Creating and running a Python unit test

What this tutorial is about

Here we'll see how PyCharm helps creating and running Python unit tests.

What this tutorial is not about

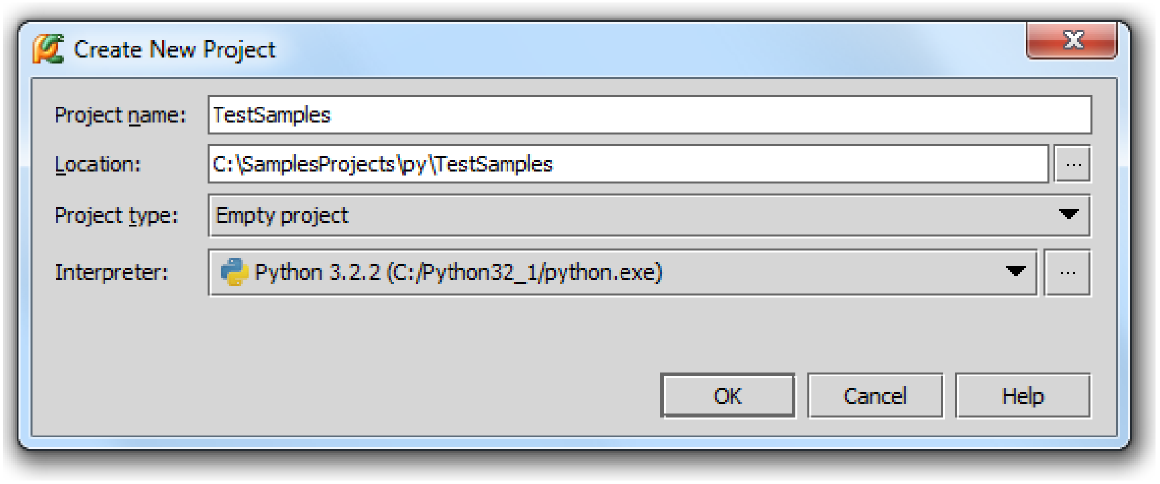
Python programming and writing Python unit tests is out of the scope of this tutorial.

Before you start...

Make sure that at least one Python interpreter (versions from 2.4 to 3.3) is properly installed on your computer.

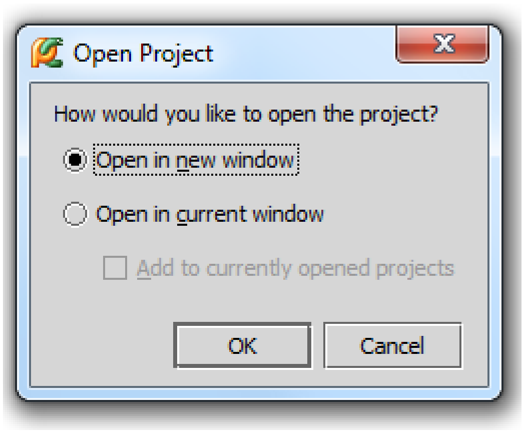
Creating a simple Python project

On the main menu, choose File | New Project.
In the Create New Project dialog box, specify the project name (let it be TestSamples), select the desired project type (here this is Empty project), and the Python interpreter you want to use.



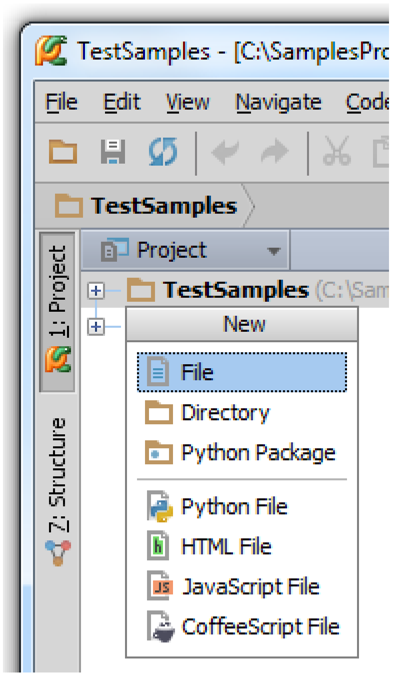
Project name: TestSamples
Location: C:\SamplesProjects\py\TestSamples
Project type: Empty project
Interpreter: Python 3.2.2 (C:/Python32_1/python.exe)

After you click OK, select the window to work with your new project in. In the Open Project dialog box, let us select the first option – open the new project in a separate window:

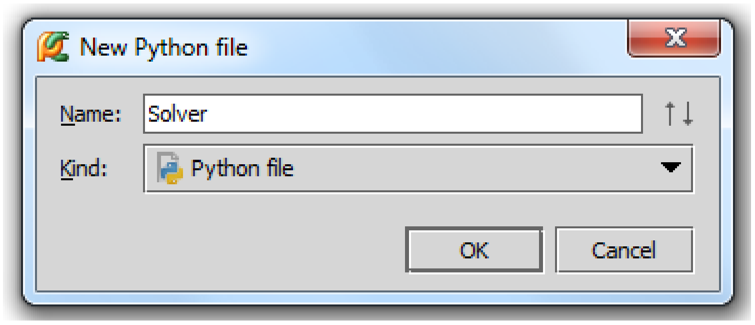

Open Project
How would you like to open the project?
Open in new window
Open in current window
Add to currently opened projects
OK
Cancel

Creating a Python class

Press Alt+Insert, and then choose Python file from the pop-up window:



In the New Python file dialog box, type the new file name:

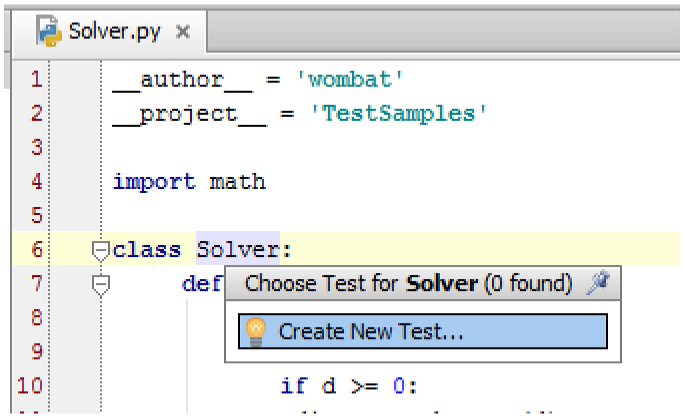


The new file opens in its tab in the editor, with the __author __ and __project__ variables already defined. Let's create a simple script that will solve a quadratic equation. Type the following code:

```
import math

class Solver:

    def demo(self, a, b, c):

           d = b ** 2 - 4 * a * c

           if d >= 0:

            disc = math.sqrt(d)

            root1 = (-b + disc) / (2 * a)

            root2 = (-b - disc) / (2 * a)

            print(root1, root2)

           else:

              raise Exception

Solver().demo(2,1,0)
```

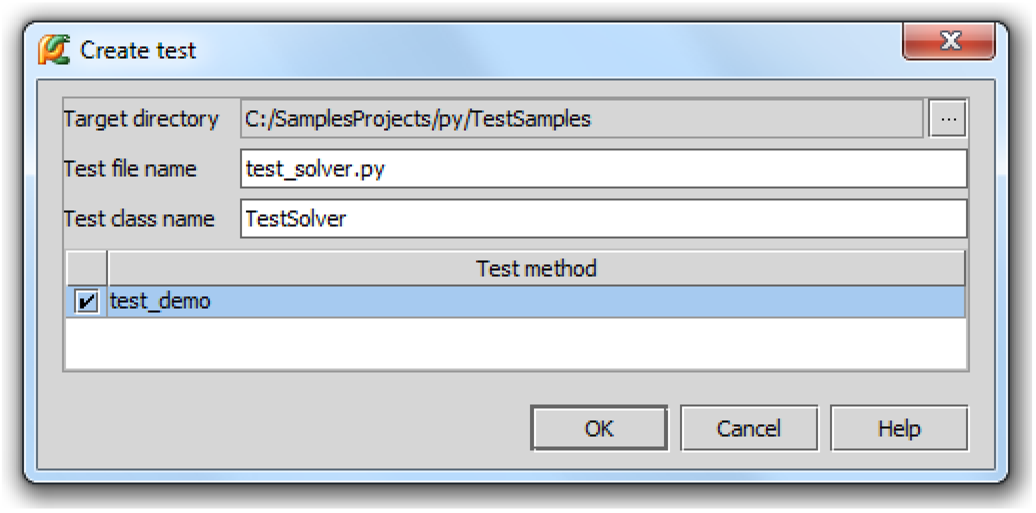
Creating test

Right-click the class name, and choose Go to | Test on the context menu, or just press Ctrl+Shift+T:



```
__author__ = 'wombat'
__project__ = 'TestSamples'

import math

class Solver:
    def

        if d >= 0:
```

In the Create test dialog box, just accept the suggested directory and names, and select the check box next to the suggested test_demo function:



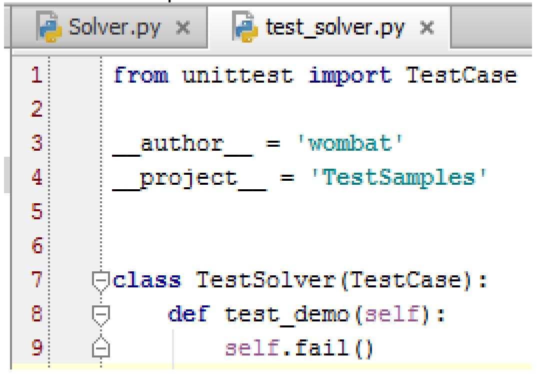
Target directory: C:/SamplesProjects/py/TestSamples

Test file name: test_solver.py

Test class name: TestSolver

	Test method
✓	test_demo

The test case opens in the editor:



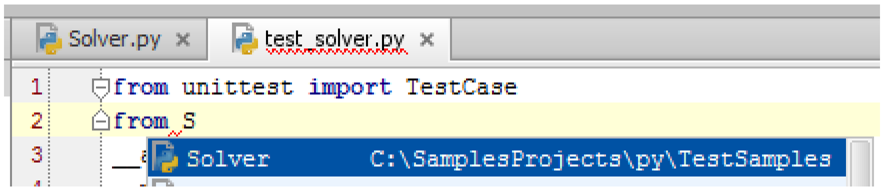
```
from unittest import TestCase

__author__ = 'wombat'
__project__ = 'TestSamples'


class TestSolver(TestCase):
    def test_demo(self):
        self.fail()
```

As you see, the created test case meets the requirements to the Python unit testing framework – it imports TestCase class from the unittest module, and the test function names begin with the string "test".

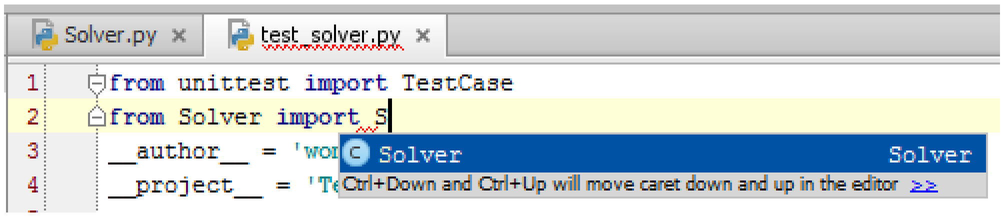
However, this test case is a kind of common place and requires some fine tuning. Let us first of all import the class we're going to test. To do that, add import statement:

```
from Solver import Solver
```

Use basic code completion while typing this statement: on pressing Ctrl+Space, PyCharm suggests suitable module and class names:



```
from unittest import TestCase
from S
```



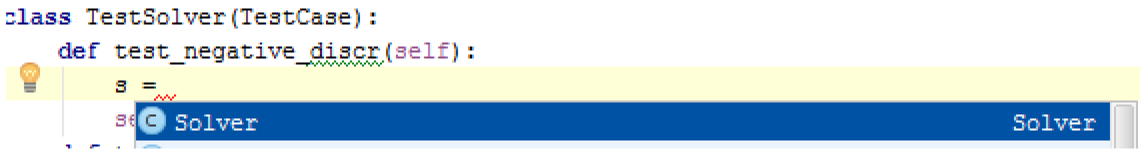
```
from unittest import TestCase
from Solver import S
__author__ = 'wo
__project__ = 'Te
```

Solver   Solver
Ctrl+Down and Ctrl+Up will move caret down and up in the editor >>

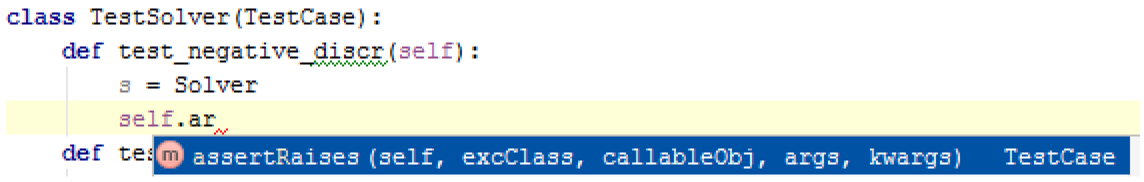
When this statement is complete, it is shown grayed out, which means that import is not used so far.

Next, let's create a test method that will assert throwing an exception, if the discriminant of a quadratic equation is negative. Add the following code to our test case:

```
def test_negative_discr(self):
    s = Solver()
    self.assertRaises(Exception,s.demo,2,1,2)
```



```
class TestSolver(TestCase):
    def test_negative_discr(self):
        s =
```



```
class TestSolver(TestCase):
    def test_negative_discr(self):
        s = Solver
        self.ar
    def te
```

So, the resulting code is:

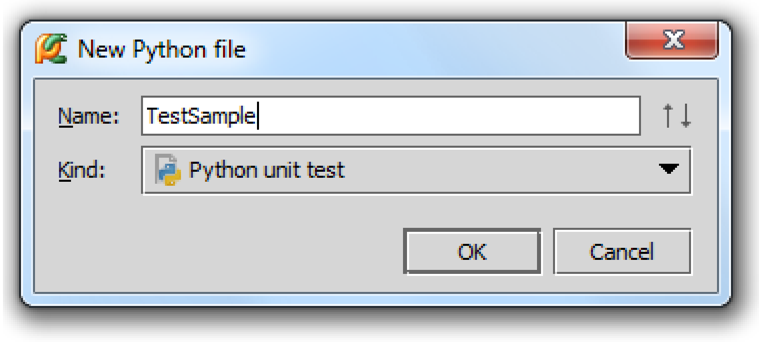
```
from unittest import TestCase
from Solver import Solver


class TestSolver(TestCase):
    def test_negative_discr(self):
        s = Solver()
        self.assertRaises(Exception,s.demo,2,1,2)
    def test_demo(self):
        self.fail()
```

Our test case contains two test methods: test_negative_discr and test_demo. The latter always fails.

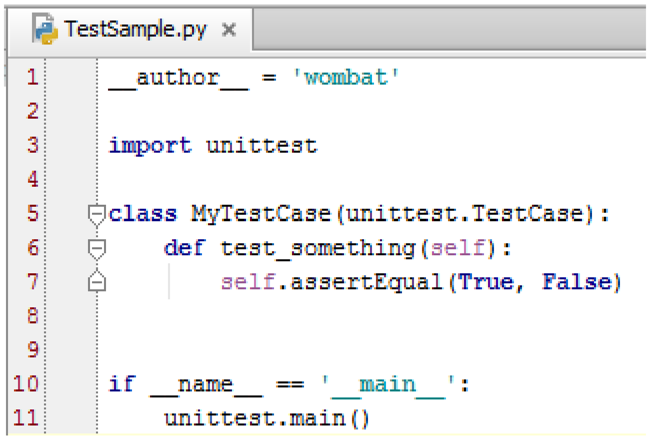
Note that import statement is not gray any more – we've used the imported Solver class in the method test_negative_discr.

Alternative way of creating a test case

Try a different approach. Press Alt+Insert, choose Python file from the pop-up window, then, in the New Python file dialog box, select Python unit test from the Kind menu, and type the name of new test:



PyCharm create a test case with some initial code, and opens it in the editor:



```
__author__ = 'wombat'

import unittest

class MyTestCase(unittest.TestCase):
    def test_something(self):
        self.assertEqual(True, False)


if __name__ == '__main__':
    unittest.main()
```

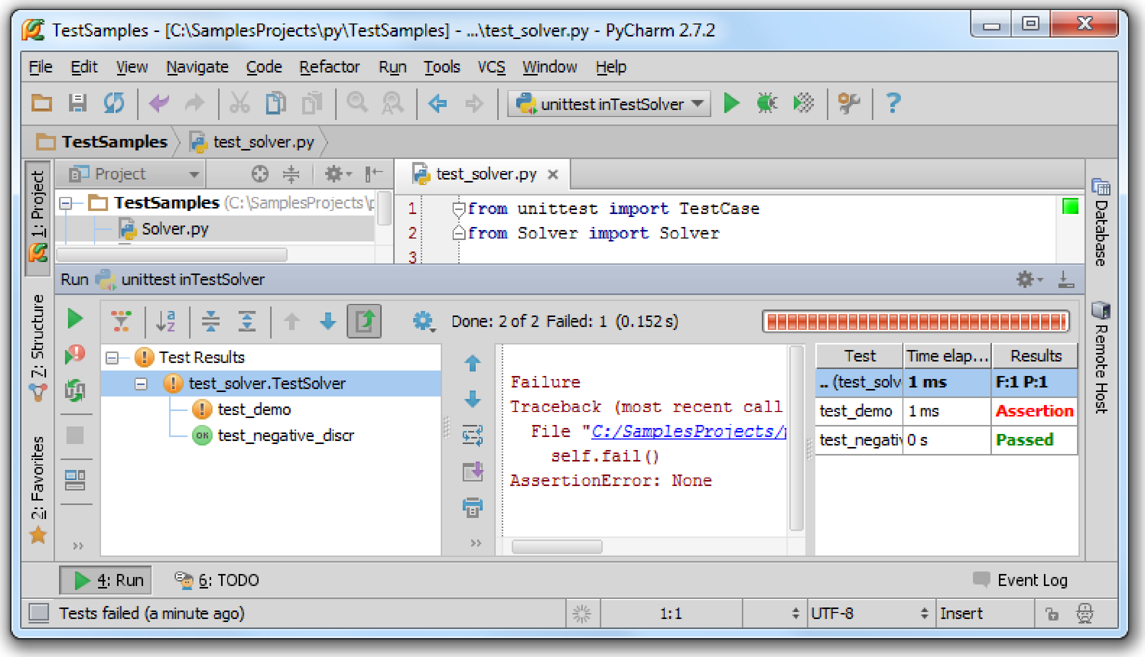
This is again nothing but a stub. So, we'll import the class to be tested, and add a test method. The resulting code is:

```
import unittest

from Solver import Solver



class MyTestCase(unittest.TestCase):

    def test_negative_discr(self):

        s = Solver()
        self.assertRaises(Exception)

    def test_something(self):

        self.assertEqual(True, False)



if __name__ == '__main__':

    unittest.main()
```

Running test

For executing our test case, PyCharm suggests a dedicated temporary run/debug configuration. It is quite ready to be used it “as is”: just press Ctrl+Shift+F10, or right-click somewhere within the class, and choose Run unittests in test_solver on the context menu.

The results are shown in the Run tool window:


TestSamples - [C:\SamplesProjects\py\TestSamples] - ...\test_solver.py - PyCharm 2.7.2
File Edit View Navigate Code Refactor Run Tools VCS Window Help
unittest inTestSolver
TestSamples
test_solver.py
Project
TestSamples (C:\SamplesProjects\p
Solver.py
Run unittest inTestSolver
Database
Remote Host
1: Project
7: Structure
2: Favorites
4: Run
6: TODO
Event Log
1:1
UTF-8
Insert

```
from unittest import TestCase
from Solver import Solver
```

Done: 2 of 2 Failed: 1 (0.152 s)

Test Results
- test_solver.TestSolver
  - test_demo
  - test_negative_discr

```
Failure
Traceback (most recent call
  File "C:/SamplesProjects/
    self.fail()
AssertionError: None
```

Test	Time elap...	Results
.. (test_solv	1 ms	F:1 P:1
test_demo	1 ms	Assertion
test_negati	0 s	Passed

Tests failed (a minute ago)

Your Rating: ☆☆☆☆☆ Results: ★★★½☆ 151 rates