**Methods and Tools in SW Development**

**Homework 5**

Joshua Bean (jab1896)

**Function Name:** openFile

**Number of tests:** 3

**Items to test:**

* Test with filename of existing file
* Test with file that does not exist
* Test with input other than a string

**Inputs used for testing:**

* “testing.txt”
* “test.txt”
* 123

**Corrections made:**

In paragraph form, discuss what changes were made to the function and discuss why they were made. Reference specific tests from the “items to test” section to list what failed previously and how these changes prevented failures.

* The possible failures are caused by passing something other than a string into ‘filename’ or by passing in a filename that does not exist. The first thing that I changed was to check the type of the input ‘filename.’ If the input is not a string, like in ‘test\_openFile\_int,’ then it will print an error message and return. The other change I made was to put the open(filename) command inside a try/except block. If ‘filename’ references a file that does not exist like in ‘test\_openFile\_wrong\_name,’ it will just print an error message.

**Tests passed:** 3

**Tests failed:** 0

**Function Name:** numbers

**Number of tests:** 4

**Items to test:**

* Two integers
* Two floats
* Divide by zero
* Wrong datatypes

**Inputs used for testing:**

* 8, 4
* 2.2, 1.1
* 2, 0
* ‘abc’, [1, 2, 3]

**Corrections made:**

In paragraph form, discuss what changes were made to the function and discuss why they were made. Reference specific tests from the “items to test” section to list what failed previously and how these changes prevented failures.

* The possible errors are datatype errors or division by zero. Accordingly, the tests that failed initially were ‘test\_numbers\_datatype’ and ‘test\_numbers\_zero.’ The first change I made was to check if num2 is 0. If it is, print an error message and return. Then I put the division inside a try/except block to catch any datatype errors such as the ‘test\_numbers\_datatype’ test which attempts to divide a string by a list.

**Tests passed:** 4

**Tests failed:** 0

**Function Name:** dist

**Number of tests:** <number of tests>

**Items to test:**

* Item 1
* Item 2
* Item 3

**Inputs used for testing:**

* Input 1, input 2
* …

**Corrections made:**

In paragraph form, discuss what changes were made to the function and discuss why they were made. Reference specific tests from the “items to test” section to list what failed previously and how these changes prevented failures.

*(optional) Failed tests:*

List what failed and why it was meant to fail

**Tests passed:** <number>

**Tests failed:** <number>

**Function Name:** isPalendrome

**Number of tests:** <number of tests>

**Items to test:**

* Item 1
* Item 2
* Item 3

**Inputs used for testing:**

* Input 1, input 2
* …

**Corrections made:**

In paragraph form, discuss what changes were made to the function and discuss why they were made. Reference specific tests from the “items to test” section to list what failed previously and how these changes prevented failures.

*(optional) Failed tests:*

List what failed and why it was meant to fail

**Tests passed:** <number>

**Tests failed:** <number>

**Function Name:** divide

**Number of tests:** <number of tests>

**Items to test:**

* Item 1
* Item 2
* Item 3

**Inputs used for testing:**

* Input 1, input 2
* …

**Corrections made:**

In paragraph form, discuss what changes were made to the function and discuss why they were made. Reference specific tests from the “items to test” section to list what failed previously and how these changes prevented failures.

*(optional) Failed tests:*

List what failed and why it was meant to fail

**Tests passed:** <number>

**Tests failed:** <number>

**Function Name:** sq

**Number of tests:** <number of tests>

**Items to test:**

* Item 1
* Item 2
* Item 3

**Inputs used for testing:**

* Input 1, input 2
* …

**Corrections made:**

In paragraph form, discuss what changes were made to the function and discuss why they were made. Reference specific tests from the “items to test” section to list what failed previously and how these changes prevented failures.

*(optional) Failed tests:*

List what failed and why it was meant to fail

**Tests passed:** <number>

**Tests failed:** <number>

**Function Name:** greetUser

**Number of tests:** 3

**Items to test:**

* Normal string input
* Other datatypes
* Strings with numbers

**Inputs used for testing:**

* “Josh”, “A”, “Bean”
* 1, [2,3], 4.5
* “J0sh”, “A”, “22”

**Corrections made:**

In paragraph form, discuss what changes were made to the function and discuss why they were made. Reference specific tests from the “items to test” section to list what failed previously and how these changes prevented failures.

* I updated the greetUser function to check if the inputs are type string and then if the inputs have letters. If either is true, the function prints “Invalid input” and returns. The test ‘test\_greetUser\_datatype’ tests non-string inputs, and the test ‘test\_greetUser\_numbers’ tests input with numbers in the string.

**Tests passed:** 3

**Tests failed:** 0

**Function Name:** displayItem

**Number of tests:** <number of tests>

**Items to test:**

* Item 1
* Item 2
* Item 3

**Inputs used for testing:**

* Input 1, input 2
* …

**Corrections made:**

In paragraph form, discuss what changes were made to the function and discuss why they were made. Reference specific tests from the “items to test” section to list what failed previously and how these changes prevented failures.

*(optional) Failed tests:*

List what failed and why it was meant to fail

**Tests passed:** <number>

**Tests failed:** <number>