



NSF's National Optical-Infrared  
Astronomy Research Laboratory



# Python Tutorial Series 2019

## Your code pretty and Healthy

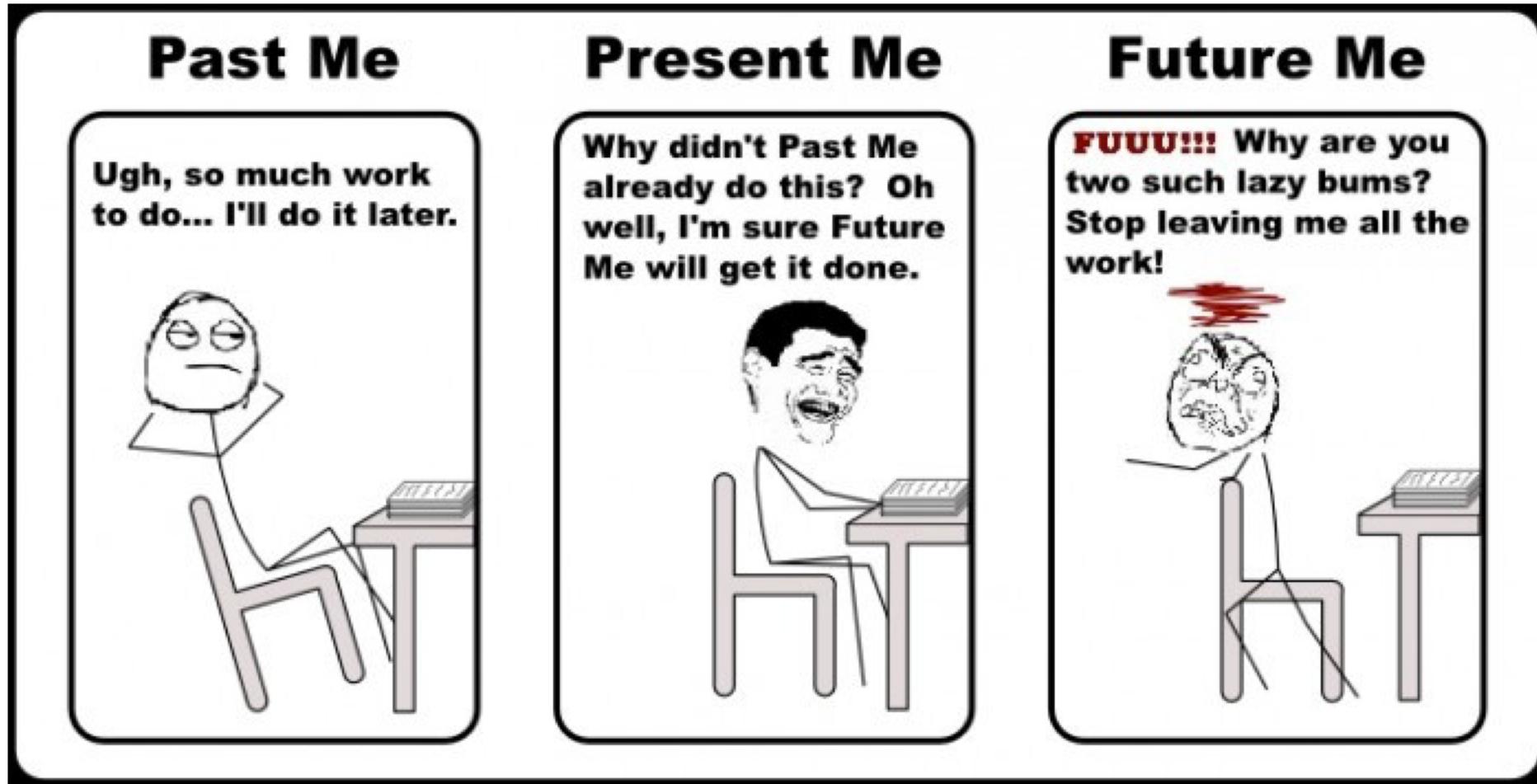
*Ph.D. Bruno C. Quint – bquint@gemini.edu*

*NSF's National Optical-Infrared Astronomy Research Laboratory*

# Contents

- Code Style
- Documentation
- Testing
- Project Structure
- Versioning and Version Control

# Introduction



# Code Style

```
>>> import this
```

## The Zen Of Python

...  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.

...

# Code Style – The PEP8

Why write if others can't read?

# Code Style – The PEP8

Why write if others can't read?

# Code Style – The PEP8

Why write if others can't read?

Indentation – Always use (four) spaces

```
for i in range(10):
....x = i * 2
....print(x)
```

Importing packages

Don't

```
from numpy import *
```

Do

```
import numpy as np
```

# Code Style – The PEP8

Why write if others can't read?

Two empty  
lines  
before  
functions  
and classes

```
GLOBAL_VARIABLE_NAME = "example"  
variable_name = 5  
  
def function_name(x, y):  
    ...return x + y  
  
class className:  
    ...  
    ...def method_name(self):  
        ...do something
```

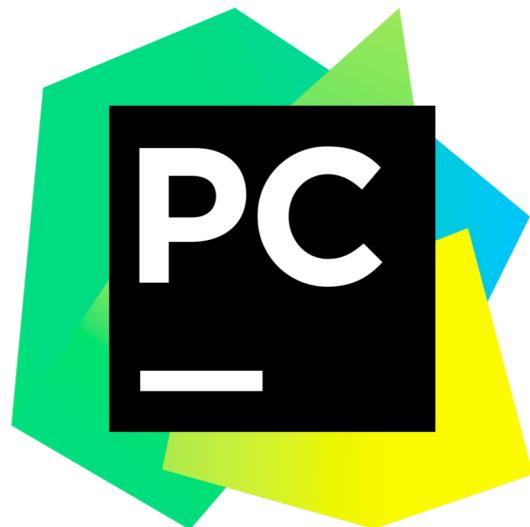
Spaces between  
operators and  
characters

# Code Style – Tools

Why write if others can't read?

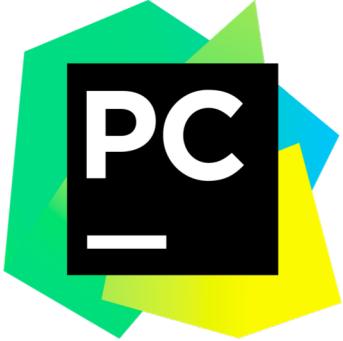
Tools for code format

- [PyLint](#)
- [flake8](#)
- [Black](#)
- [PyCharm CE](#)



# Code Style – Tools

Why write if others can't read?



```
1
2     import numpy as np
3
4     def myFunction():
5         # do something great
6
7 PEP 8: expected 2 blank lines, found 1 :  
Reformat file ↕ More actions... ↕
```

# Documentation

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""


```

## My Sample Code

This is such a simple documentation! The only thing I have to do is to add three single quotes ('') or double quotes (") together to start a docstring block! Then just write down in a few words what you want to do in your file.

by J. Bond - Feb 31, 202X

"""

# Documentation

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""It is ok if the docstring is just one line"""
```

# Documentation

```
def say_hello(name):
    """
    This function says hello to the person whose name
    is given as parameter.
    """
    print("Hello, {0}!".format(name))
```

# Documentation The NumPy Style

```
def say_hello(name):
    """
    When adding documentation, think of "what?", "why?" and "How?".

    Parameters
    -----
    name : str
        Name of the person you want to greet.

    """
    print("Hello, {}!".format(name))
```

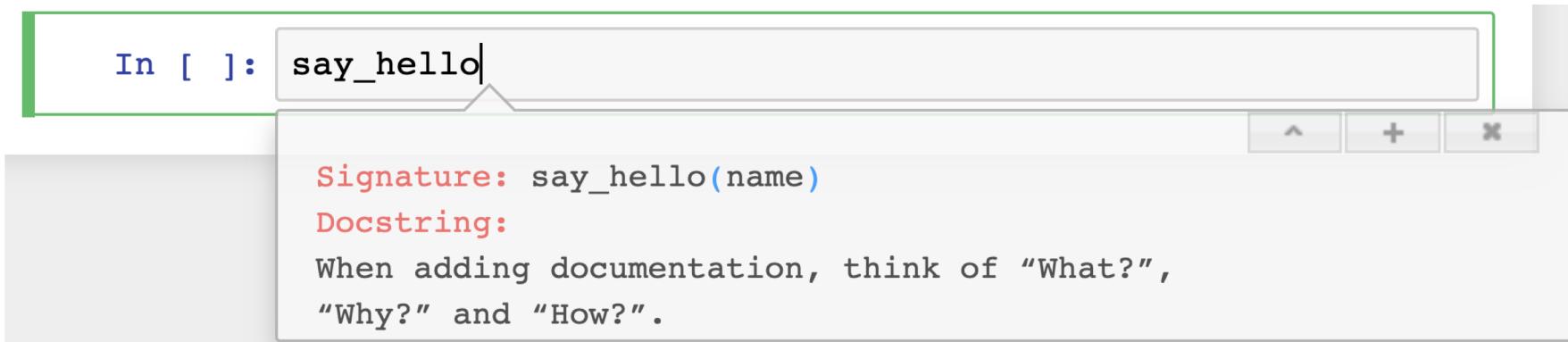
# Documentation

```
In [2]: say_hello?
```

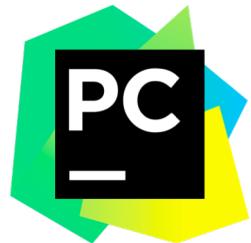
**Signature:** say\_hello(name)

**Docstring:**

When adding documentation, think of “What?”,  
“Why?” and “How?”.



# Documentation



say\_hello

```
/Users/bquint/Repos/DRAGONS/.jenkins/local  
def say_hello(name: str) -> None
```

Params: name – Name of the person you want to greet.

⋮

# Testing

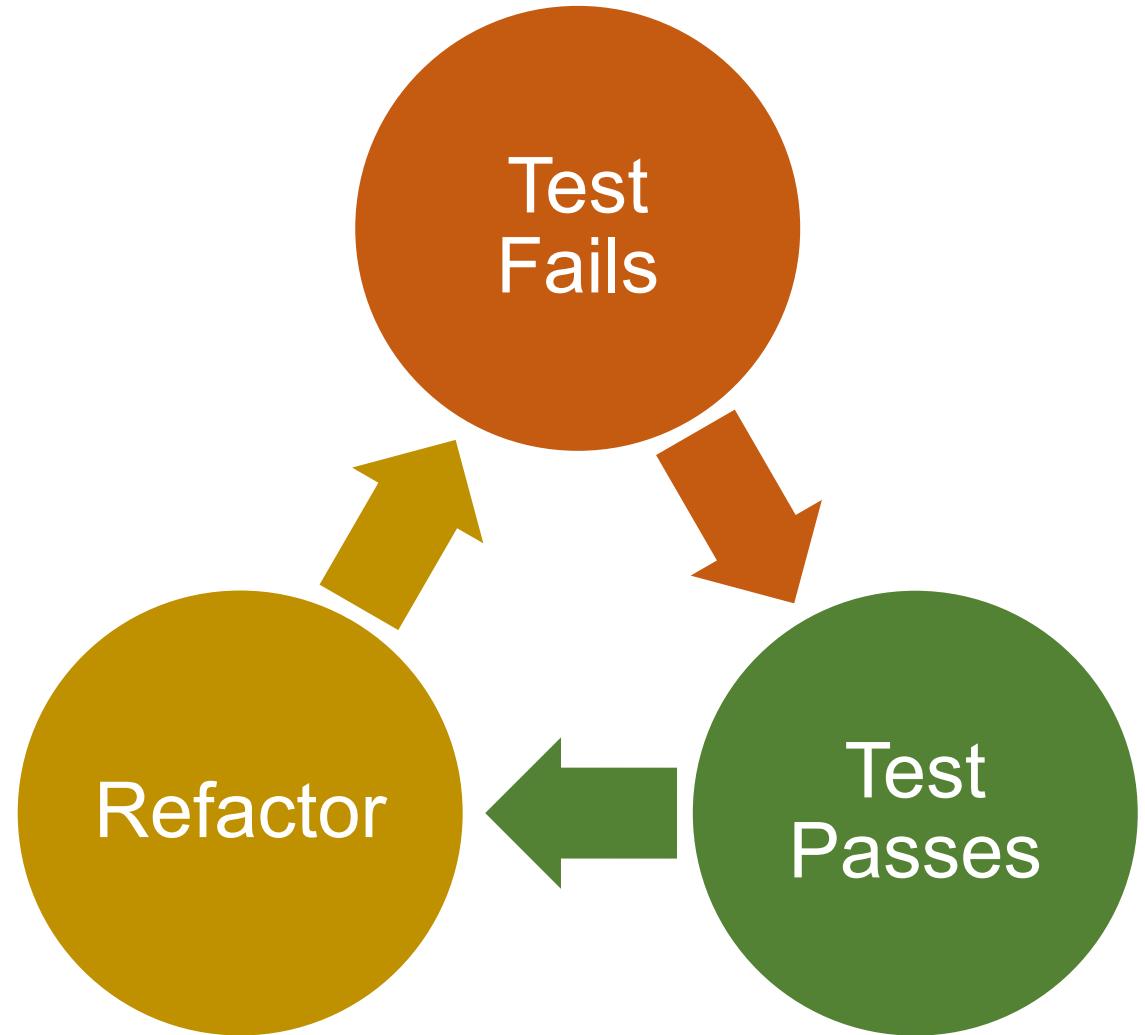
The refactoring cat



# Testing

Test-Driven Development

The refactoring cat



# Testing

Tools for code format

- [unittest](#)
- [nosetest](#)
- [pytest](#)
- [numpy.testing](#)



**pytest**

- Tests can be top simple top-level functions
- Simple result assertion
- Fixtures provide modularized test suite
- Tests can be easily parametrized

# Testing

```
# contents of test\_capital\_case.py
from manipulate_strings import capital_case

def test_capital_case():
    assert capital_case('semaphore') == 'Semaphore'
```

# Testing

```
# contents of test\_capital\_case.py
from manipulate_strings import capital_case

def test_capital_case():
    assert capital_case('semaphore') == 'Semaphore'
```

```
# contents of manipulate\_strings.py
def capital_case(my_str):
    return my_str.capitalize()
```

# Testing

```
pts7_code_testing — bquint@bquint-ml1 — zsh — 99x13
.._code_testing
$ pytest
===== test session starts =====
platform darwin -- Python 3.6.9, pytest-5.2.2, py-1.8.0, pluggy-0.13.0
rootdir: /Users/bquint/Repos/Python-Tutorial-Series/Examples/pts7_code_testing
plugins: asdf-2.4.2, arraydiff-0.3, remotedata-0.3.2, doctestplus-0.4.0, openfiles-0.4.0
collected 1 item

test_manipulate_string.py .

===== 1 passed in 0.01s =====
(dev) (master *) bquint@bquint-ml1 [~/Repos/Python-Tutorial-Series/Examples/pts7_code_testing]
$
```

# Testing

```
# contents of test\_capital\_case.py
...
def test_raises_exception_on_non_string_arguments():
    with pytest.raises(TypeError):
        capital_case(9)
```

# Testing

```
pts7_code_testing — bquint@bquint-ml1 — zsh — 99x22
.._code_testing
test_manipulate_string.py .F [100%]

=====
 FAILURES =====
----- test_raises_exception_on_non_string_arguments -----
def test_raises_exception_on_non_string_arguments():
    with pytest.raises(TypeError):
        manipulate_strings.capitalize(9)
>     my_str = 9
def capitalize(my_str):
    return my_str.capitalize()
E     AttributeError: 'int' object has no attribute 'capitalize'

manipulate_strings.py:8: AttributeError
=====
 1 failed, 1 passed in 0.07s =====
(dev) (master *) bquint@bquint-ml1 [~/Repos/Python-Tutorial-Series/Examples/pts7_code_testing]
$ Discovering our Universe Together
1 ↵ 24
```

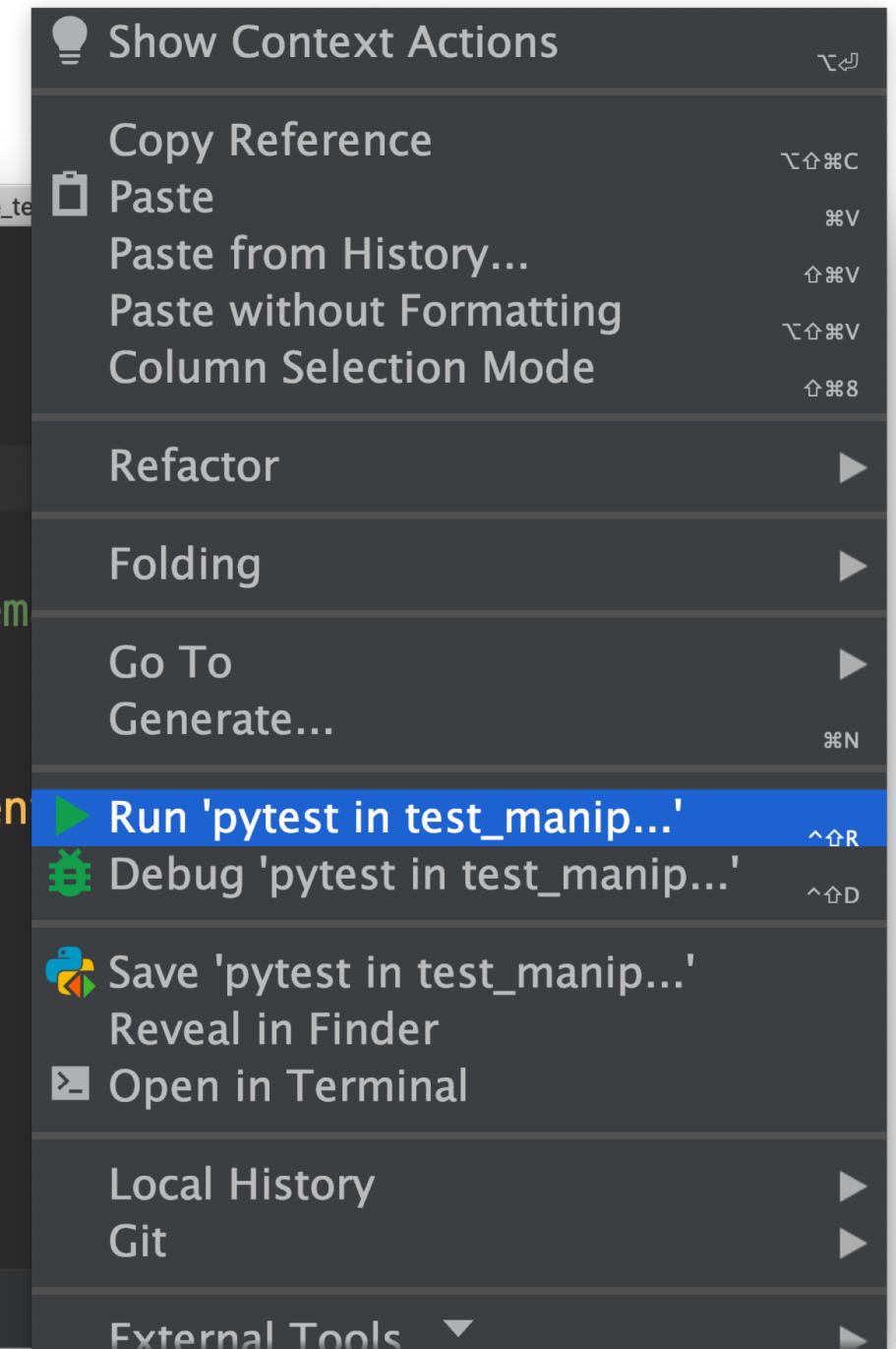
# Testing

```
Python-Tutorial-Series [~/Repos/Python-Tutorial-Series] - .../Examples/pts7_code_testing/test_manipulate_string.py

6  import manipulate_strings
7  import pytest
8
9
10 ► def test_capital_case():
11     assert manipulate_strings.capital_case('semaphore') == 'Semaphore'
12
13
14 ► def test_raises_exception_on_non_string_arguments():
15     with pytest.raises(TypeError):
16         manipulate_strings.capital_case(9)
17
18
19 ► if __name__ == '__main__':
20     pytest.main()
21
```

# Testing

```
Python-Tutorial-Series [~/Repos/Python-Tutorial-Series] - .../Examples/pts7_code_te  
6 import manipulate_strings  
7 import pytest  
8  
9  
10 ► def test_capital_case():  
11     assert manipulate_strings.capital_case('sem') == 'SEM'  
12  
13  
14 ► def test_raises_exception_on_non_string_argument():  
15     with pytest.raises(TypeError):  
16         manipulate_strings.capital_case(9)  
17  
18  
19 ► if __name__ == '__main__':  
20     pytest.main()  
21  
NSF
```



# Testing

Python-Tutorial-Series [~/Repos/Python-Tutorial-Series] - .../Examples/pts7\_code\_testing/test\_manipulate\_string.py

```
8
9
10 ► def test_capital_case():
11     assert manipulate_strings.capitalize('semaphore') == 'Semaphore'
12
13
```

Run:  pytest in test\_manipulate\_string.py X

Test Results

Test	Result	Time
test_manipulate_string	Failed	0 ms
test_capital_case	Passed	0 ms
test_raises_exception_on_r	Failed	0 ms

Tests failed: 1, passed: 1 of 2 tests -

Test Results

- test\_manipulate\_string 0 ms
- test\_capital\_case 0 ms
- test\_raises\_exception\_on\_r 0 ms

Testing started at 11:13 ...  
/Users/bquint/miniconda3/envs/dev/b  
Launching pytest with arguments /Us  
===== test  
platform darwin -- Python 3.6.9, py  
cachedir: pytest\_cache

NSF's >>  
Astronomy Research Laboratory, >> 27

# Testing

```
def test_can_perform_task():
    ...
    assert test_was_performed()
```

Yes

```
def test_cpt():
    ...
    assert test_was_performed()
```

No

```
def test_1():
    ...
    assert test_was_performed()
```

NEVER!!!

# Project Structure

```
+--- README.rst
+--- LICENSE
+--- setup.py
+--- requirements.txt

+--- docs/
|   +--- conf.py
|   +--- index.rst
...
+--- sample/
|   +--- __init__.py
|   +--- core.py
|   +--- helpers.py
...
+--- tests/
|   +--- __init__.py
|   +--- conftest.py
|   +--- test_core.py
...
...
```

# Project Structure

```
+--- README.rst  
+--- LICENSE  
+--- setup.py  
+--- requirements.txt  
  
+--- docs/  
|   +--- conf.py  
|   +--- index.rst  
  
...
```

```
+--- sample/  
|   +--- __init__.py  
|   +--- core.py  
|   +--- helpers.py  
|   +--- tests/  
|       +--- __init__.py  
|       +--- conftest.py  
|       +--- test_core.py  
  
...
```

# Project Structure

```
+--- README.rst
+--- LICENSE
+--- setup.py
+--- requirements.txt
+--- .gitignore
+--- bin/
+--- scripts/
+--- docs/
|   +--- conf.py
|   +--- index.rst
...
+--- sample/
|   +--- __init__.py
|   +--- core.py
|   +--- helpers.py
|   +--- tests/
|       +--- __init__.py
|       +--- conftest.py
|       +--- test_core.py
...
...
```

# Project Structure setup.py

## Using [setuptools](#)

```
# contents of setup.py
from setuptools import setup, find_packages

setup(
    name="HelloWorld",
    version="0.1",
    packages=find_packages(),
)
```

# Project Structure setup.py and setup.cfg

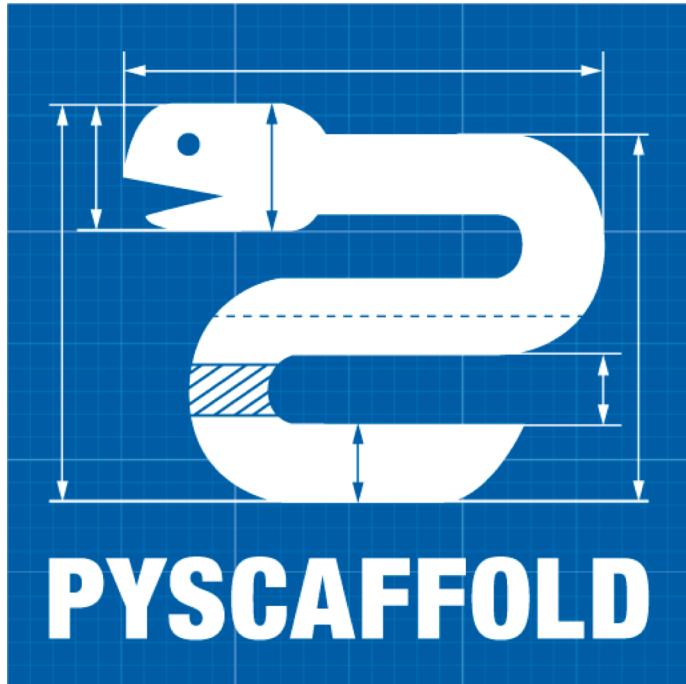
## Using [setuptools](#)

```
# contents of setup.cfg
[metadata]
name = HelloWorld
version = 0.1

[options]
packages = find:
```

```
# contents of setup.py
from setuptools import setup, find_packages
setup()
```

# Project Structure



```
$ putup my_project
```

```
$ ls my_project
```

AUTHORS.rst	requirements.txt
CHANGELOG.rst	setup.cfg
LICENSE.txt	setup.py
README.rst	src
docs	tests

Link to [PyScaffold](#) documentation

# Versioning and Version Control

## Semantic Versioning 2.0.1

MAJOR version when you make incompatible API changes

MINOR version when you add functionality in a backwards compatible manner

PATCH version when you make backwards compatible bug fixes.

## Calendar Versioning 2019.0.1

YYYY format year of release

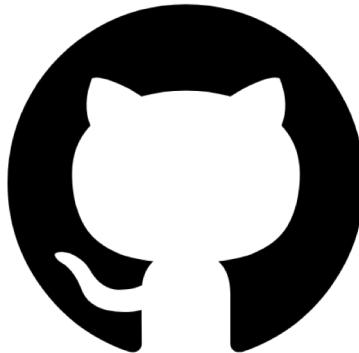
MINOR release version

PATCH version

# Versioning and Version Control



[Git Online Book](#) (free)

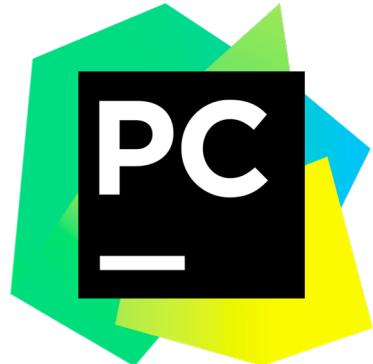


[GitHub](#) (free)



[GitLab](#)

(paid – available for Gemini Users)



[PyCharm CE](#) (free)



[GitKraken](#) (free)

# Versioning and Version Control



The screenshot shows the GitKraken Pro Free Trial interface for a GitHub repository named "Python-Tutorial-Series". The repository has one branch, "master", which is currently checked out. The timeline shows a series of commits from various users, with the most recent commit being a merge from the remote-tracking branch "origin/master". The commit history includes:

- Merge remote-tracking branch 'origin/master'
- Added pytest example
- Update README.md
- Updated missing slides
- Updated README.md
- Updated PTS2019 - Astropy 1
- Updated notebook and PPTX on AstroPy and Image Anal...
- Updated notebook on AstroPy and Image Analysis
- Created new Notebook for Astropy/Images
- Moved data to Notebooks/data
- Added PTS2019: Astropy - Image
- Added sample files
- Updated Jupyter notebook with exercise
- Update environment.yml
- Update README.md
- Updated DV2 notebook

The right panel displays the current state of the repository, showing 1 file change on the master branch. It lists "Unstaged Files" containing "Presentations/PTS2019\_Python\_Projecs.pptx" and provides options to "Stage all changes", "Commit Message", "Summary", and "Description". A large green button at the bottom right says "Stage files/changes to commit".

# Versioning and Version Control

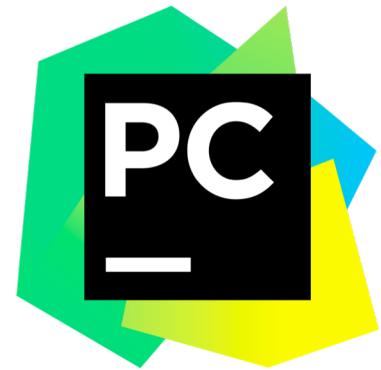


The screenshot shows the GitKraken Pro Free Trial interface for a GitHub repository named "Python-Tutorial-Series". The repository has one branch, "master", which is currently checked out. The timeline shows a series of commits from various users, with the most recent commit being a merge from the remote-tracking branch "origin/master". The commit history includes:

- Merge remote-tracking branch 'origin/master'
- Added pytest example
- Update README.md
- Updated missing slides
- Updated README.md
- Updated PTS2019 - Astropy 1
- Updated notebook and PPTX on AstroPy and Image Anal...
- Updated notebook on AstroPy and Image Analysis
- Created new Notebook for Astropy/Images
- Moved data to Notebooks/data
- Added PTS2019: Astropy - Image
- Added sample files
- Updated Jupyter notebook with exercise
- Update environment.yml
- Update README.md
- Updated DV2 notebook

The right panel displays the current state of the repository, showing 1 file change on the master branch. It includes sections for Unstaged Files (1) containing "Presentations/PTS2019\_Python\_Projecs.pptx" and Staged Files (0). There are fields for Commit Message, Summary, and Description, and a large green button at the bottom labeled "Stage files/changes to commit".

# Versioning and Version Control



The screenshot shows a Jupyter Notebook interface with a dark theme. The top navigation bar includes tabs for 'Version Control', 'Local Changes', 'Log', 'Console', and 'Update Info: 22-11-19 11:11'. The 'Log' tab is selected. The log pane displays a history of changes:

Commit	Author	Date
Merge remote-tracking branch 'origin/master'	bquint	22-11-19 11:11
Added pytest example	bquint	22-11-19 11:11
Update README.md	Bruno Quint*	16-08-19 10:29
Updated missing slides	Bruno Quint	13-08-19 16:12
Updated README.md	Bruno Quint*	13-08-19 14:33
Updated PTS2019 – Astropy 1	Bruno Quint	13-08-19 14:23
Updated notebook and PPTX on AstroPy and Image Analysis	Bruno Quint	12-08-19 11:21
Updated notebook on AstroPy and Image Analysis	Bruno Quint	09-08-19 17:44
Created new Notebook for Astropy/Images	Bruno Quint	08-08-19 10:54

The right side of the interface shows a code editor with a file named 'README.md'. The code contains several commits from Bruno Quint, with the most recent commit being the 'Update README.md' entry from August 16, 2019. The code editor has a 'Unified viewer' dropdown set to 'Unified viewer'.



Python-Tutorial-Series > Examples > pts7\_code\_testing > test\_manipulate\_string.py

pytest in test\_manipulate\_string.py

Project

manipulate\_strings.py x test\_manipulate\_string.py x

```
1 #!/usr/bin/env python
2 """
3     Tests for manipulate_strings.py
4 """
5
6 import manipulate_strings
7 import pytest
8
9
10 def test_capital_case():
11     assert manipulate_strings.capital_case('semaphore') == 'Semaphore'
12
13
14 def test_raises_exception_on_non_string_arguments():
15     with pytest.raises(TypeError):
16         manipulate_strings.capital_case(9)
17
18
19 if __name__ == '__main__':
20     pytest.main()
21
22 if __name__ == '__main__':
23
```

Structure

R Graphics R Packages

Version Control: Local Changes Log Console x Update Info: 22-11-19 11:11 x

Log

Merge remote-tracking branch 'origin/master' origin & master bquint 22-11-19 11:11  
Added pytest example bquint 22-11-19 11:11  
Update README.md Bruno Quint\* 16-08-19 10:29  
Updated missing slides Bruno Quint 13-08-19 16:12  
Updated README.md Bruno Quint\* 13-08-19 14:33  
Updated PTS2019 – Astropy 1 Bruno Quint 13-08-19 14:23  
Updated notebook and PPTX on AstroPy and Image Analysis Bruno Quint 12-08-19 11:21  
Updated notebook on AstroPy and Image Analysis Bruno Quint 09-08-19 17:44  
Created new Notebook for AstroPy/Images Bruno Quint 08-08-19 10:54

2: Favorites

2: Structure

6: TODO 9: Version Control Terminal Python Console

Tests failed: 1, passed: 1 (21 minutes ago)

19:24 LF UTF-8 4 spaces Git: master Python 3.6 (dev) Event Log 40

# Versioning and Version Control

## Write meaningful commit messages

	COMMENT	DATE
O	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
O	ENABLED CONFIG FILE PARSING	9 HOURS AGO
O	MISC BUGFIXES	5 HOURS AGO
O	CODE ADDITIONS/EDITS	4 HOURS AGO
O	MORE CODE	4 HOURS AGO
O	HERE HAVE CODE	4 HOURS AGO
O	AAAAAAA	3 HOURS AGO
O	ADKFJSLKDFJSDFKLJ	3 HOURS AGO
O	MY HANDS ARE TYPING WORDS	2 HOURS AGO
O	HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

<https://xkcd.com/1296/>

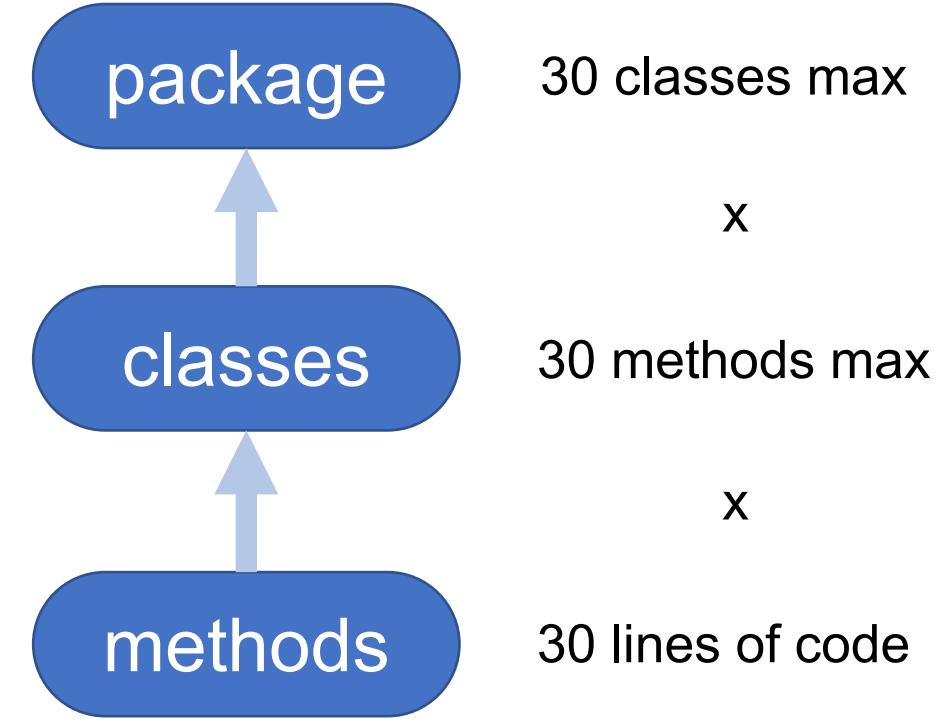
# Before you go...

## Rule of 3

- Three strikes and you refactor
- Duplication: hard to maintain
- Seeing some cases helps code design

## Rule of 30

If an element consists of more than 30 subelements, it is highly probable that there is a serious problem.



# Before you go...

```
def snellsLaw(n, t):  
    return np.rad2deg(np.arcsin(np.sin(np.deg2rad(t))/n))
```

# Before you go...

```
def snells_law(incident_angle, refractive_index):
    """
    ...
    theta = np.deg2rad(indicent_angle)
    n = refractive_index

    temp = np.sin(theta) / n
    temp = np.arcsin(theta)
    output_angle = np.rad2deg(temp)

    return output_angle
```

# Questions?

