

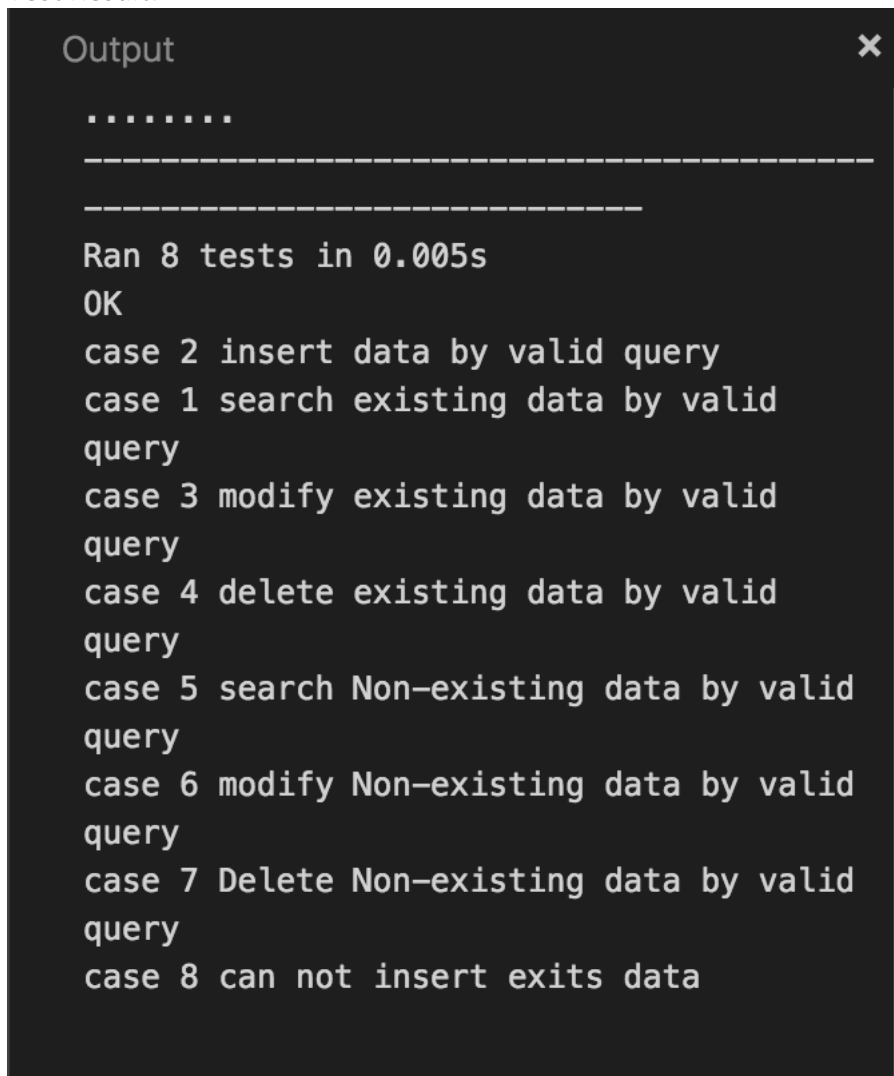
SOFTWARE TESTING METHODOLO HW 7 TinyDB

Evaluation Test

In this assignment I will test tinyDB driver some operations.

- case 1 search existing data by valid query
- case 2 insert data by valid query
- case 3 modify existing data by valid query
- case 4 delete existing data by valid query
- case 5 search Non-existing data by valid query
- case 6 modify Non-existing data by valid query
- case 7 Delete Non-existing data by valid query
- case 8 can not insert exists data

Test Result:



```
.....
-----
-----
Ran 8 tests in 0.005s
OK
case 2 insert data by valid query
case 1 search existing data by valid
query
case 3 modify existing data by valid
query
case 4 delete existing data by valid
query
case 5 search Non-existing data by valid
query
case 6 modify Non-existing data by valid
query
case 7 Delete Non-existing data by valid
query
case 8 can not insert exists data
```

Codes:

```
1 from tinydb import TinyDB, where
2 import unittest
3
4 def condb():
5     db = TinyDB('db.json')
6     return db
7
8 class Test_001_Insert_by_valid_query_Function(unittest.TestCase):
9
10     def setUp(self):
11         self.db = TinyDB('db.json')
12
13     def tearDown(self):
14         self.db.purge()
15         self.db.all()
16
17     def test_simple_insert_valid_exist(self):
18         print("case 2 insert data by valid query")
19         self.db.insert({'Name': 'cshu', 'Email': 'cshu1@kent.edu', 'int' : 1, 'char':1})
20         result=self.db.search(where('Name') == 'cshu')
21         self.assertEqual(result,[{'Name': 'cshu', 'Email': 'cshu1@kent.edu', 'int' : 1,
22
23 class Test_002_Search_existing_data_by_valid_query_Function(unittest.TestCase):
24     def setUp(self):
25         self.db = TinyDB('db.json')
26
27     def tearDown(self):
28         self.db.purge()
29         self.db.all()
30
31     def test_simple_search_valid_exist(self):
32         print("case 1 search existing data by valid query")
33         self.db.insert({'Name': 'cshu', 'Email': 'cshu1@kent.edu', 'int' : 1, 'char':1})
34         result=self.db.search(where('Name') == 'cshu')
35
36         self.assertEqual(result,[{'Name': 'cshu', 'Email': 'cshu1@kent.edu', 'int' : 1,
37
38 class Test_003_Modify_existing_data_by_valid_query_Function(unittest.TestCase):
39     def setUp(self):
40         self.db = TinyDB('db.json')
41
42     def tearDown(self):
43         self.db.purge()
44         self.db.all()
45
46     def test_simple_modify_valid_exist(self):
47         print("case 3 modify existing data by valid query")
48         self.db.insert({'Name': 'first', 'Email': 'first@kent.edu', 'int' : 1, 'char':1})
49         self.db.update({'int': 10}, where('Name') == 'first')
50
51         result=self.db.search(where('Name') == 'first')
```

```

class Test_004_Delete_existing_data_by_valid_query_Function(unittest.TestCase):
    def setUp(self):
        self.db = TinyDB('db.json')

    def tearDown(self):
        self.db.purge()
        self.db.all()

    def test_simple_delete_valid_exist(self):
        print("case 4 delete existing data by valid query")
        self.db.insert({'Name': 'cshu', 'Email': 'cshu1@kent.edu', 'int' : 1, 'char':1})
        self.db.remove(where('Name') == 'cshu')
        result=self.db.search(where('Name') == 'cshu')

        self.assertEqual(result,[])

class Test_005_Search_Not_existing_data_by_valid_query_Function(unittest.TestCase):
    def setUp(self):
        self.db = TinyDB('db.json')

    def tearDown(self):
        self.db.purge()
        self.db.all()

    def test_simple_search_not_exist(self):
        print("case 5 search Non-existing data by valid query")
        result=self.db.search(where('Name') == 'Ted')
        self.assertEqual(result,[])

class Test_006_Modify_Not_existing_data_by_valid_query_Function(unittest.TestCase):
    def setUp(self):
        self.db = TinyDB('db.json')

    def tearDown(self):
        self.db.purge()
        self.db.all()

    def test_simple_modify_not_exist(self):
        print("case 6 modify Non-existing data by valid query")
        result=self.db.update({'int': 10}, where('Name') == 'Ted')
        self.assertEqual(result,None)

class Test_007_Delete_Not_existing_data_by_valid_query_Function(unittest.TestCase):
    def setUp(self):
        self.db = TinyDB('db.json')

    def tearDown(self):
        self.db.purge()
        self.db.all()

    def test_simple_delete_not_exist(self):
        print("case 7 Delete Non-existing data by valid query")

```

```
109
110 class Test_008_Insert_exits_data_Function(unittest.TestCase):
111     def setUp(self):
112         self.db = TinyDB('db.json')
113
114     def tearDown(self):
115         self.db.purge()
116         self.db.all()
117
118     def test_simple_insert_by_query(self):
119         print("case 8 can not insert exits data")
120         self.db.insert({'Name': 'Yingyu Wu', 'Email': 'ywu23@kent.edu', 'int' : 1, 'char'
121         self.db.insert({'Name': 'Yingyu Wu', 'Email': 'ywu23@kent.edu', 'int' : 1, 'char'
122         result_array = self.db.search(where('Name') == 'Yingyu Wu')
123         num = len(result_array)
124         # print (result_array)
125         #print("search one key,get %d result" %(num))
126         self.assertEqual(2,num)
127
128 if __name__ == "__main__":
129     unittest.main()
```