

Effective Programming Practices for Economists

Software engineering

Idea of unit testing

Janoś Gabler and Hans-Martin von Gaudecker

The assert statement

```
>>> assert 5 == 6, "Numbers are not equal."
```

```
-----  
AssertionError      Traceback (most recent call last)  
/home/janos/file.py line 1  
----> 1 assert 5 == 6, "Numbers are not equal"
```

```
AssertionError: Numbers are not equal
```

```
>>> assert False, "This always fails."
```

```
-----  
AssertionError      Traceback (most recent call last)  
/home/janos/file.py line 1  
----> 1 assert False, "This always fails"
```

```
AssertionError: This always fails
```

- `assert` raises an error if a condition is not fulfilled
- Often used to check that assumptions about inputs are fulfilled
- Can also be used to test the behavior of functions

Testing a simple function

```
def cobb_douglas(labor, capital, alpha):  
    return labor ** alpha * capital ** (1 - alpha)  
  
assert cobb_douglas(1, 1, 0.5) == 1  
assert cobb_douglas(16, 1, 0.25) == 2  
assert cobb_douglas(1, 16, 0.75) == 2
```

- Test cases can be calculated by hand for simple edge cases
- Sometimes you can get results from other libraries, textbooks, etc.
- If this runs without error, we are confident the function works for other inputs

Testing interfaces, not implementation

```
def combine_keys_and_values(keys, values):  
    return dict(zip(keys, values))
```

```
def combine_keys_and_values_2(keys, values):  
    return {k: v for k, v in zip(keys, values)}
```

```
expected = {"a": 1, "b": 2}  
got1 = combine_keys_and_values(["a", "b"], [1, 2])  
got2 = combine_keys_and_values_2(["a", "b"], [1, 2])
```

- If you define good functions, their interface remains stable
- Can improve the implementation without worrying it will break things

This works for any project!

Unit tests don't work for scientific code. If we knew the result in advance it wouldn't be science

– Anonymous scientist

- Many scientists think they cannot use unit tests in their projects
- But any project can be decomposed into small steps for which you do know what they should do
- Test the steps, not the whole

What are testing frameworks

- In the above examples, python would abort after the first failed test
- Would be nicer to be able to:
 - run all tests and get a report in the end
 - quickly specify subsets of tests to run
 - run the same tests with multiple inputs
- Testing frameworks do just that
- Industry standard is pytest which is basically pytask for tests