Functional Testing
Software Engineering for Scientists

Functional tests run full programs to ensure the correct result, including proper error handling, considering different inputs and behaviors. Similar to **unit tests**, functional tests help prevent bug fixes and feature updates from breaking other parts of the project. Functional tests can also be helpful when recreating and fixing bugs.

Functional Tests with Stupid Simple baSh Testing (https://github.com/ryanlayer/ssshtest)

`ssshtest` provides a simple framework for testing command-line programs in a bash script. To use `ssshtest`, create a test script (e.g., `test_calc.sh`) that start with the following two lines:

`test_calc.sh`

```
test -e ssshtest || wget -qhttps://raw.githubusercontent.com/ryanlayer/ssshtest/master/ssshtest
. ssshtest
```

The first line tests for the presence of the `ssshtest` library, and downloads a copy if it is not. The next line incorporates the variables and functions defined in the library into the current runtime environment.

Next, come the tests, each of which has two stages.

1. The `run` stage executes the program with specific input parameters. The syntax of the run command is:
   `run <test name> <program> <argument 1> <argument 2> <...>`
   All of the output and exit codes from the program and its arguments are stored so that they can be tested in the next stage. These values will remain available for testing until the next `run` command.
2. The `assert` stage tests various aspects of the previous `run` including output and the return value. There are eight different assertions to choose from, and each `run` can have any number of `asserts`

`assert_equal <obs> <exp>`	Assert that 2 things are equal (string comparison)
`assert_stdout`	Assert that stdout is not empty
`assert_in_stdout <pattern>`	Assert that stdout contains this text.
`assert_no_stdout`	Assert that stdout is empty
`assert_stderr`	Assert that stderr is not empty
`assert_in_stderr <pattern>`	Assert that stderr contains this text.
`assert_no_stderr`	Assert that stderr is empty
`assert_exit_code <exit code>`	Assert that the program exited with a particular code

Together a test will look like:

```
run basic_add python calc.py add 1 1
assert_in_stdout 2
assert_exit_code 0
```

Where the output of `python calc.py add 1 1` is captured and then the output is tested to see if it contains 2, and the exit code is tested to see if it equals 0.
Consider the following python files

`math_lib.py` `calc.py`

```python
def div(a, b):
    return a / b

def add(a, b):
    return a + b
```

```python
import math_lib, sys

op = sys.argv[1]
a = int(sys.argv[2])
b = int(sys.argv[3])

if op == 'add':
    print(math_lib.add(a,b))
elif op == 'div':
    print(math_lib.div(a,b))
```

And testts

```
test -e ssshtest || wget -q https://raw.githubusercontent.com/ryanlayer/ssshtest/master/ssshtest
. ssshtest

run basic_add python calc.py add 1 1
assert_in_stdout 2
assert_exit_code 0

run basic_div python calc.py div 1 0
assert_exit_code 0
```

Gives

```
$ bash test_calc.sh

basic_add ran in 0 sec with 0/1 lines to STDERR/OUT
 PASS STDOUT CONTAINS "2" (LINE 6)
 PASS EXIT CODE (LINE 7)

basic_div ran in 0 sec with 6/0 lines to STDERR/OUT
 FAIL EXIT CODE (LINE 11)
-->   expected EX_OK, observed Unknown code: 1
Traceback (most recent call last):
  File "calc.py", line 11, in <module>
    print(math_lib.div(a,b))
  File "/Users/rl/Box
Sync/Research/teaching/swe4s/lectures/functional_testing/math_lib.py", line 2, in div
    return a / b
ZeroDivisionError: integer division or modulo by zero


sshtest v0.1.5

2         Tests
1         Failures
2         Successes
```