# 2810ICT/7810ICT Software Technologies Workshop 9 – Testing

|  |  |
| --- | --- |
| *When* | **Week 9** |
| *Goal* | In this workshop you will learn to use the   * **Pytest** test tool and write some simple python testing programs. * **Coverage** test tool to analyse the coverage of code being   tested. |

**Note: In order to show the difference between tasks, I created different versions (e.g., wordcount\_v1, wordcount\_v2, test\_\*\_v\*.py). But when you do the lab, you don’t need to create those versions and can directly modify wordcount.py and histogram.py.**

**Task 1 – Create a Project in PyCharm**

* Install pytest into your project environment
  + conda env list
  + conda activate your\_project\_env
  + conda install pytest
* Create a new project in pycharm
* Add wordhistogram.py (available on the Week 9 course content page) into the project.
* Now we need to make the **wordhistogram.py** more testable. This can be done by converting the codes into **two python files** with functions:
  + One file is named **wordcount.py**, which contains:
    - One function (**f1**) for reading a file, creating and returning a dictionary.
    - One function (**f2**) for sorting a dictionary reversely based on count of each word and returning the result.
  + One file is named **histogram.py**, which contains:
    - One function (**f3**) for converting a dictionary into a histogram.
* Write a main function and call the methods from the main function. This can be in a separate file or as part of histogram.py
* Solutions:
  + wordcount.py
  + histogram.py
  + main\_run.py

# Task 2 – Unit Test Example 1: Word Histogram

* **Note: Pytest’s standard test discovery rules**
  + **files (modules) must start with test\_\*.py or end with \*\_test.py**
  + **functions must start with test\_\***
  + **class names must start with Test\_\***
* Learn how to use the unit test module in Python.
* Write a program **test\_word.py** to test the above methods with various test cases:
  + For f1, try empty filename and wrong file name
    - Solutions: test\_word.py
  + Fix the bugs in f1
    - Solutions: wordcount\_v1.py test\_word\_v1.py
  + For f2, try input types other than dictionary, empty dictionary
    - Solutions: test\_word\_v2.py
  + Fix the bugs in f2
    - Solutions:
      * wordcount\_v2.py
      * test\_word\_v3.py
  + How do you test f3? Write your test code.
    - Solutions:
      * test\_word\_v4.py
      * fix the bugs and retest
        + histogram\_v1.py
        + test\_word\_v5.py

# Task 3 – Unit Test Example 2: Recursion

* Two versions to calculate factorial (iterative and recursion)

def iterative\_factorial(n): result = 1

for i in range(2,n+1): result \*= i

return result def factorial(n):

if n == 1: return 1

else:

return n \* factorial(n-1)

* Create test cases for the recursion version, and store them in **test\_factorial.py**
  + Solutions:
    - recursion.py
    - test\_recursion.py
* Fix the functions so they deal with invalid input
  + Solutions:
    - Recursion\_v1.py
    - Test\_recursion\_v1.py

# Task 4 – Coverage test

* Use conda to install coverage test tool.
  + conda env list
  + conda activate your\_project\_env
  + conda list coverage
  + conda install coverage -c conda-forge
* Run the coverage tool on the **test\_word.py** code and produce a report
  + Solutions:
    - Open your terminal and activate your project python env, **cd** to your project path where the test\_word\_v5 is located, then run the following cmds
      * **coverage run -m pytest** test\_word\_v5.py
      * **coverage html**
* Copy the report in index.html (only) into the Word document.
  + Note: you will find the **index.html** in the folder **htmlcov** in your project folder

# Task 5 – Improve coverage test

* Analyse the report and find untested code. Figure out the reason.
* Write new unit test code so untested code can be covered
  + Solutions:
    - **coverage run -m pytest** test\_word\_v6.py
    - **coverage html**

Note: you are expected to see 100% coverage for histogram\_v1.py and wordcount\_v2.py.

It does not matter for test\_\*.py without 100% coverage, as the test\_\*.py files are designed to test codes.