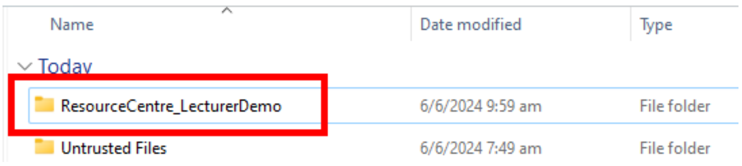
Setup a project in VS Code

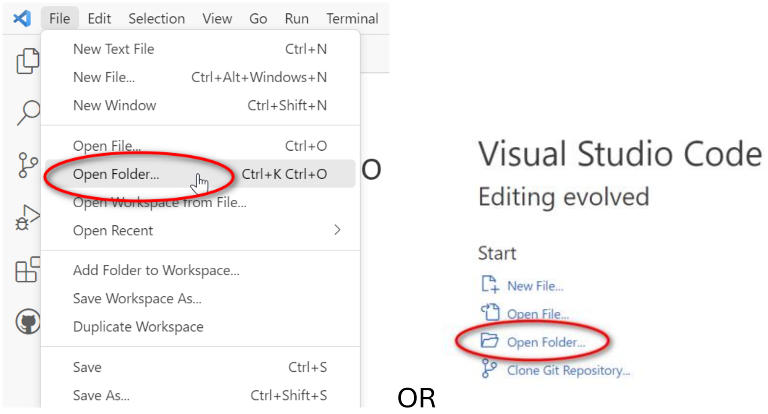
1. Create a folder using **File Explorer**. Name this folder as **ResourceCentre_<StudentID>.**



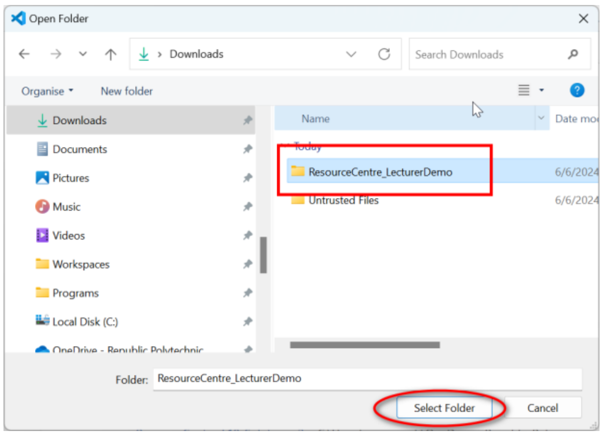
Name	Date modified	Type
Today		
ResourceCentre_LecturerDemo	6/6/2024 9:59 am	File folder
Untrusted Files	6/6/2024 7:49 am	File folder

Note: remember to replace the "LecturerDemo" with your Student ID.

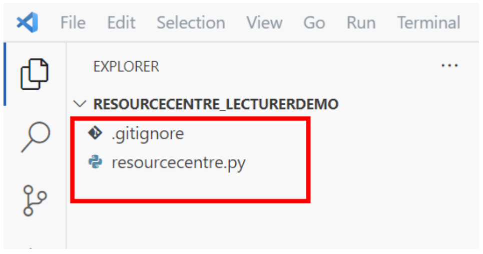
2. Launch your VS Code. Click on **Open Folder….**



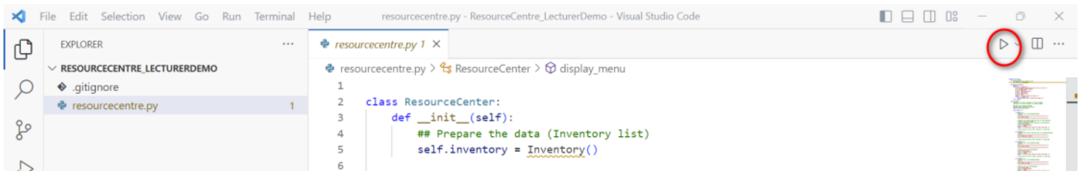
3. Select the newly created folder and click **Select Folder**.



4. Drag and drop '**.gitignore**' and '**resourcecentre.py**' into the VS Code.

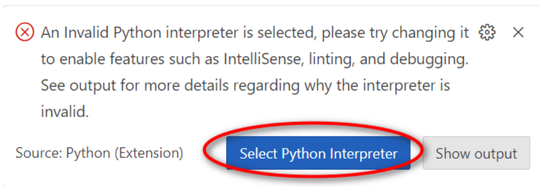

File Edit Selection View Go Run Terminal
EXPLORER
RESOURCECENTRE_LECTURERDEMO
.gitignore
resourcecentre.py

5. Open the **resourcecentre.py** and click on the 'run' button.

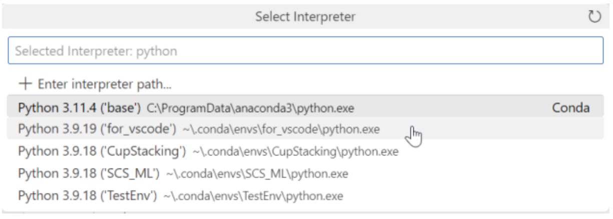

File Edit Selection View Go Run Terminal Help
resourcecentre.py - ResourceCentre_LecturerDemo - Visual Studio Code
EXPLORER
RESOURCECENTRE_LECTURERDEMO
.gitignore
resourcecentre.py 1
resourcecentre.py 1
resourcecentre.py > ResourceCenter > display_menu

```
class ResourceCenter:
    def __init__(self):
        ## Prepare the data (Inventory list)
        self.inventory = Inventory()
```

6. If you see a message as shown below, click on **Select Python Interpreter.** Select a suitable python interpreter. (Select 'base' if you are not sure.)



An Invalid Python interpreter is selected, please try changing it to enable features such as IntelliSense, linting, and debugging. See output for more details regarding why the interpreter is invalid.



```
Python 3.11.4 ('base') C:\ProgramData\anaconda3\python.exe                Conda
Python 3.9.19 ('for_vscode') ~\.conda\envs\for_vscode\python.exe
Python 3.9.18 ('CupStacking') ~\.conda\envs\CupStacking\python.exe
Python 3.9.18 ('SCS_ML') ~\.conda\envs\SCS_ML\python.exe
Python 3.9.18 ('TestEnv') ~\.conda\envs\TestEnv\python.exe
```

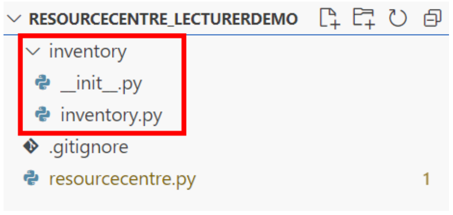
Test-Driven Development

7. Upon running the **resourcecentre.py**, you will receive an error message as shown below.

```
Microsoft Windows [Version 10.0.22621.3593]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jimmy_goh\Downloads\ResourceCentre_LecturerDemo>C:/Users/jimmy_goh/.conda/envs/for_vscode/python.exe c:/
Users/jimmy_goh/Downloads/ResourceCentre_LecturerDemo/resourcecentre.py
Traceback (most recent call last):
  File "c:\Users\jimmy_goh\Downloads\ResourceCentre_LecturerDemo\resourcecentre.py", line 96, in <module>
    app = ResourceCenter()
  File "c:\Users\jimmy_goh\Downloads\ResourceCentre_LecturerDemo\resourcecentre.py", line 5, in __init__
    self.inventory = Inventory()
NameError: name 'Inventory' is not defined
```

This error is due to the missing **Inventory** class. Now we shall create the class.

8. Create a new folder named **inventory.**
9. In this **inventory** folder, create two python files named **__init__.py** and **inventory.py**.



10. Open **inventory.py** and copy-paste the code below into it.

```
class Inventory():
    pass
```

11. In **resourcecentre.py**, insert the code below to Line 1.

```
from inventory.inventory import Inventory
```

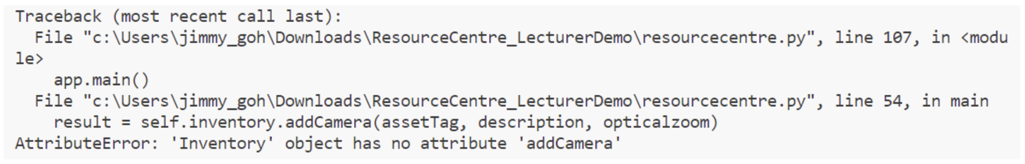
12. Run and there should be no error now. However, the code does nothing.

13. Now we code the adding of a digital camera into the inventory.
    Copy-paste the code below to its correct position. (*Observe the comment*)

```
# TO-DO: Write the code to ADD a digital camera or laptop.
if option == 1:
    assetTag = input("Enter asset tag >")
    description = input("Enter descrition >")
    opticalzoom = int(input("Enter optical zoom >"))

    result = self.inventory.addCamera(assetTag, description, opticalzoom)

    if result:
        print("Digital camera added.")
    else:
        print("Error adding digital camera.")
```

Note: You need to manually adjust the indentation after pasting the code.

14. Run the code and you will receive an error message when attempting to add a camera into the inventory.



```
Traceback (most recent call last):
  File "c:\Users\jimmy_goh\Downloads\ResourceCentre_LecturerDemo\resourcecentre.py", line 107, in <modu
le>
    app.main()
  File "c:\Users\jimmy_goh\Downloads\ResourceCentre_LecturerDemo\resourcecentre.py", line 54, in main
    result = self.inventory.addCamera(assetTag, description, opticalzoom)
AttributeError: 'Inventory' object has no attribute 'addCamera'
```

This error is due to the missing method **addCamera()** in the **Inventory** class.

15. Before we start to code for the **addCamera()**, in the TDD approach, we need to write the test cases for this method first.
    Based on the *acceptance criteria* of User Story 1 (US_01), we design the test cases.

User Story 1

As a Resource Centre staff member, I want to be able to input details of laptops or digital cameras into the system, so that I can easily track the ownership and specifications of each device.

Given that there is a new laptop or digital camera, when the Resource Centre staff member inputs the details of laptops or digital cameras into the system, then the system should store the ownership and specifications of each device for easy tracking.

Acceptance Criteria:

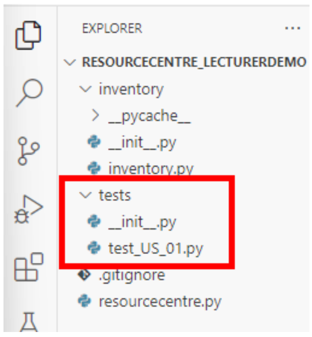
- The system allows the staff member to input all required details of a laptop or digital camera.
- The system validates that all required details are entered before allowing the data to be saved.
- The system displays a confirmation message to the staff member once the data is successfully saved.

No.	**Unit Test Case**	**Expected Result**
1	Add a digital camera into the inventory. - Add one camera to the list. - Assert Tag: C001 - Description: Test camera 1 - Optical zoom: 5	The digital camera is added to the inventory.

2	Add a digital camera that already exists in the inventory. - Add one camera to the list. - Assert Tag: C002 - Description: Test camera 2 - Optical zoom: 10 - Add the same camera to the list.	The digital camera is not added to the inventory.
3	Add a digital camera with missing description. - Add one camera to the list. - Assert Tag: C004 - Description: <no description> - Optical Zoom: 10	The digital camera is not added to the inventory.
4	Add a digital camera with incorrect optical zoom. - Add one camera to the list. - Assert Tag: C004 - Description: Test camera 4 - Optical Zoom: -1	The digital camera is not added to the inventory.

Note: This test case documentation is not compulsory for Scrum Framework. However, it guides you to design and derive the necessary test code for the `pytest`.

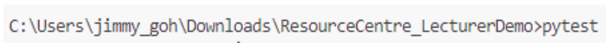
16. Next, we will code the test cases into pytest codes.
    Create a folder named **tests.**
    In this **tests** folder, create two files named **__init__.py** and **test_US_01.py**.



17. Copy-paste the pytest codes below into **test_US_01.py**.

```python
from inventory.inventory import Inventory

class Test_US_01:
    ############### Test add camera ######################
    def test_add_camera(self):
        test_inventory = Inventory()
        assert len(test_inventory.cameraList) == 0

        result = test_inventory.addCamera("C001", "Test camera 1", 5)

        assert result == True
        assert len(test_inventory.cameraList) == 1
```

18. Open a new **Terminal** and run **pytest**.



```
C:\Users\jimmy_goh\Downloads\ResourceCentre_LecturerDemo>pytest
```

19. The test will *fail*.

```
================================================== short test summary info ===================================================
FAILED tests/test_US_01.py::Test_US_01::test_add_camera - AttributeError: 'Inventory' object has no attribute 'cameraList'
===================================================== 1 failed in 0.11s ======================================================
```

The error message shows that there are missing attributes 'cameraList'.

20. Based on the error message, we add in the code to **inventory.py** to create the 'cameraList'.

```
    def __init__(self):
        self.cameraList = []
```

21. Run **pytest** again and a new error message appears.

```
================================================== short test summary info ===================================================
FAILED tests/test_US_01.py::Test_US_01::test_add_camera - AttributeError: 'Inventory' object has no attribute 'addCamera'
===================================================== 1 failed in 0.12s ======================================================
```

This error message shows that the method 'addCamera()' is missing.

22. Therefore, we add in the 'addCamera()' method to fix the error.

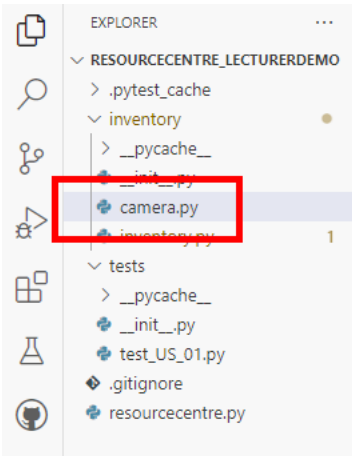
```
    def addCamera(self, assetTag, description, opticalzoom):
        new_camera = Camera(assetTag, description, opticalzoom)
        self.cameraList.append(new_camera)

        return True
```

23. Run **pytest** again and a new error message appears.

```
================================================== short test summary info ===================================================
FAILED tests/test_US_01.py::Test_US_01::test_add_camera - NameError: name 'Camera' is not defined
===================================================== 1 failed in 0.12s ======================================================
```

This error message indicates that the 'Camera' class is missing.

24. We will now code for the missing **Camera** class.
In the **inventory** folder, create a new file named **camera.py**.



25. Copy-paste the following codes into **camera.py**.

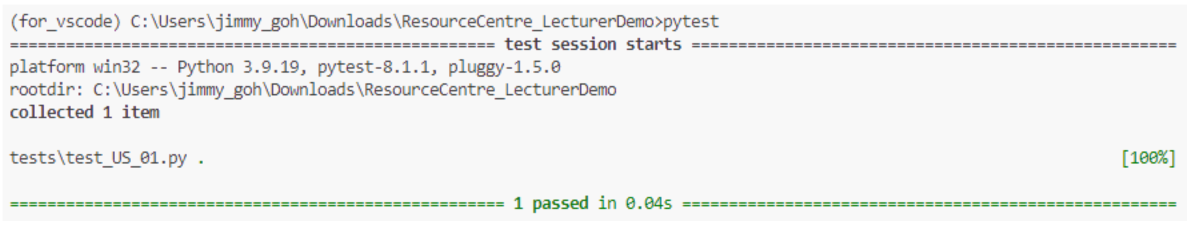
```
class Camera():
    # Refactor (E): Extract duplicate attributes and methods.
    # There are several common attributes and methods in
    # Camera.py and Laptop.py. Extract these common attributes
    # and methods into a super class, named item.py
    def __init__(self, assetTag, description, opticalZoom):
        self._assetTag = assetTag
        self._description = description
```

```python
        self._dueDate = ""
        self._isAvailable = True
        self._opticalZoom = opticalZoom

    def getAssetTag(self):
        return self._assetTag

    def getDescription(self):
        return self._description

    def getDueDate(self):
        return self._dueDate

    def getIsAvailable(self):
        if self._isAvailable:
            return "Yes"
        else:
            return "No"

    def getOpticalZoom(self):
        return self._opticalZoom

    def setDueDate(self, dueDate):
        self._dueDate = dueDate

    def setIsAvailable(self, isAvailable):
        self._isAvailable = isAvailable
```

26. Add the import code below to **inventory.py**.

```python
from inventory.camera import Camera
```

27. Run **pytest** and now the test passes.



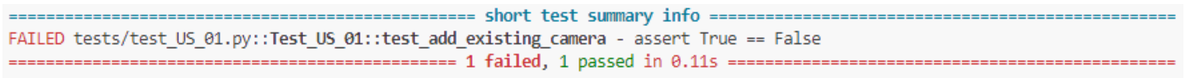
```
(for_vscode) C:\Users\jimmy_goh\Downloads\ResourceCentre_LecturerDemo>pytest
====================================================== test session starts ======================================================
platform win32 -- Python 3.9.19, pytest-8.1.1, pluggy-1.5.0
rootdir: C:\Users\jimmy_goh\Downloads\ResourceCentre_LecturerDemo
collected 1 item

tests\test_US_01.py .                                                                                                     [100%]

======================================================= 1 passed in 0.04s =======================================================
```

28. After the test passes, we should perform "code refactoring". We will cover code refactoring in the next lesson, we will skip this action for now.

29. We now add in the next test case to **test_US_01.py**.

```python
    def test_add_existing_camera(self):
        test_inventory = Inventory()
        result = test_inventory.addCamera("C001", "Test camera 1", 5)
        result = test_inventory.addCamera("C002", "Test camera 2", 10)
        original_len = len(test_inventory.cameraList)

        result = test_inventory.addCamera("C002", "Test camera 2", 10)

        assert result == False
        assert len(test_inventory.cameraList) == original_len
```

30. Run **pytest** and it will fail (as expected).



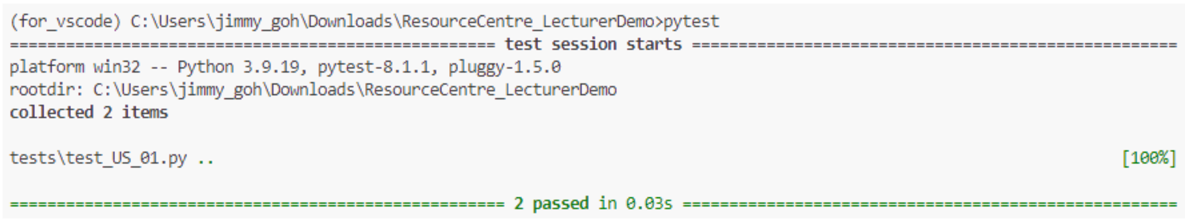
```
====================================================== short test summary info ======================================================
FAILED tests/test_US_01.py::Test_US_01::test_add_existing_camera - assert True == False
================================================= 1 failed, 1 passed in 0.11s ==================================================
```

This error is because the code does not check for duplicate asset tag number.
Note: Asset tag numbers must be unique.

31. We add in the code to check for duplicate asset tag numbers.
Copt-paste the codes below and ***replace*** the existing 'addCamera()' method.

```
    def addCamera(self, assetTag, description, opticalzoom):
        # Refactor (C): Extract long methods to findCamera(assetTag),
        # return the found camera, return None if not found.
        # **Don't forget to create test cases for this new method.
        # Check for existing camera
        notExist = True
        for c in self.cameraList:
            currentTag = c.getAssetTag()
            if currentTag == assetTag:
                notExist = False
                error_message = "Asset already exists."

        if notExist:
            new_camera = Camera(assetTag, description, opticalzoom)
            self.cameraList.append(new_camera)
            return True
        else:
            print(error_message)
            return False
```

32. Run **pytest** and all the test passes.

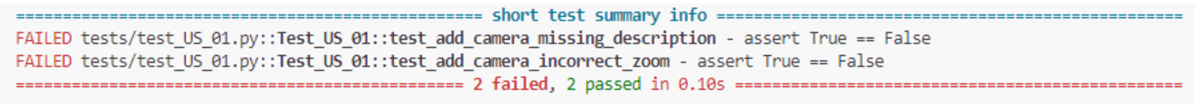


```
(for_vscode) C:\Users\jimmy_goh\Downloads\ResourceCentre_LecturerDemo>pytest
====================================================== test session starts ======================================================
platform win32 -- Python 3.9.19, pytest-8.1.1, pluggy-1.5.0
rootdir: C:\Users\jimmy_goh\Downloads\ResourceCentre_LecturerDemo
collected 2 items

tests\test_US_01.py ..                                                                                                    [100%]

====================================================== 2 passed in 0.03s =======================================================
```

33. Now we add in the last two test cases to **test_US_01.py**.

```
    def test_add_camera_missing_description(self):
        test_inventory = Inventory()
        result = test_inventory.addCamera("C001", "Test camera 1", 5)
        result = test_inventory.addCamera("C002", "Test camera 2", 10)
        original_len = len(test_inventory.cameraList)

        result = test_inventory.addCamera("C004", "", 10)

        assert result == False
        assert len(test_inventory.cameraList) == original_len

    def test_add_camera_incorrect_zoom(self):
        test_inventory = Inventory()
        result = test_inventory.addCamera("C001", "Test camera 1", 5)
        result = test_inventory.addCamera("C002", "Test camera 2", 10)
        original_len = len(test_inventory.cameraList)

        result = test_inventory.addCamera("C004", "Test camera 4", -1)

        assert result == False
        assert len(test_inventory.cameraList) == original_len
```

34. Run **pytest** and the two new tests will fail.



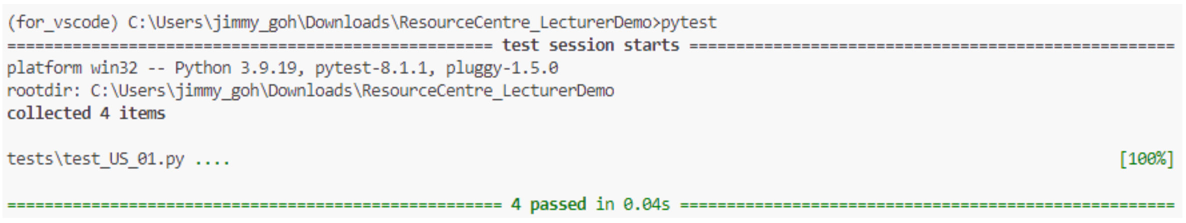
```
================================================= short test summary info =================================================
FAILED tests/test_US_01.py::Test_US_01::test_add_camera_missing_description - assert True == False
FAILED tests/test_US_01.py::Test_US_01::test_add_camera_incorrect_zoom - assert True == False
================================================ 2 failed, 2 passed in 0.10s ================================================
```

This error is due to no validation on the arguments.

35. Now we add in the codes to validate the arguments.
    Copy-paste the codes below to **replace** the 'addCamera()' method.

```
    def addCamera(self, assetTag, description, opticalzoom):
        # Check for correct values
        correct = True
        if len(assetTag)==0 or len(description)==0 or opticalzoom<0:
            correct = False
            error_message = "Incorrect values."

        # Refactor (C): Extract long methods to findCamera(assetTag),
        # return the found camera, return None if not found.
        # **Don't forget to create test cases for this new method.
        # Check for existing camera
        notExist = True
        for c in self.cameraList:
            currentTag = c.getAssetTag()
            if currentTag == assetTag:
                notExist = False
                error_message = "Asset already exists."

        if correct and notExist:
            new_camera = Camera(assetTag, description, opticalzoom)
            self.cameraList.append(new_camera)
            return True
        else:
            print(error_message)
            return False
```

36. Run **pytest** again and now all the 4 tests passed.



```
(for_vscode) C:\Users\jimmy_goh\Downloads\ResourceCentre_LecturerDemo>pytest
=================================================== test session starts ===================================================
platform win32 -- Python 3.9.19, pytest-8.1.1, pluggy-1.5.0
rootdir: C:\Users\jimmy_goh\Downloads\ResourceCentre_LecturerDemo
collected 4 items

tests\test_US_01.py ....                                                                                            [100%]

==================================================== 4 passed in 0.04s ====================================================
```

37. Run the **resourcecentre.py** and add a new camera.
    There should be no error now.

38. You have successfully developed the feature of adding a new camera.

Test-Driven Development for adding a new laptop

39. Repeat the TDD process of adding a new camera and code for the adding a new laptop.

40. The code below is for the user interface to add a new laptop.

```
            elif option == 2:
                assetTag = input("Enter asset tag >")
                description = input("Enter descrition >")
                os = input("Enter os >")

                result = self.inventory.addLaptop(assetTag, description, os)

                if result:
                    print("Laptop added.")
                else:
                    print("Error adding laptop.")
            else:
                print("Invalid item type.")
```

41. Below are the *pytest codes* for **addLaptop()**.

```
    ############### Test add laptop ######################
    def test_add_laptop(self):
        test_inventory = Inventory()
        assert len(test_inventory.laptopList) == 0

        result = test_inventory.addLaptop("L001", "Test Laptop 1", "WINXP")

        assert result == True
        assert len(test_inventory.laptopList) == 1

    def test_add_existing_laptop(self):
        test_inventory = Inventory()
        result = test_inventory.addLaptop("L001", "Test Laptop 1", "WINXP")
        result = test_inventory.addLaptop("L002", "Test Laptop 2", "MACOS")
        original_len = len(test_inventory.laptopList)

        result = test_inventory.addLaptop("L002", "Test Laptop 2", "MACOS")

        assert result == False
        assert len(test_inventory.laptopList) == original_len

    def test_add_laptop_missing_description(self):
        test_inventory = Inventory()
        result = test_inventory.addLaptop("L001", "Test Laptop 1", "WINXP")
        result = test_inventory.addLaptop("L002", "Test Laptop 2", "MACOS")
        original_len = len(test_inventory.laptopList)

        result = test_inventory.addLaptop("L004", "", "WIN10")

        assert result == False
        assert len(test_inventory.laptopList) == original_len

    def test_add_laptop_missing_os(self):
        test_inventory = Inventory()
        result = test_inventory.addLaptop("L001", "Test Laptop 1", "WINXP")
        result = test_inventory.addLaptop("L002", "Test Laptop 2", "MACOS")
        original_len = len(test_inventory.laptopList)

        result = test_inventory.addLaptop("L004", "Test Laptop 4", "")
        assert result == False
        assert len(test_inventory.laptopList) == original_len
```

42. The code below is for the **addLaptop()** method.

```
from inventory.laptop import Laptop
```

```
    def __init__(self):
        self.cameraList = []
        self.laptopList = []

    def addLaptop(self, assetTag, description, os):
        # Check for correct values
        correct = True
        if len(assetTag)==0 or len(description)==0 or len(os)==0:
            correct = False
            error_message = "Incorrect values."

        # Refactor (C): Extract long methods to findLaptop(assetTag),
        # return the found laptop, return None if not found.
        # **Don't forget to create test cases for this new method.
        # Check for existing laptop
        notExist = True
        for l in self.laptopList:
            currentTag = l.getAssetTag()
            if currentTag == assetTag:
                notExist = False
                error_message = "Asset already exists."

        if correct and notExist:
            new_laptop = Laptop(assetTag, description, os)
            self.laptopList.append(new_laptop)
            return True
        else:
            print(error_message)
            return False
```

43. The code below is for the **Laptop** class.

```
class Laptop():
    # Refactor (E): Extract duplicate attributes and methods.
    # There are several common attributes and methods in
    # Camera.py and Laptop.py. Extract these common attributes
    # and methods into a super class, named item.py
    def __init__(self, assetTag, description, os):
        self._assetTag = assetTag
        self._description = description
        self._dueDate = ""
        self._isAvailable = True
        self._os = os

    def getAssetTag(self):
        return self._assetTag

    def getDescription(self):
        return self._description

    def getDueDate(self):
        return self._dueDate

    def getIsAvailable(self):
        if self._isAvailable:
            return "Yes"
        else:
            return "No"

    def getOS(self):
        return self._os

    def setDueDate(self, dueDate):
        self._dueDate = dueDate

    def setIsAvailable(self, isAvailable):
        self._isAvailable = isAvailable
```