Testing in Python

LINCS Python Workshop

Python’s dynamic nature

- Python is a dynamic and interpreted language
  - Implicit typing
  - (Almost) no compilation step to catch errors
  - Errors show up at runtime

- Solution 1: static analysis
  - Type annotations and mypy
  - pylint, flake8, ...

- Solution 2: automated tests

Automated tests

- *Write code to test code*

- Increase the confidence in the correctness of the code
- Increase productivity
- Detect regressions
- Facilitate refactoring
- Promote modular and easy to test code

doctest

```
class Account:
    """
    A (very unsecure) bank account.
    >>> account = Account()
    >>> account.deposit(10)
    10
    """
```

unittest

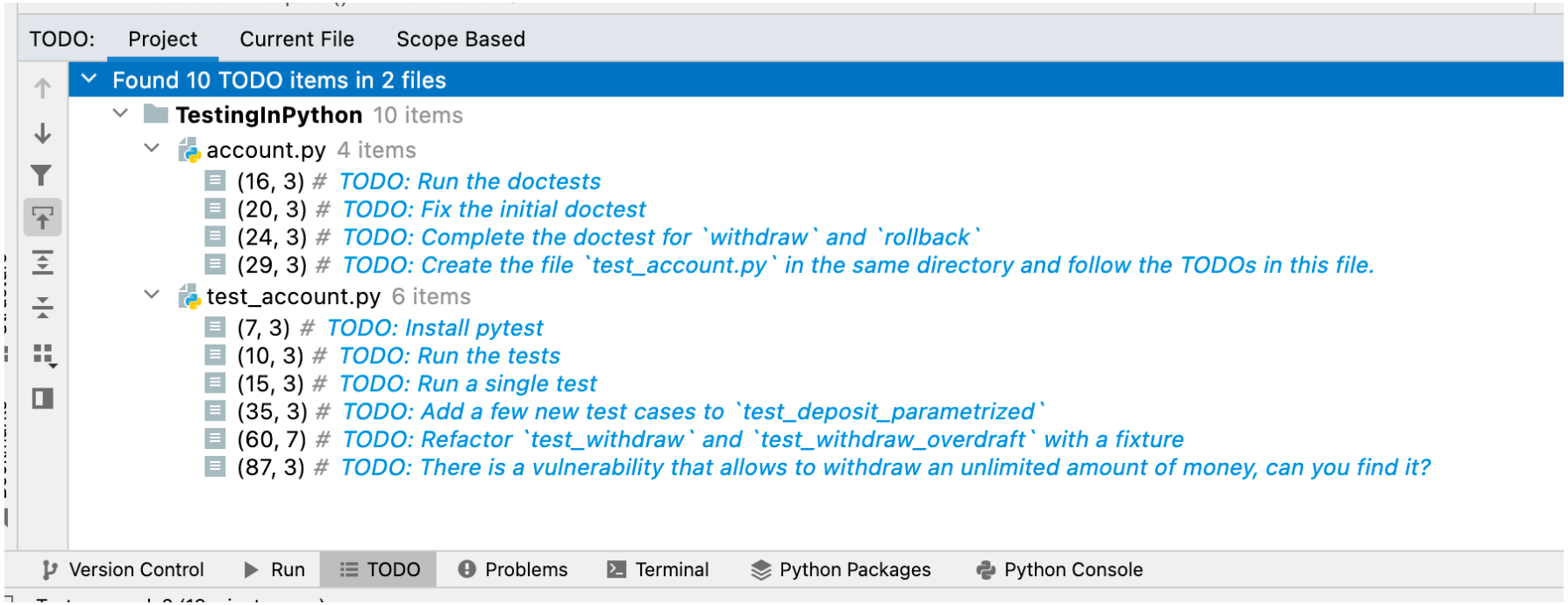
```
class TestAccount(unittest.TestCase):
    def test_deposit(self):
        account = Account()
        balance = account.deposit(10)
        self.assertEqual(balance, 10)
```

pytest

```
def test_deposit():
    account = Account()
    assert account.deposit(10) == 10
```

Today's plan

- doctest & pytest basics
- Can be followed in PyCharm or from the terminal



Let’s get started

1. Go to https://www.lincs.fr/events/testing-in-python-2/
2. Create the two files *account.py* and *test_account.py*
3. Open the progress sheet