Introduction to Software Engineering - ISAD1000
Final Assignment
Student Name - Yashi Vaidehi Domun
Student ID - 20779173
Practical Class - Tuesday/8.30am

Introduction

In this assignment, multiple concepts such as software testing, modularity,version control and ethics were applied in the scenario of a small software project. While creating the scenario of category one,for each functionality,the user is allowed to to input a string via keyboard and receive the results on the terminal. As for category two,input is taken from the user via keyboard and the results are both printed on the terminal and written to text file(each result is seperated on different lines). As modularity is included in this mini software project, a checklist and the refactor of the code was implemented to check whether the assignment follow all basic modularity guidelines as applied from the lectures. For software testing, test designs such as Black box testing which include equivalance partitioning and boundary value analysis are used to test the two categories implemented.White box testing another test design in software testing was included.

Module Description
Category 1:
1. Convert a string which is prompted from the user via keyboard to lowercase or uppecase depending on the choice of the user.
   - menu() used to print different options avalaible
   - lowerconversion() convert a given string to lowercase
   - upperconversion() convert a given string to uppercase
   - menuinput() used to get the choice of the user on whether to convert the string that is also prompted by the user into lowercase or uppercase.

2. To identify whether numeric values are present in the string entered by the user via keyboard.
   - numbersfound() used to check character by character in the string given by the user if it contain any numeric values with the use of .isdigit()

3. To identify whether the given string from the user via keyboard is a valid number.
   - checknumbers() used to check character by character whether the string is a valid number with the use of .isnumeric()

4. To remove any numeric values in the given string from the user via keyboard and then to convert it into either lowercase or uppercase.
   - lists() used to print the options avalaible for the user.
   - removingnumbers() used to remove any numeric character from the user input string using regex.
   - menu() used to get the choice of the user on whether to convert the string to lowercase or uppercase

Category 2:
It is a program that contain a class with different functions that convert a user input number to hours, minutes or seconds.
In class category2-
1. convertHours() used to convert a string in minutes
2. convertminutes() used to convert a number to seconds
3. convert() used to convert a number to seconds
4. convertm() used to convert a number to hours
5. main() contains the function calls where a user has to enter numbers for the program to convert them and then the results are written in textfile called category2.txt.

Modularity
1. User guide to run production code for category one
   - Part a, run the program with python3 q1.py, on the terminal a sort of menu will appear where the user gets to choose whether he/she wants to convert a string into lowercase or uppercase. The user can enter his/her choice via keyboard.

   - Part b, run the program with python3 q2.py, on the terminal, the user will be asked to input a string.

   - Part c, run the program with python3 q3.py, the user will be prompted to input a string.

   - Part d, run the program with python3 q4.py.

2. User guide to run production code for category two
   - Run the program with python3 category.py, the user will be prompted to input multiple values for its conversion to either hours, minutes or seconds.

Checklist
1. Each function performs the task given well when called?
2. Does code follow modularity guidelines?
3. Will the code be easy to implement in the future depending on future adaptation?
4. Is the code implemented so that no global variable is used?
5. Is the code implemented so that it is free from control flags?
6. Is each operator working with the correct datatype?
7. Are the correct datatypes passed to the functions/methods in the code?
8. Are there any hidden dependencies in the code?

Review
1. No global variable was used throughout the code.
2. No control flags are present in the code.
3. No form of redundancy found when checking for hidden dependcy in category 1 and 2.
4. For category 2, when writing the results in the textfile, None is written instead of the results shown on the terminal.

Refactoring of the code after review and checklist
1. To implement the code for it to write the result in the textfile, assigning the function call to variables is deleted.

Black Box test cases

1. Category 1

Equivalence partitioning is used in all parts of category one as the input data can be easily partitioned into valid or invalid results as all partitions bhave in the same way. Thus easy to test it through the input of the user.

- Part a, equivalence partitioning is used to test the function that converts a string to either lowercase or uppercase.

Category	Test Data	Expected result
String to uppercase	‘Yashi’	‘YASHI’
String to lowercase	‘YAShi’	‘yashi’
Input string by user is not a letter	‘534’	error

- Part b, equivalence partitioning is used to test the function numbersfound() on whether there are numbers in the string entered by the user.

Category	Test Data	Expected result
Numbers present in string	‘qwer3ty’	‘True’
Only numbers present in string	‘2345’	‘True’
No numbers present in string	‘qwerty’	‘False’

- Part c, equivalence partitioning is used to test the function checknumbers() on whether a valid number string in entered by the user.

Category	Test Data	Expected result
Contains text in string	‘qwer3ty’	‘False’
Valid number	‘2345’	‘True’
No numbers present in string	‘qwerty’	‘False’

- Part d, equivalence partitioning is used to test the function removenumbers().

Category	Test Data	Expected result
yashi	‘yas5hi’	‘5 is removed’
YASHI	‘YASH67HI’	’67 is removed’
error	‘error’	‘no number found to be removed’

2. Category 2

Boundary value analysis is used in part c of category 2 asthis testing technique test the program using boundary values found in a range. This technique halped in testing category 2 as conversion of time has boundaries. For example, time cannot be negative or that in one hour there can only have 60 minutes.

Boundary	Test Data	Expected result
1-24	-1 0	“error message” “0”
1-60	1 0	“60” “0”
1-60	2 20	“120” “1200”

White Box test cases

White box testing was used as it can reveal structural errorsas well as hidden mistakes within specific components. Unit tests was used because they are tests that are created as part of the testing code of the production code that ensure that each modules or function works as planned.

1. Category 1, Part a

Path	Test Data	Expected result
Lower conversion of the input string	‘YasHI Domun’	‘yashi domun’
Lower conversion of the input string	‘vaIDEHi’	‘vaidehi’

2. Category 1, Part a

Path	Test Data	Expected result
upper conversion of the input string	‘YasHI Domun’	‘YASHI DOMUN’
upper conversion of the input string	‘vaIDEHi’	‘VAIDEHI’

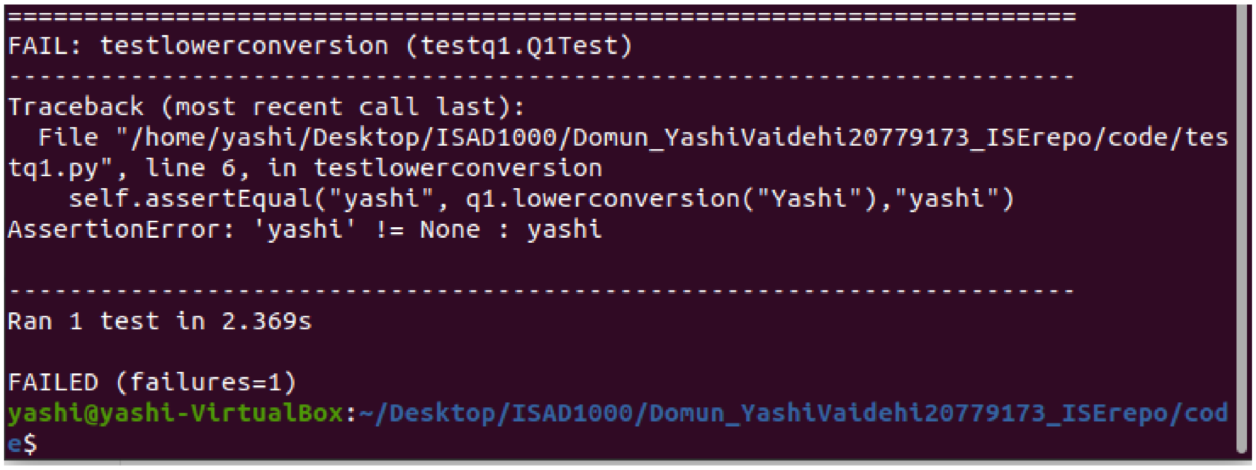
3. Category 1, Part d

Path	Test Data	Expected result
Removing numbers in the input string	‘yashi2022’	‘yashi ’
Removing numbers in the input string	‘LOTR2’	‘LOTR’

Test implementation and Results
Unit test framework was implemented in all the test designs that is black box and white box testing.
The table below shows the what test has been implented for each category:

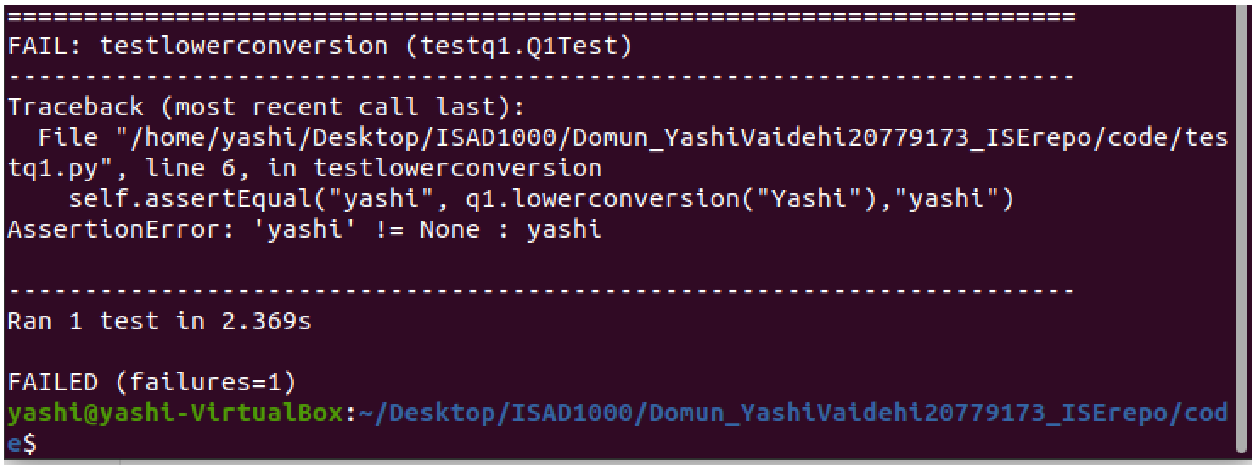
Module name	BB test design(EP)	BB test design (BVA)	WB test design	EP test code (implemented/run)	BVA testcode (implemented/run)
Lowerconversion()	Done	Not done	Done	implemented	
Upperconversion()	Done	Not done	Done	Implemented	
Numbersfound()	Done	Not done	Not done	Implemented	
Checknumbers()	Done	Not done	Not done	Implemented	
Removingnumbers()	Done	Not done	Done	Run with unit test	
convertHours()	Not done	Done	Not done		Implemented
Convertminutes()	Not done	Done	Not done		implemented

For white box testing, all test design were run succesfully using unit test.
However, some tests failed -
 1. For lowerconversion module, the test run without errors when compiled with python3 testq1.py but when using unit test to compile it, it crashes with errors. To run with unit test, python3 -m unittest testq1.py command is used.
    To correct these errors, many changes were made to the module.



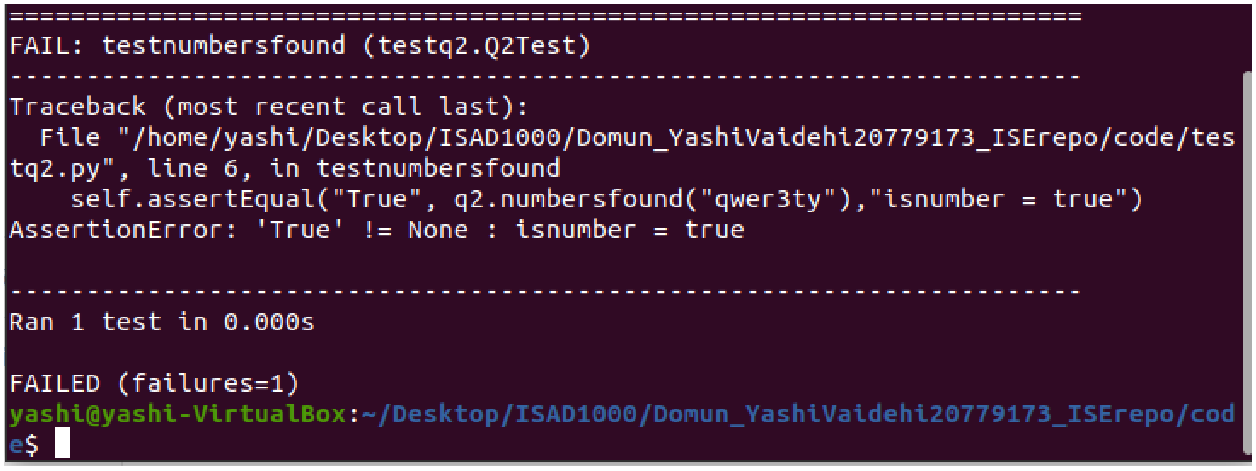
```
======================================================================
FAIL: testlowerconversion (testq1.Q1Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/yashi/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/code/tes
tq1.py", line 6, in testlowerconversion
    self.assertEqual("yashi", q1.lowerconversion("Yashi"),"yashi")
AssertionError: 'yashi' != None : yashi

----------------------------------------------------------------------
Ran 1 test in 2.369s

FAILED (failures=1)
yashi@yashi-VirtualBox:~/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/cod
e$
```

2. For upperconversion module, the test run without errors when compiled with python3 testq1.py but when using unit test to compile it, it crashes with errors. To run with unit test, python3 -m unittest testq1.py command is used.
    To correct these errors, many changes were made to the module.



```
======================================================================
FAIL: testlowerconversion (testq1.Q1Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/yashi/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/code/tes
tq1.py", line 6, in testlowerconversion
    self.assertEqual("yashi", q1.lowerconversion("Yashi"),"yashi")
AssertionError: 'yashi' != None : yashi

----------------------------------------------------------------------
Ran 1 test in 2.369s

FAILED (failures=1)
yashi@yashi-VirtualBox:~/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/cod
e$
```

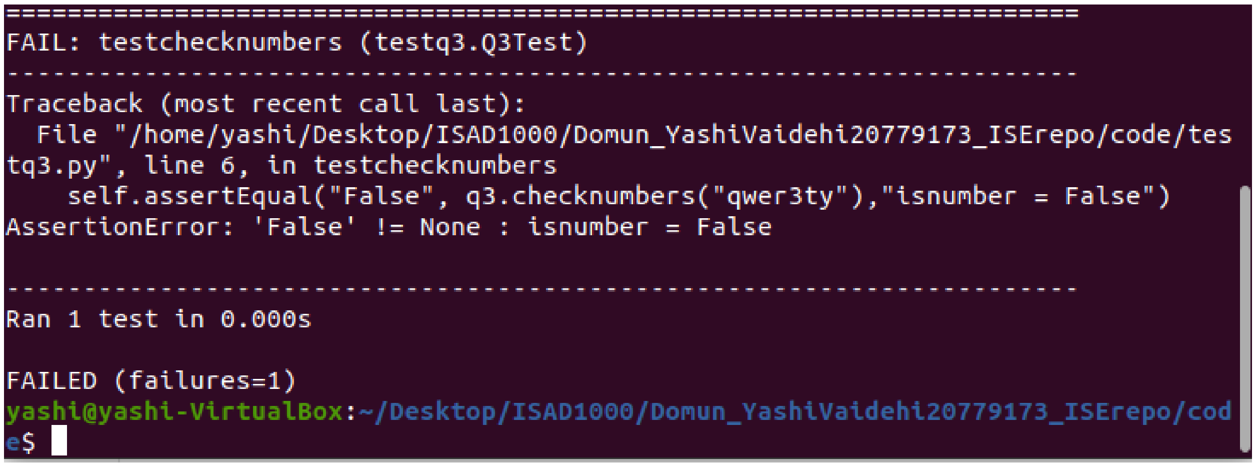
3. For numbersfound module, the test run without errors when compiled with python3 testq2.py but when using unit test to compile it, it crashes with errors. To run with unit test, python3 -m unittest testq2.py command is used.
    To correct these errors, many changes were made to the module.



```
======================================================================
FAIL: testnumbersfound (testq2.Q2Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/yashi/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/code/tes
tq2.py", line 6, in testnumbersfound
    self.assertEqual("True", q2.numbersfound("qwer3ty"),"isnumber = true")
AssertionError: 'True' != None : isnumber = true

----------------------------------------------------------------------
Ran 1 test in 0.000s

FAILED (failures=1)
yashi@yashi-VirtualBox:~/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/cod
e$ 
```

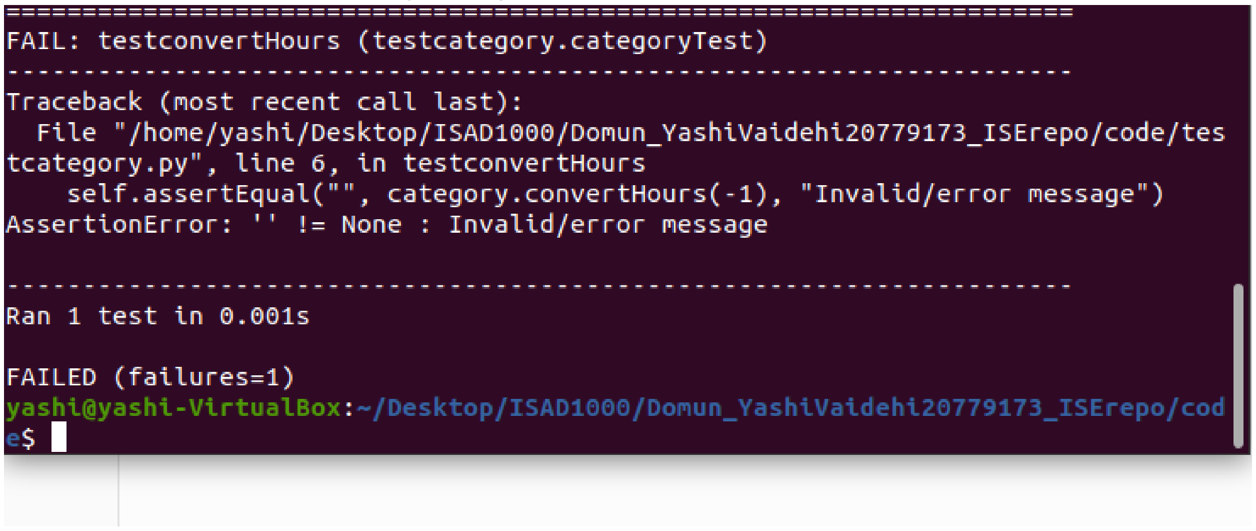
4. For checknumbers module, the test run without errors when compiled with python3 testq3.py but when using unit test to compile it, it crashes with errors. To run with unit test, python3 -m unittest testq3.py command is used.

To correct these errors, many changes were made to the module.



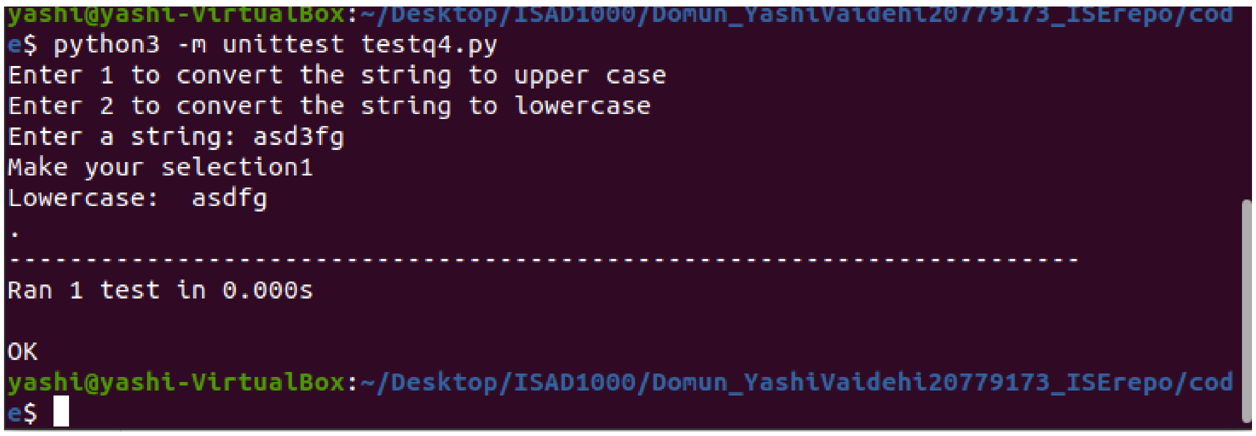
```
======================================================================
FAIL: testchecknumbers (testq3.Q3Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/yashi/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/code/tes
tq3.py", line 6, in testchecknumbers
    self.assertEqual("False", q3.checknumbers("qwer3ty"),"isnumber = False")
AssertionError: 'False' != None : isnumber = False

----------------------------------------------------------------------
Ran 1 test in 0.000s

FAILED (failures=1)
yashi@yashi-VirtualBox:~/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/cod
e$ 
```

5. For convertHours and convertminutes modules, the test run without errors when compiled with python3 testcategory.py but when using unit test to compile it, it crashes with errors. To run with unit test, python3 -m unittest testcategory.py command is used.
To correct these errors, many changes were made to the module.

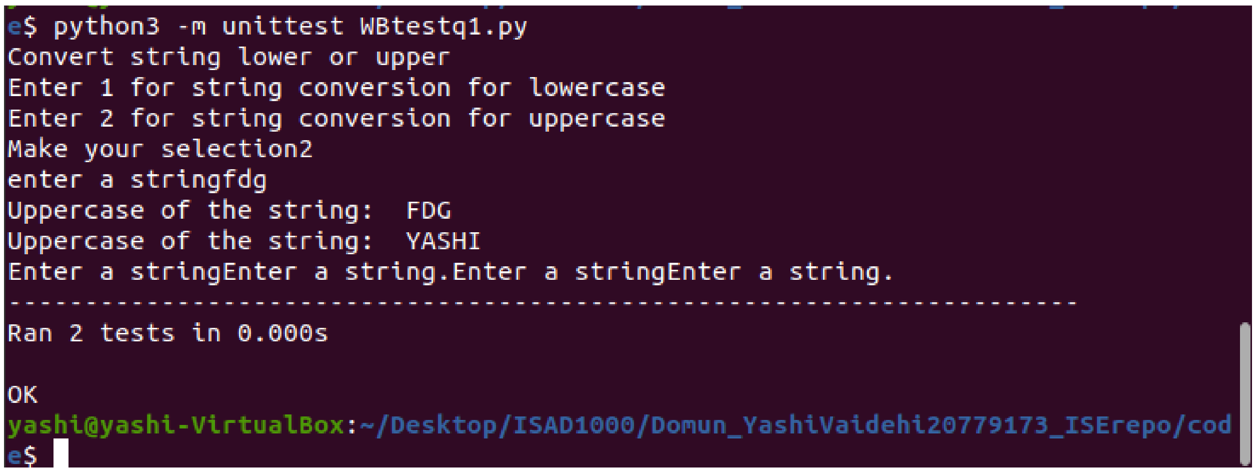


```
======================================================================
FAIL: testconvertHours (testcategory.categoryTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/yashi/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/code/tes
tcategory.py", line 6, in testconvertHours
    self.assertEqual("", category.convertHours(-1), "Invalid/error message")
AssertionError: '' != None : Invalid/error message

----------------------------------------------------------------------
Ran 1 test in 0.001s

FAILED (failures=1)
yashi@yashi-VirtualBox:~/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/cod
e$
```

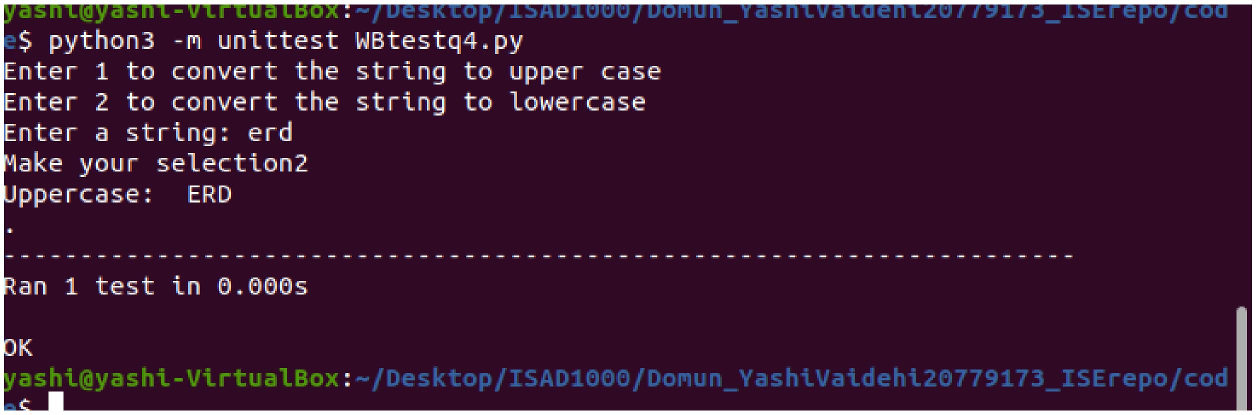
6. Black box testing for category 1 part d worked as well as White box testing.



```
yashi@yashi-VirtualBox:~/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/cod
e$ python3 -m unittest testq4.py
Enter 1 to convert the string to upper case
Enter 2 to convert the string to lowercase
Enter a string: asd3fg
Make your selection1
Lowercase:  asdfg
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
yashi@yashi-VirtualBox:~/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/cod
e$
```



```
e$ python3 -m unittest WBtestq1.py
Convert string lower or upper
Enter 1 for string conversion for lowercase
Enter 2 for string conversion for uppercase
Make your selection2
enter a stringfdg
Uppercase of the string:  FDG
Uppercase of the string:  YASHI
Enter a stringEnter a string.Enter a stringEnter a string.
----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK
yashi@yashi-VirtualBox:~/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/cod
e$
```



```
yashi@yashi-VirtualBox:~/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/cod
e$ python3 -m unittest WBtestq4.py
Enter 1 to convert the string to upper case
Enter 2 to convert the string to lowercase
Enter a string: erd
Make your selection2
Uppercase:  ERD
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
yashi@yashi-VirtualBox:~/Desktop/ISAD1000/Domun_YashiVaidehi20779173_ISErepo/cod
e$ 
```

Ethics and Professionalism

1. For the mini software project/scenario created for this assignment, if functionalities that were asked to implement are missing. It will cause harm to specifically to guidelines 1,2,3 and 6. Guideline one which represents that software engineers should act consistently with public interest may seem violated as a functionality that was asked in the project has to important hence not implementing will cause harm to public interest and failing to mention that this particular functionality was failed to be implemented goes against guideline three (Product) and guideline four (Judgement). The integrity and reputation of either the product or the profession might be targeted as an important part of the project was either forgotten or contains errors which results in failed usage of the functionality.

2. In order to avoid ethical and professional issues discussed in the first part of the question, one might take into consideration of the ACS guidelines such as Honesty and Professionalism. By following the guideline honesty, one might report or write a policy on how certain functionalities are not working in the project given. Some action will be taken but not because of a violated ethic and profession guideline. An extension to do the assignment may be asked or the failed functionality can be mentioned in both the report and future works to showcase how one might solve it.
Guideline Professionalism might be another alternative for this assignment as by coming forward and being honest about the failed functionalities, it can count as enhancing the integrity of ACS among your peers/ colleagues.

Discussion & Future works

Black box testing with either equivalence partitioning and boundary value analysis had errors and failures. This could have been implemented further if the concept of both of those testing was grasped enough for it to be implemented in different types of coding environment.