

Data management

UserManager class and testing

UserManager

- we will create a UserManager class to organize our DB operations

```
# UserManager.py
import pymongo

class UserManager:

    def __init__(self, conn_str:str):
        '''connect to the DB and set self.col'''

    def reset(self):
        '''drop the collection'''

    # [...]
```

UserManager ctd.

- we want create_user and read_user to work with strings rather than ObjectIds.
- This will make our lives easier later

```
class UserManager:
    # [...]
    def create_user(self, user_dict:dict) -> str | None:
        '''insert and return _id'''

    def read_user(self, user_id:str) -> dict:
        '''read and return user'''
```

read many UserManager

- add a read_users method to your UserManager
- The parameter 'query' is optional, the default value is an empty list, which will return all

```
class UserManager:
    # [...]
    def read_users(self, query={}) -> list:
        '''find with query and return list'''
```

update and delete

```
class UserManager:
    # [...]
    def update_user(self, user_id:str, updates:dict) -> int
        '''find by id and apply updates with $set
```

Working with UserManager

```
um = UserManager("mongodb://localhost:27017/")
um.reset()
_id = um.create_user({'username': 'joe', 'password': 'schmoe'})
u = um.read_user(_id)
print(u)
```

Testing - overview

- Each test should be independent

```
# file: UserManager_tests.py

import UserManager
um = UserManager("mongodb://localhost:27017/")

class TestUserManager(unittest.TestCase):

    def test_basic(self):
        ''' reset, create user, read user'''

if __name__ == '__main__':
    unittest.main()
```

Unittest assertions

- You can print to manually inspect results
- You should use assertions to automate testing

```
def test_basic(self):  
    ''' reset, create user, read user'''  
  
    # u = read_user(..)  
    u = None  
    self.assertIsNotNone( u )
```


Adding tests

- we should test for successful and unsuccessful cases
- what are some other tests we should create?

```
def test_notfound(self):  
    ''' try to read user with invalid id, assert no user'''  
    u = read_user('nonsense')  
    self.assertIsNone(u)
```

Other assertions

- other assertions
 - `assertEqual(a,b)` / `assertNotEqual(a,b)`
 - `assertTrue(p)` / `assertFalse(p)`
 - `assertIsNone(a)` / `assertIsNotNone(a)`

```
self.assertFalse(0) # Passes  
self.assertFalse(None) # Passes  
#self.assertIsNone(False) # Fails
```

Activity

- Implement the methods of UserManager above
- Implement appropriate tests