS Pro’s conda installation.



```
recipe
  meta.yaml
src
tests
.gitignore
KeenePYT.HelloLocal.pyt.xml
KeenePYT.pyt
KeenePYT.pyt.xml
README.md
setup.py
```

Our environments (review)



arcgispro-py3 (default)

unused

dev

localtest

main (production)

ArcGIS Pro

local build folder

production conda channel

code

New Version

Old Version

```
import arcpy
```

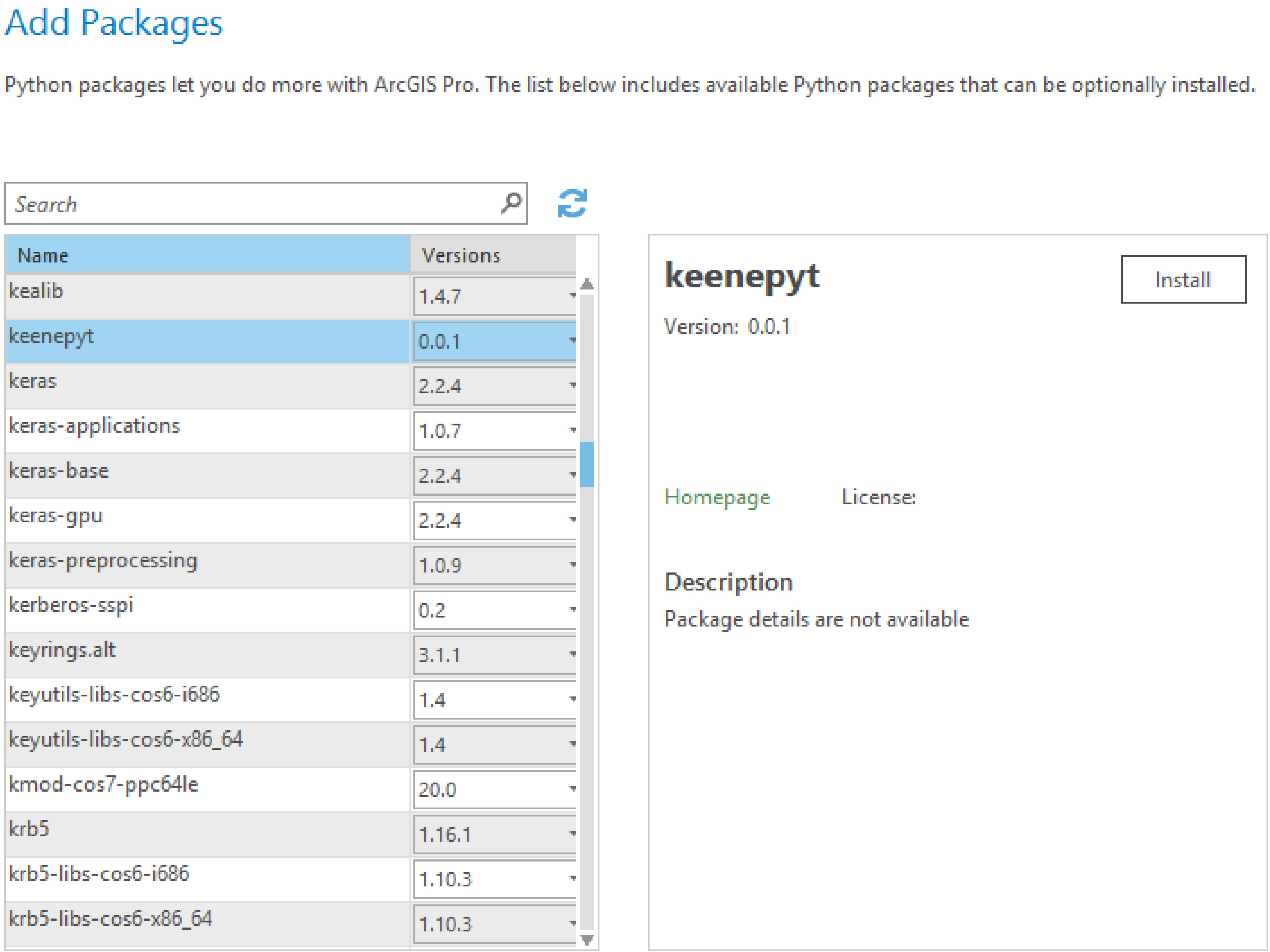
IDE (editing code)

+build environment (separate conda installation)

Install the package.

You should now be able to install keenepyt in localtest.

This will fix the toolbox.



Add Packages

Python packages let you do more with ArcGIS Pro. The list below includes available Python packages that can be optionally installed.

Name	Versions
kealib	1.4.7
keenepyt	0.0.1
keras	2.2.4
keras-applications	1.0.7
keras-base	2.2.4
keras-gpu	2.2.4
keras-preprocessing	1.0.9
kerberos-sspi	0.2
keyrings.alt	3.1.1
keyutils-libs-cos6-i686	1.4
keyutils-libs-cos6-x86_64	1.4
kmod-cos7-ppc64le	20.0
krb5	1.16.1
krb5-libs-cos6-i686	1.10.3
krb5-libs-cos6-x86_64	1.10.3

keenepyt

Version: 0.0.1

Homepage License:

Description

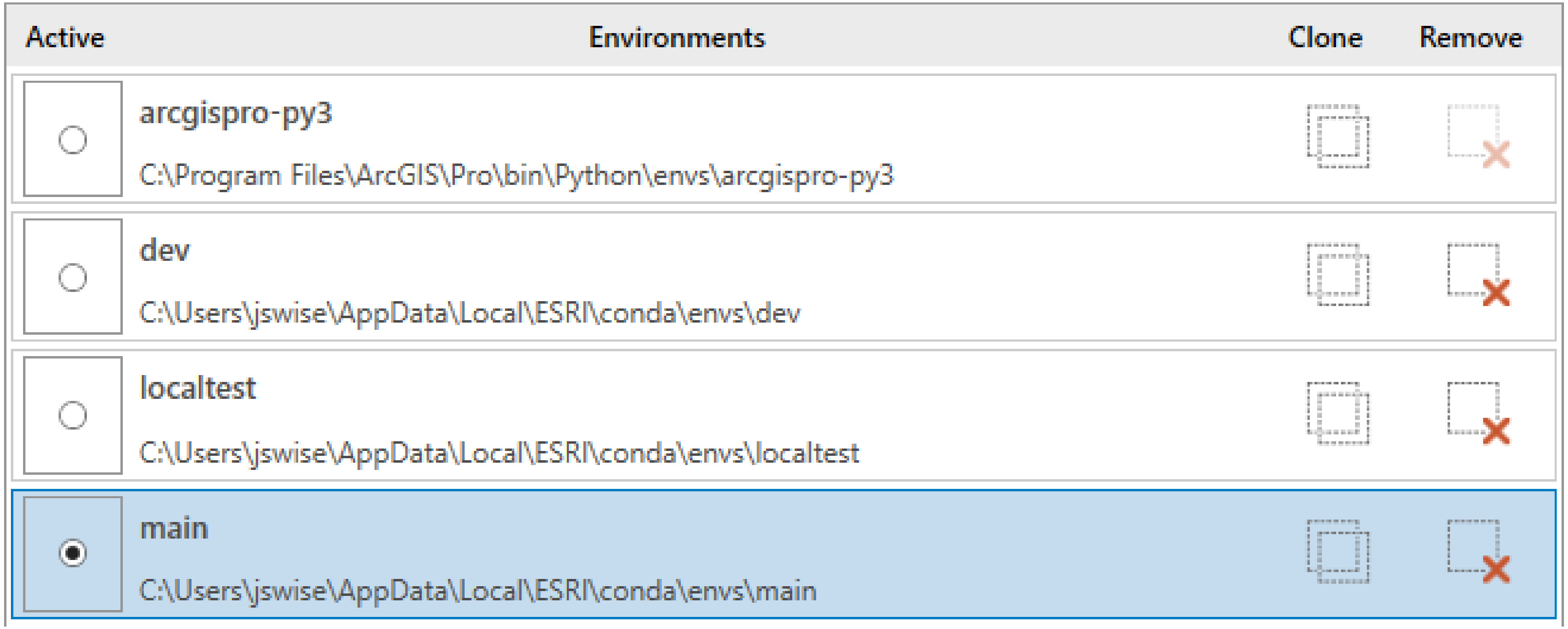
Package details are not available

Terracon

Create the production environment.

In **C:\Projects\NEARC\code\keenepyt\batch**, run **create-main-env.bat.**

This will clone your default Python environment another new environment, *main*.



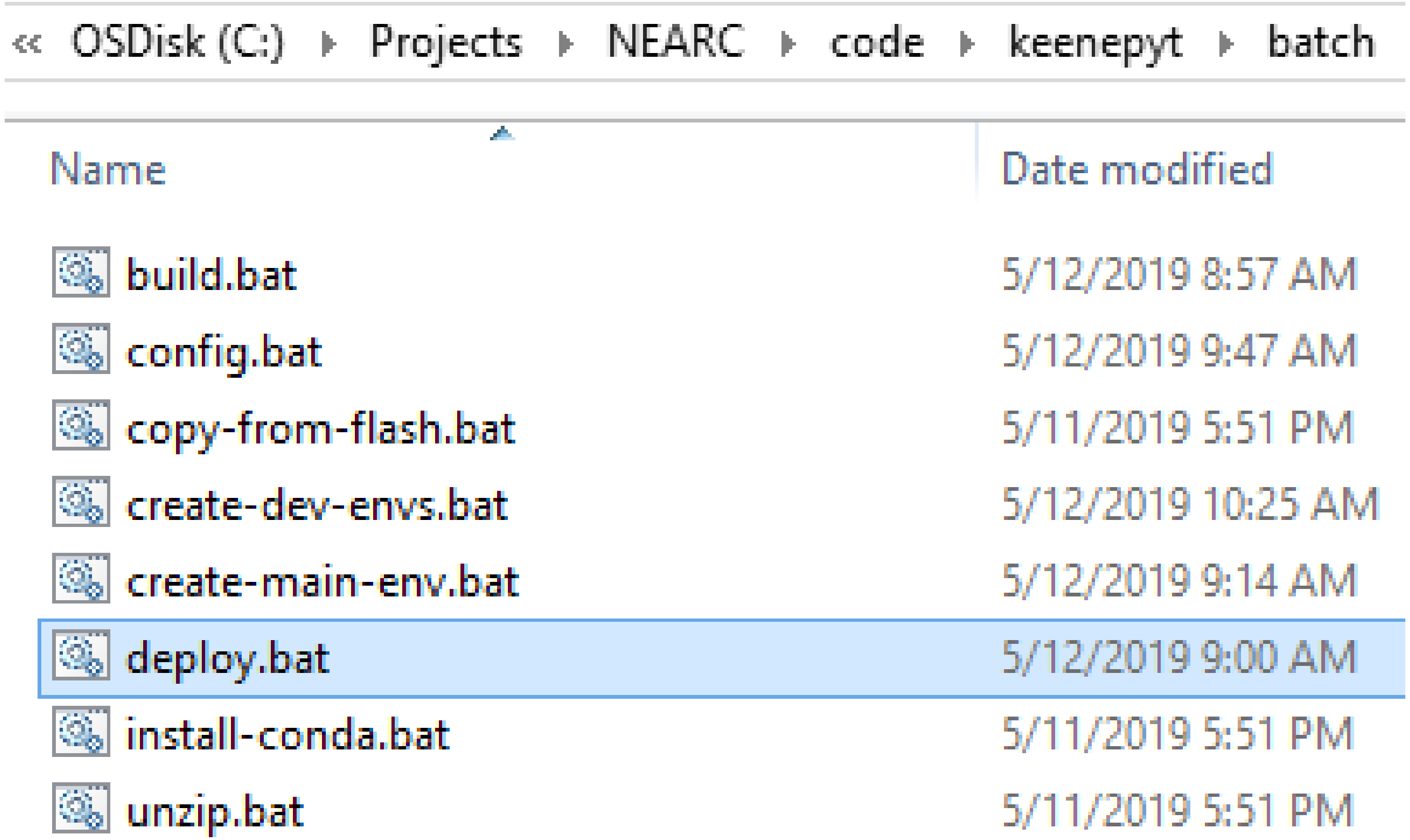
Active	Environments	Clone	Remove
○	arcgispro-py3 C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3		
○	dev C:\Users\jswise\AppData\Local\ESRI\conda\envs\dev		
○	localtest C:\Users\jswise\AppData\Local\ESRI\conda\envs\localtest		
●	main C:\Users\jswise\AppData\Local\ESRI\conda\envs\main		

Deploy the package.

Run **deploy.bat**.

This creates a new channel, **C:\Projects\NEARC\channel**.

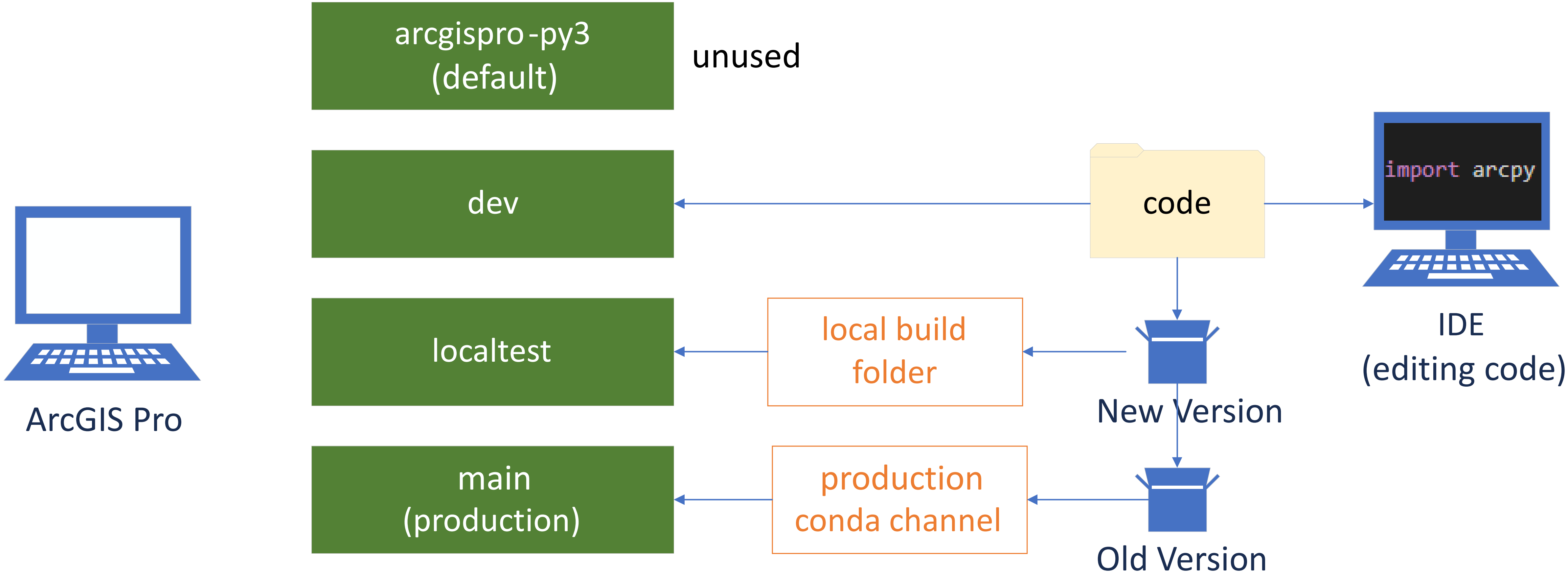
In real life, this would be on a server that your coworkers can see.



« OSDisk (C:) ▸ Projects ▸ NEARC ▸ code ▸ keenepyt ▸ batch

Name	Date modified
build.bat	5/12/2019 8:57 AM
config.bat	5/12/2019 9:47 AM
copy-from-flash.bat	5/11/2019 5:51 PM
create-dev-envs.bat	5/12/2019 10:25 AM
create-main-env.bat	5/12/2019 9:14 AM
deploy.bat	5/12/2019 9:00 AM
install-conda.bat	5/11/2019 5:51 PM
unzip.bat	5/11/2019 5:51 PM

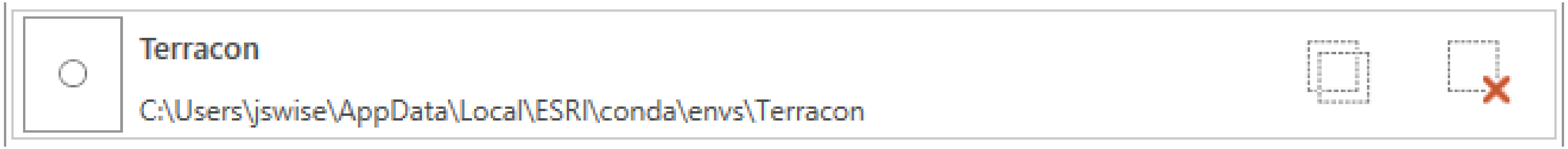
Our environments (review)



+build environment (separate conda installation)

Install the package in main.

In real life, you might want to name this environment with the name of your organization.

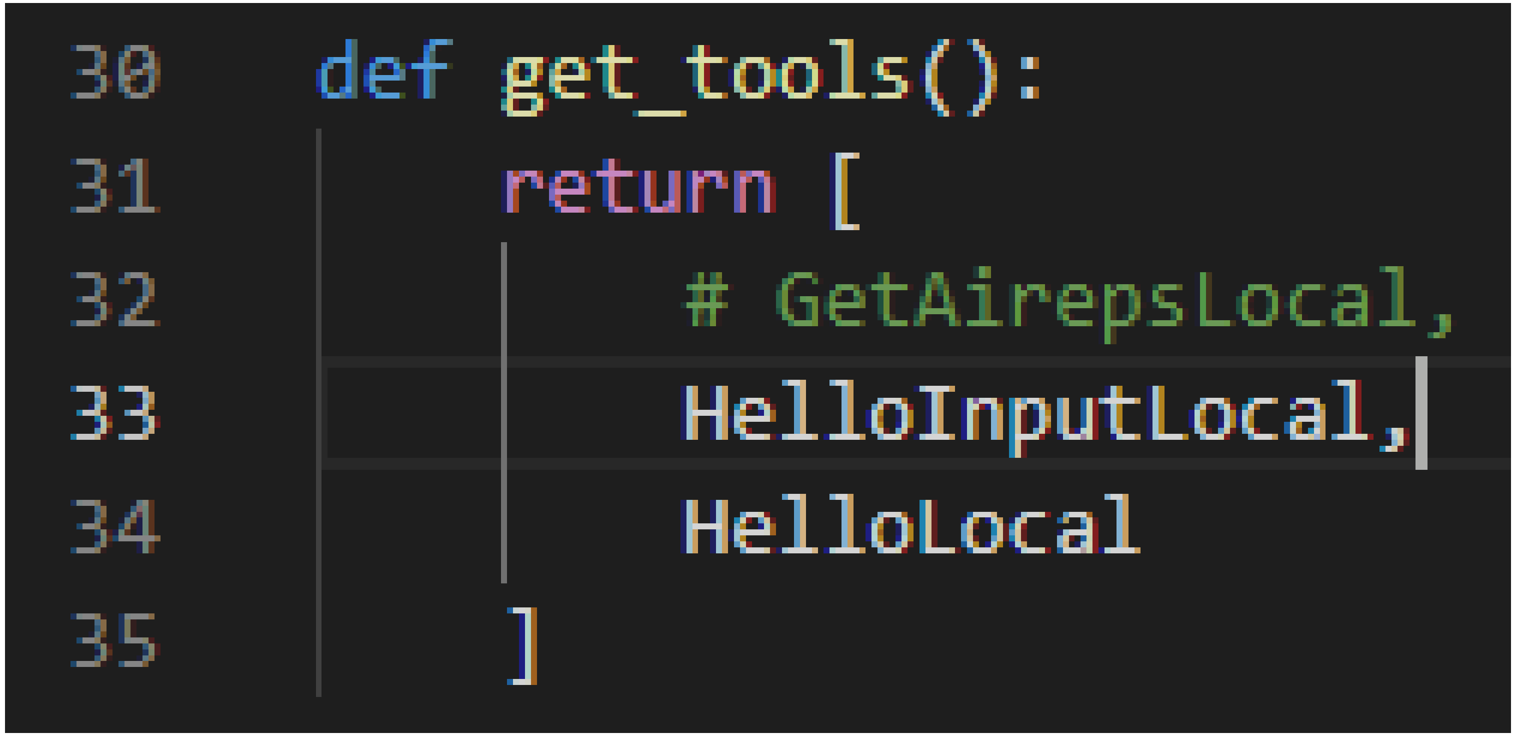


○ Terracon

C:\Users\jswise\AppData\Local\ESRI\conda\envs\Terracon

Improve the code.

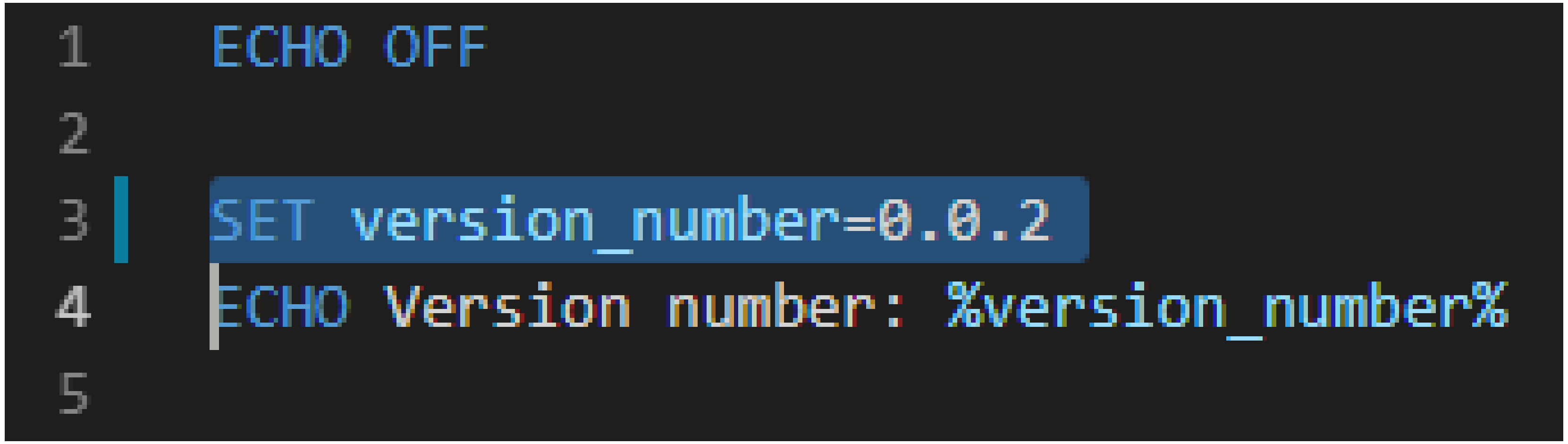
Add another tool by uncommenting it in *src.keenepyt.tools.toolfactory.get_tools*.



```python
def get_tools():
    return [
        # GetAirepsLocal,
        HelloInputLocal,
        HelloLocal
    ]
```

Improve the code.

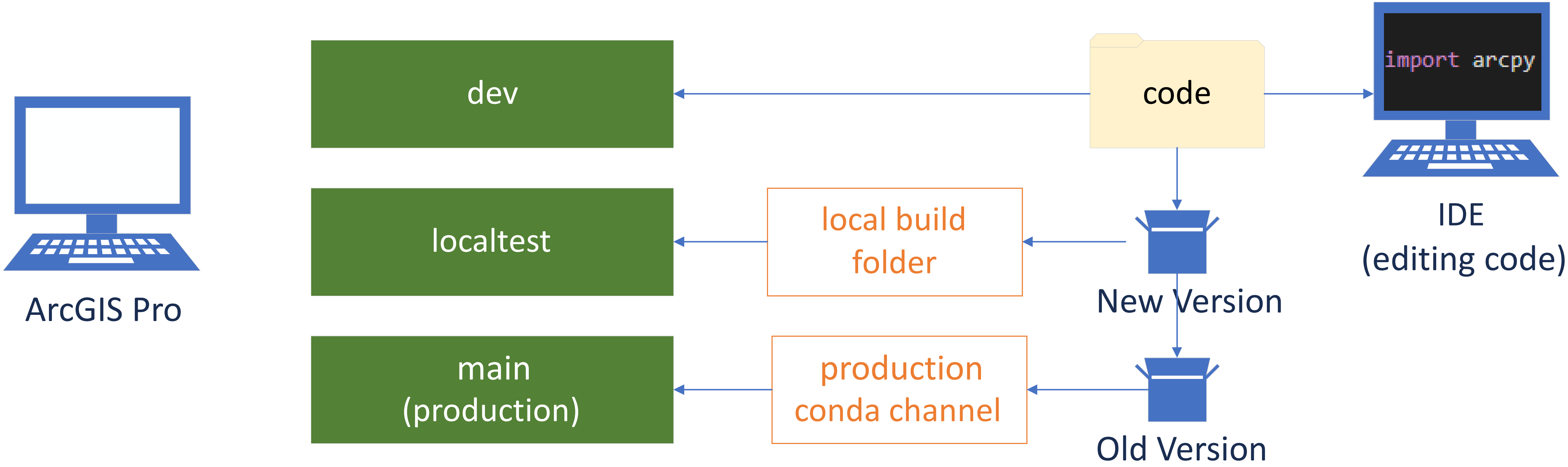
In config.bat, update the version number from 0.0.1 to 0.0.2.



```
ECHO OFF

SET version_number=0.0.2
ECHO Version number: %version_number%

```

Run build.bat again.

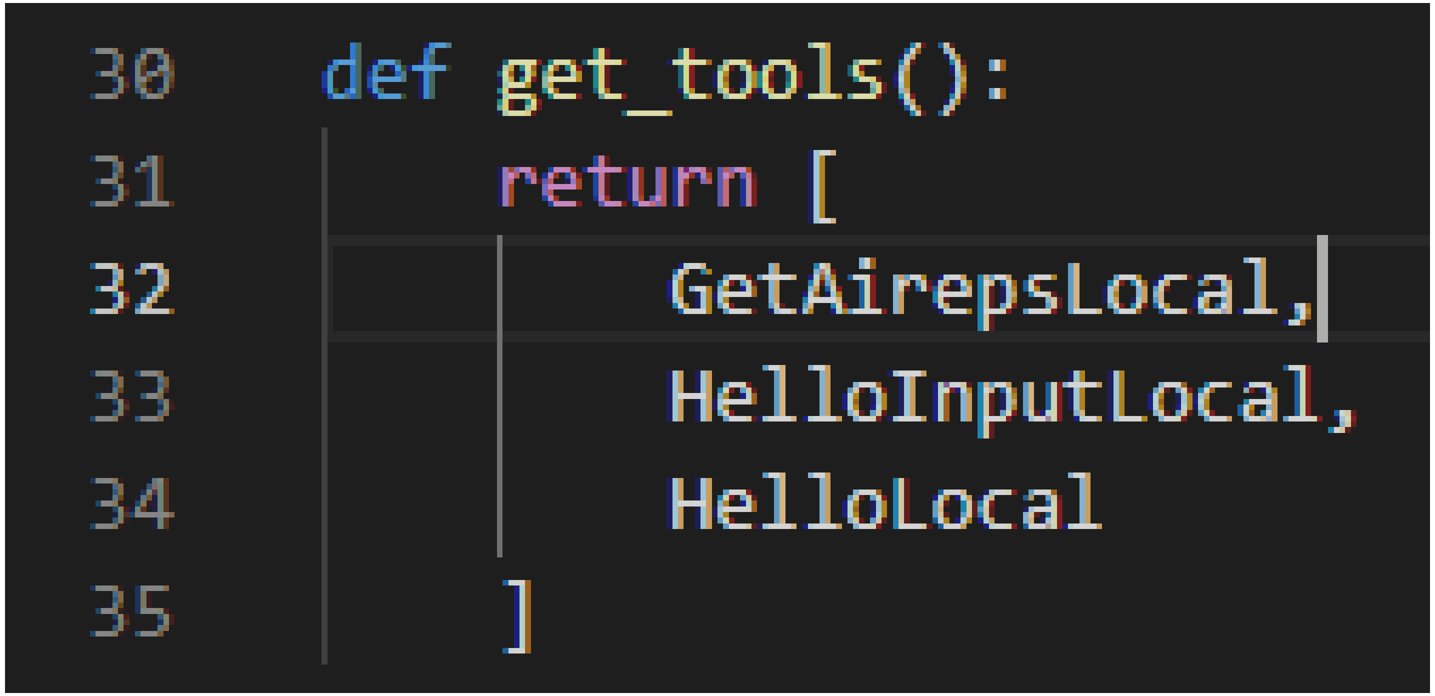
Now localtest is different from main.



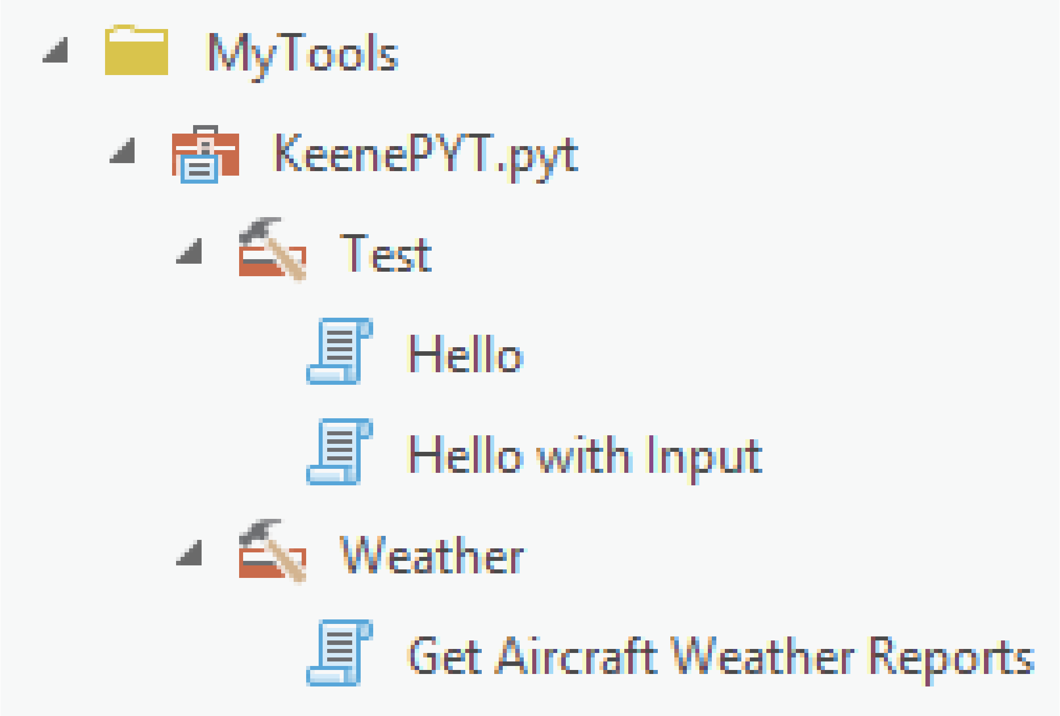
Add another tool.

Now dev, localtest, and main are all different.

Restart Pro to see the change.

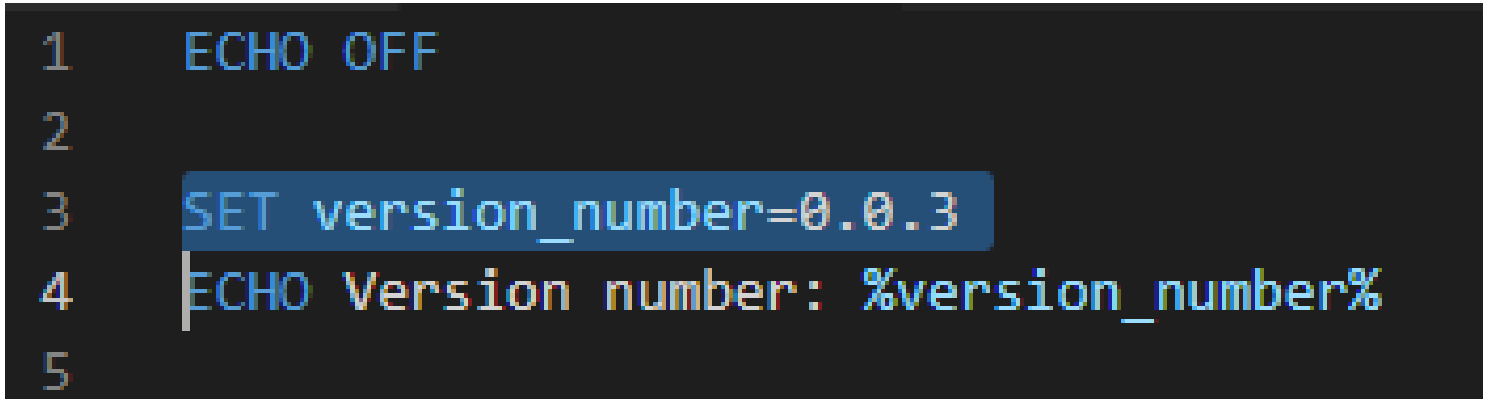


```
def get_tools():
    return [
        GetAirepsLocal,
        HelloInputLocal,
        HelloLocal
    ]
```



Add another tool.

Update the version number again.



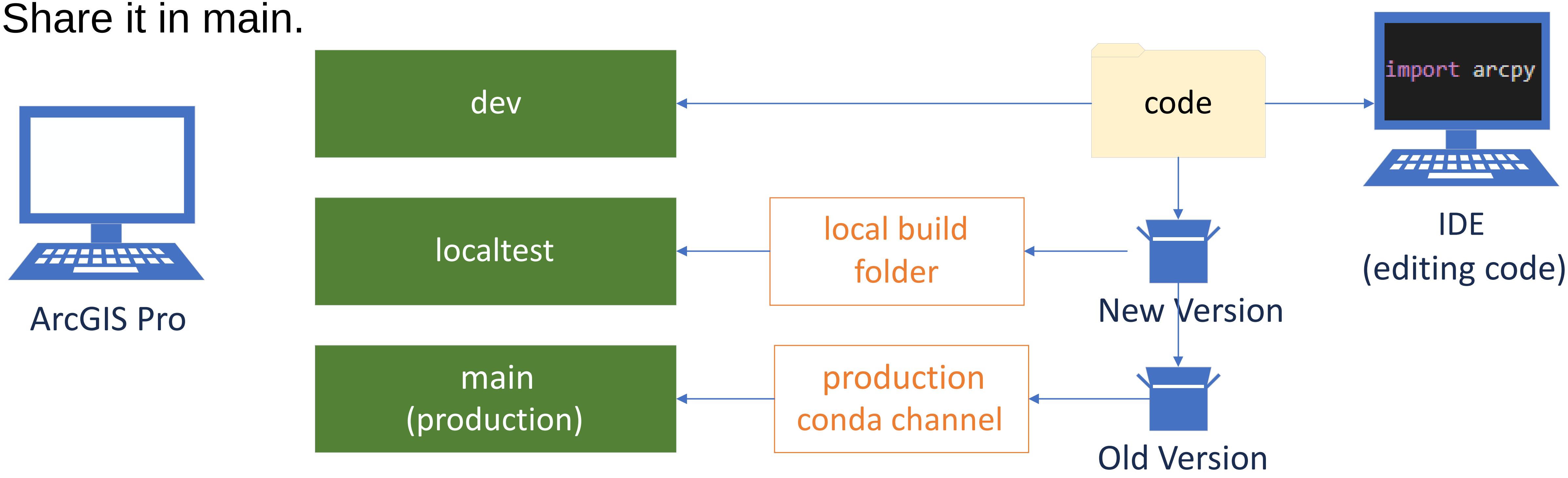
```
ECHO OFF

SET version_number=0.0.3
ECHO Version number: %version_number%

```

Development cycle

Write & test code in dev.

Test the package in localtest.

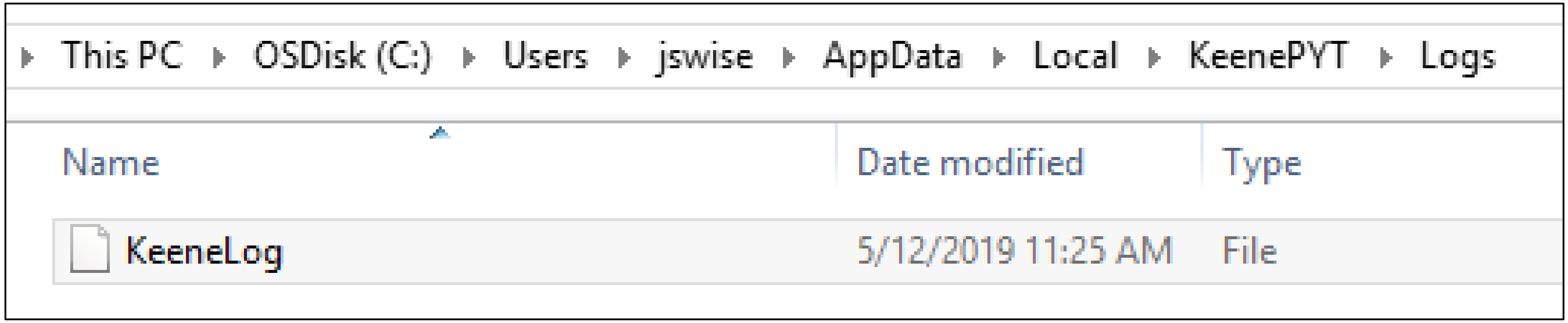
Share it in main.



Terracon

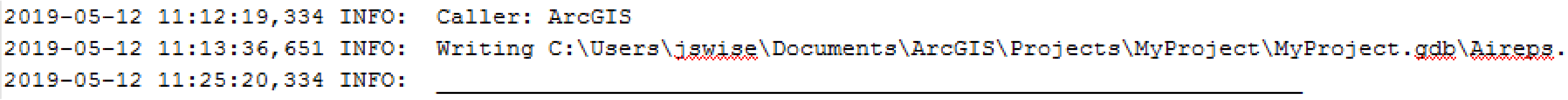
More Python fun

Check out the log file.



This PC ▸ OSDisk (C:) ▸ Users ▸ jswise ▸ AppData ▸ Local ▸ KeenePYT ▸ Logs

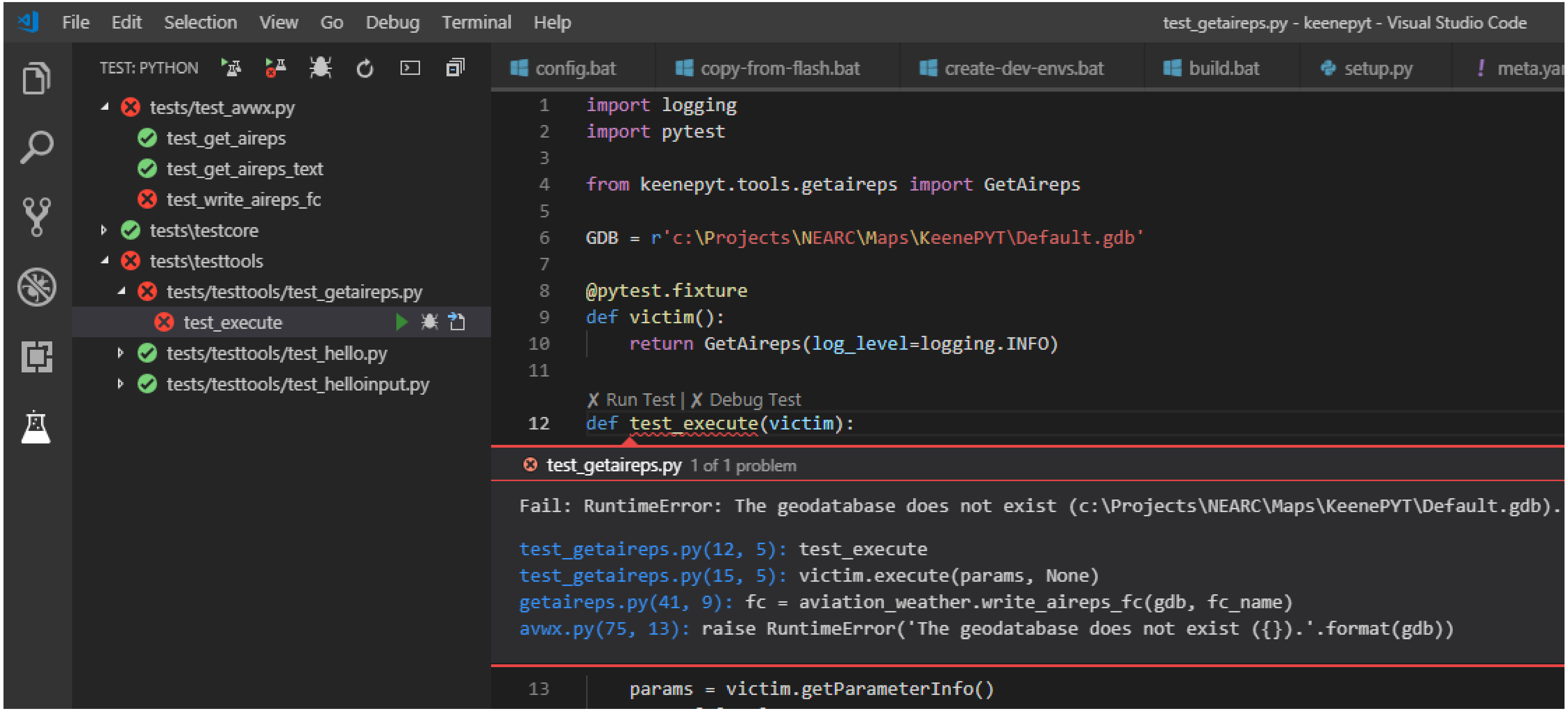
Name	Date modified	Type
KeeneLog	5/12/2019 11:25 AM	File



```
2019-05-12 11:12:19,334 INFO:  Caller: ArcGIS
2019-05-12 11:13:36,651 INFO:  Writing C:\Users\jswise\Documents\ArcGIS\Projects\MyProject\MyProject.gdb\Aireps.
2019-05-12 11:25:20,334 INFO:  ______________________________________________________________
```

More Python fun

Use PyTest to run unit tests.



```
tests/test_avwx.py
    test_get_aireps
    test_get_aireps_text
    test_write_aireps_fc
tests\testcore
tests\testtools
    tests/testtools/test_getaireps.py
        test_execute
    tests/testtools/test_hello.py
    tests/testtools/test_helloinput.py
```

```
import logging
import pytest

from keenepyt.tools.getaireps import GetAireps

GDB = r'c:\Projects\NEARC\Maps\KeenePYT\Default.gdb'

@pytest.fixture
def victim():
    return GetAireps(log_level=logging.INFO)

✗ Run Test | ✗ Debug Test
def test_execute(victim):
```

test_getaireps.py 1 of 1 problem

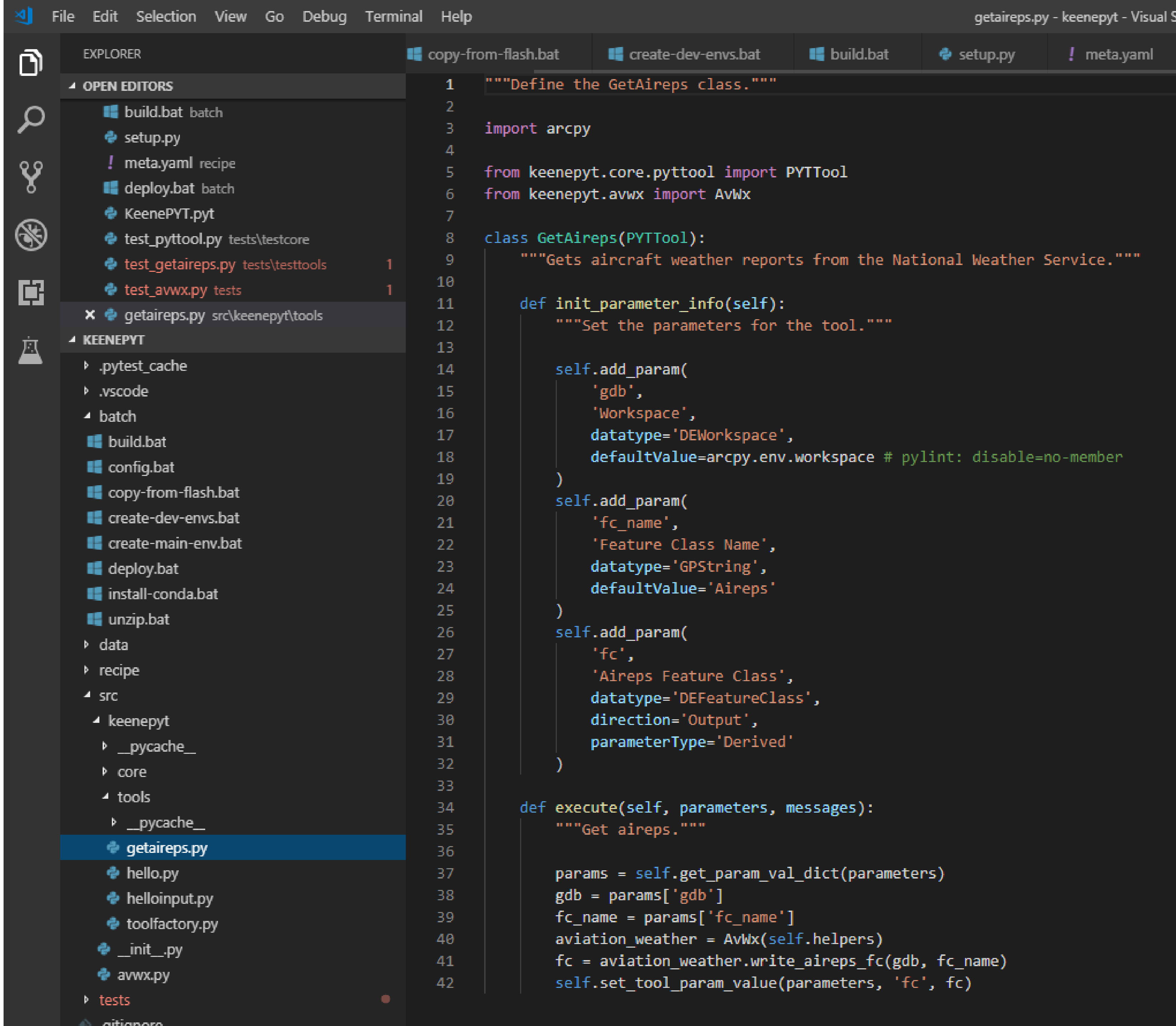
```
Fail: RuntimeError: The geodatabase does not exist (c:\Projects\NEARC\Maps\KeenePYT\Default.gdb).

test_getaireps.py(12, 5): test_execute
test_getaireps.py(15, 5): victim.execute(params, None)
getaireps.py(41, 9): fc = aviation_weather.write_aireps_fc(gdb, fc_name)
avwx.py(75, 13): raise RuntimeError('The geodatabase does not exist ({}).'.format(gdb))
```

```
    params = victim.getParameterInfo()
```

More Python fun

See how these tools work.

They’re not like the examples you’ll see elsewhere.

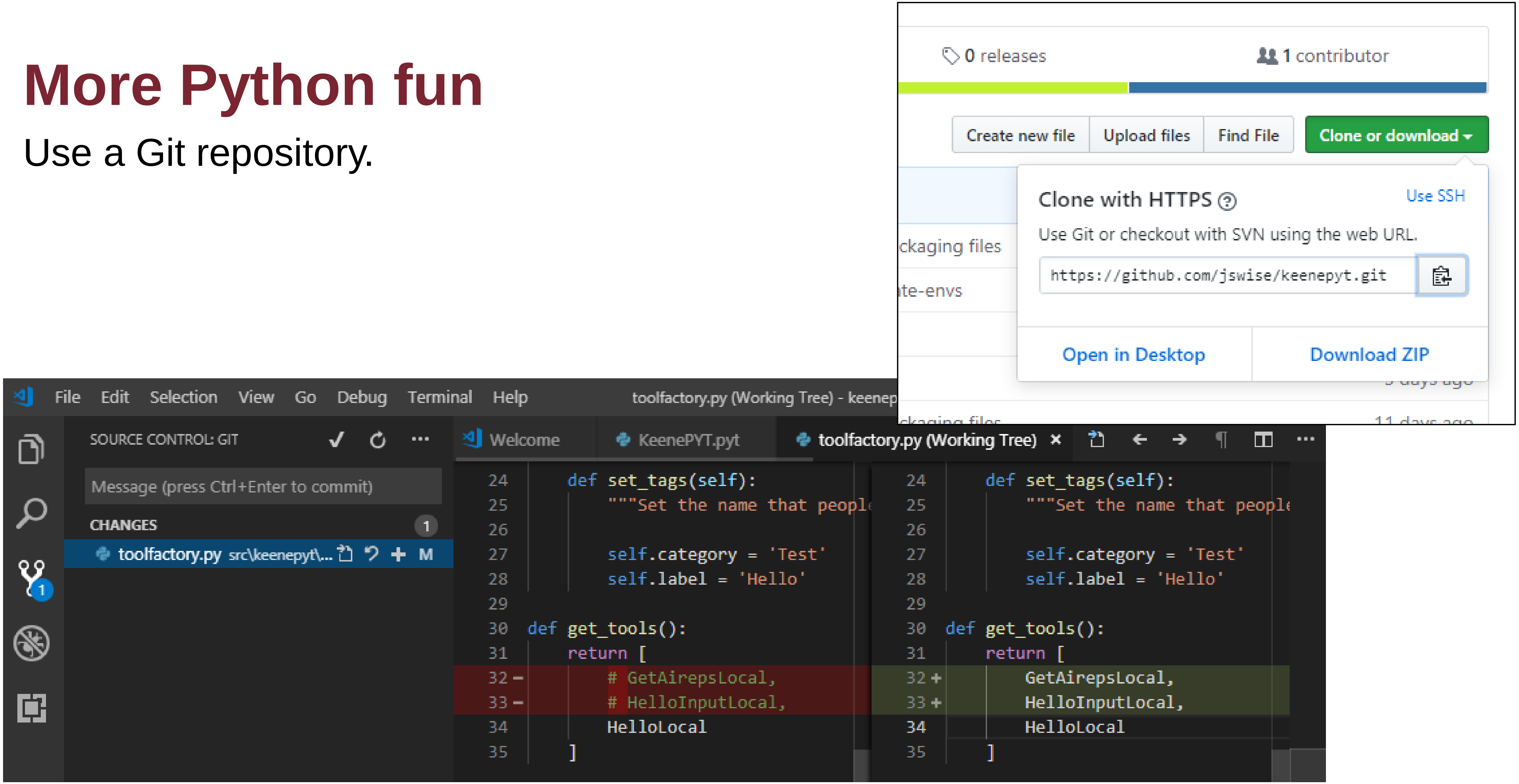


```python
"""Define the GetAireps class."""

import arcpy

from keenepyt.core.pyttool import PYTTool
from keenepyt.avwx import AvWx

class GetAireps(PYTTool):
    """Gets aircraft weather reports from the National Weather Service."""

    def init_parameter_info(self):
        """Set the parameters for the tool."""

        self.add_param(
            'gdb',
            'Workspace',
            datatype='DEWorkspace',
            defaultValue=arcpy.env.workspace # pylint: disable=no-member
        )
        self.add_param(
            'fc_name',
            'Feature Class Name',
            datatype='GPString',
            defaultValue='Aireps'
        )
        self.add_param(
            'fc',
            'Aireps Feature Class',
            datatype='DEFeatureClass',
            direction='Output',
            parameterType='Derived'
        )

    def execute(self, parameters, messages):
        """Get aireps."""

        params = self.get_param_val_dict(parameters)
        gdb = params['gdb']
        fc_name = params['fc_name']
        aviation_weather = AvWx(self.helpers)
        fc = aviation_weather.write_aireps_fc(gdb, fc_name)
        self.set_tool_param_value(parameters, 'fc', fc)
```

Terracon

More Python fun

Use a Git repository.



Clone with HTTPS

Use Git or checkout with SVN using the web URL.

https://github.com/jswise/keenepyt.git

toolfactory.py (Working Tree) - keenep

Message (press Ctrl+Enter to commit)

toolfactory.py src\keenepyt\...

```
    def set_tags(self):
        """Set the name that people

        self.category = 'Test'
        self.label = 'Hello'

def get_tools():
    return [
-       # GetAirepsLocal,
-       # HelloInputLocal,
        HelloLocal
    ]
```

```
    def set_tags(self):
        """Set the name that people

        self.category = 'Test'
        self.label = 'Hello'

def get_tools():
    return [
+       GetAirepsLocal,
+       HelloInputLocal,
        HelloLocal
    ]
```

Jason Wise

jason.wise@Terracon.com
https://github.com/jswise

Terracon

RESPONSIVE.
RESOURCEFUL.
RELIABLE.