Classroom Lab 5

ITEC 1150 Programming Logic and Design

Hi, I'm Clara

- I'm an instructor at MCTC too, mostly 2nd-year programming classes
- I've been working on a project to help students work on their assignments

Problem

- How do you know that your code is working correctly?
  - And, if you change it, or add features, does it still work?
- Sure, you run and test it - but there's lots of possible valid and inputs and expected outputs
- And when you develop and change code, it's a lot of work to keep testing it (with a range of valid and invalid inputs) to make sure everything is still working

Solution

- Running code with different inputs every time it's changed is repetitive and boring
- Repetitive, boring tasks are perfect tasks for a program!
- So, we are working on bundling **automatic tests** with assignments
- **tests** = code that runs your code, and checks if your code is working correctly

Solution

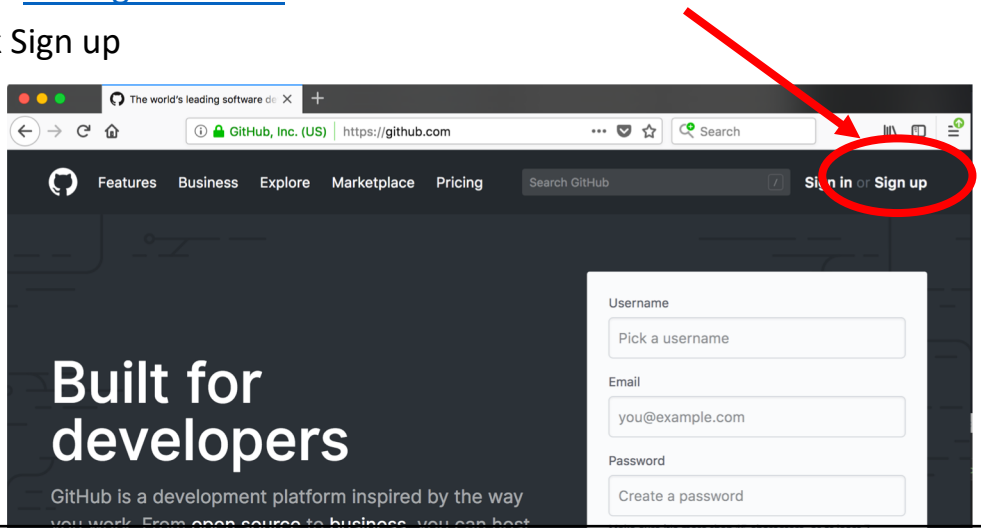
- Covering how to write automatic tests is outside of the scope of this class
  - But please ask me after class if you are interested in learning more!
  - And you'll cover testing in 2nd year programming classes
- So, we can give you a PyCharm project with the tests in
- You'll write code as usual
- And then you can run the tests as you work, to make sure your code is running correctly - in the way the assignment or lab expects it to

Sharing code - GitHub

- Could give you a zip file of code, or put code in D2L...
- But that's not what professional code developers do. It's not organized enough for bigger projects
- We'll use the **GitHub** website to share the initial code (and tests) with you
- PyCharm can download code from GitHub
- When you've written your code, PyCharm can upload your work to GitHub
- No zip files, no word documents :)
- A little tricky to start with but really easy once you've got the hang of it

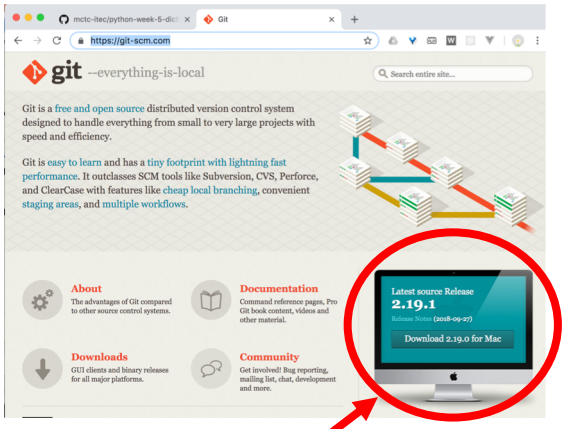
Sign up for a GitHub account

- Go to www.github.com
- Click Sign up



Working on your own Windows PC?

- Install the git application from https://git-scm.com/
- Use all the defaults for the installation
- **Shout at me when the install is done, you need to configure PyCharm to find git, I'll walk you through it**
- Mac and Linux users: you already have git, skip this step
- Lab PCs: you already have git, skip this step



Download button

GitHub

- A code hosting website
- Hosts open-source code for millions of programs
- You can save your own code there too - useful for
  - backups
  - sharing your code - with the instructor, other students
  - making a portfolio of your work - great for showing off your skills job-hunting

And, instructors can share code with you

- Today, we'll share an outline for your classroom lab assignment
- It includes the start of the program you'll finish in lab
- And, some tests to check if your program is working correctly

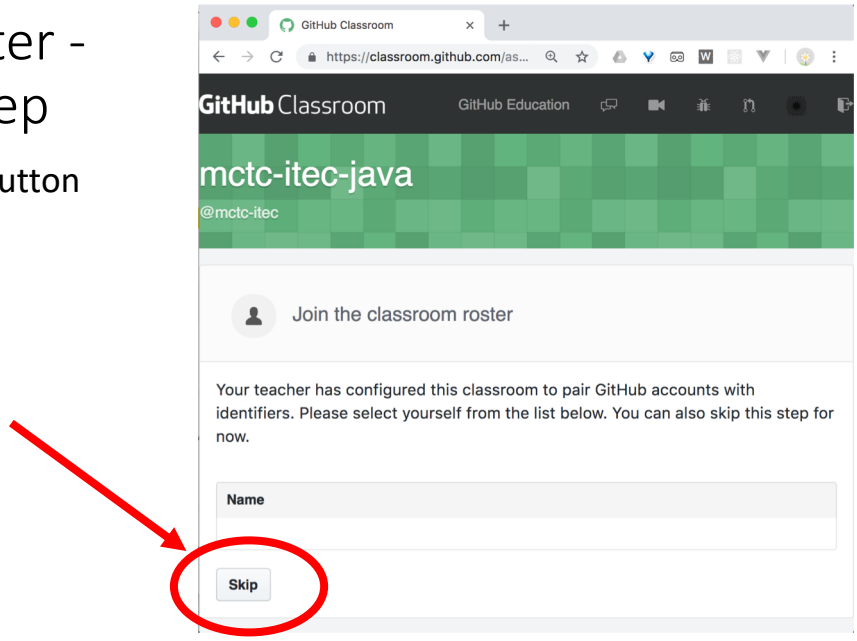
Get the code

- Make sure you have activated your GitHub account
- And are signed into GitHub

- Now, go to this link: https://classroom.github.com/a/jdIuJE9T

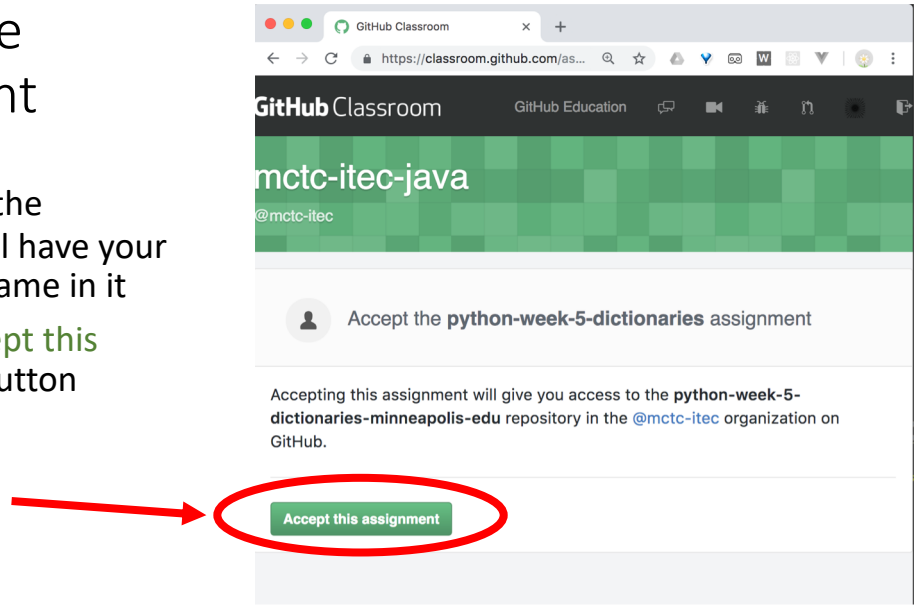
Add to roster - skip this step

- Click the Skip button



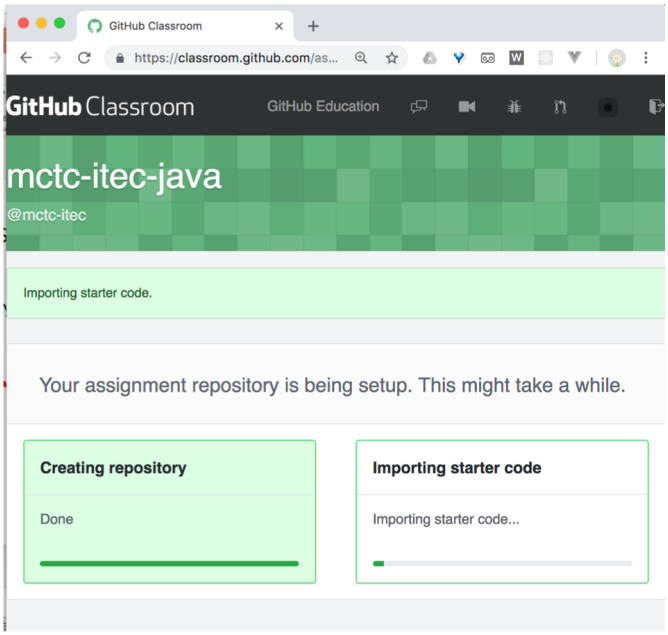
Accept the assignment

- The name of the repository will have your GitHub username in it
- Click the Accept this assignment button



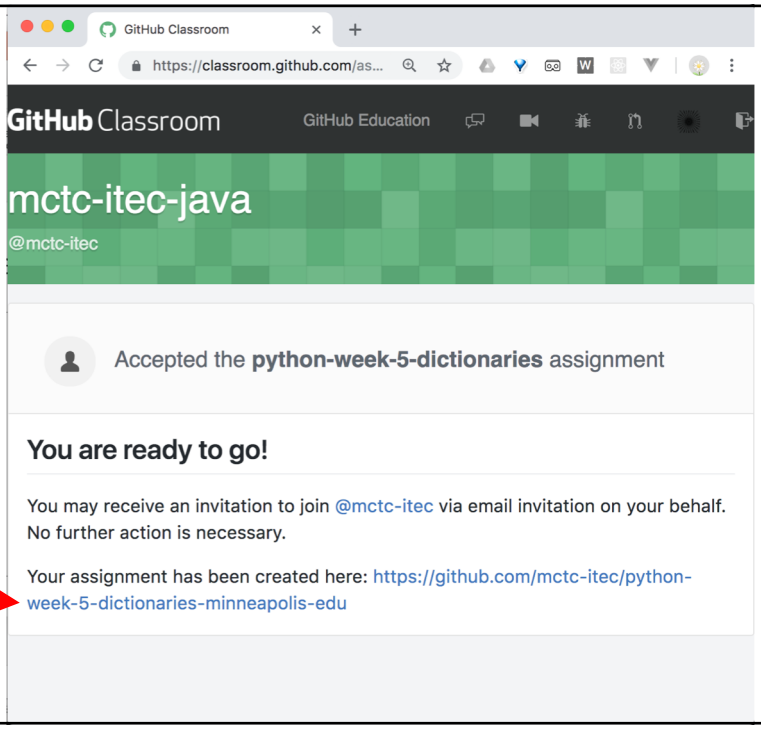
Wait...

- GitHub will create a repository for you
- This will have a copy of a starter PyCharm project
- For this lab, you'll finish the code given



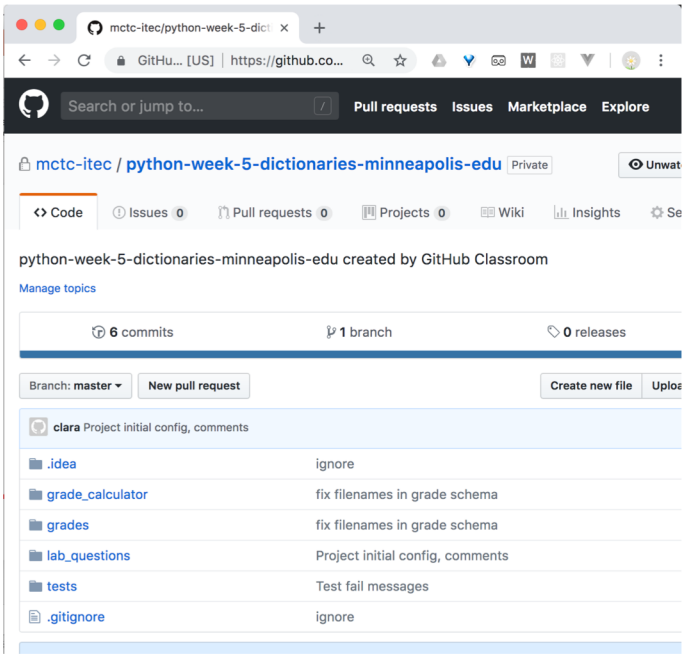
Repository Created

- Make a copy of the 'Your assignment has been created here' link - this is the link for your repository
- Yours will be different to the screenshot - it will have your own GitHub name in
- Click on the repository link



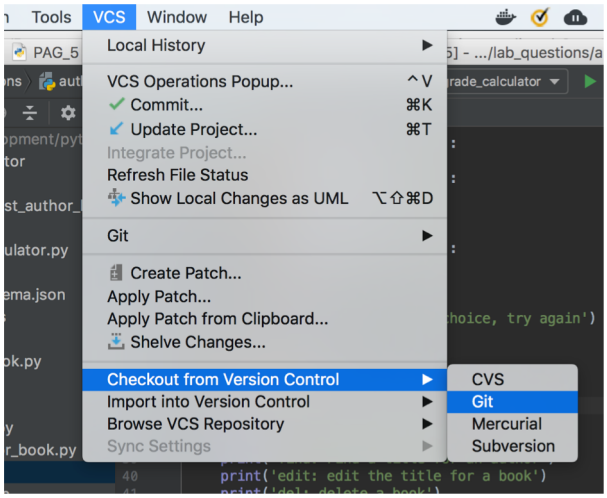
Your repository

- Here's your repository
- This is your copy of the code
- **Keep this page open - you'll need the URL several times, you'll come back and check this page during the lab**



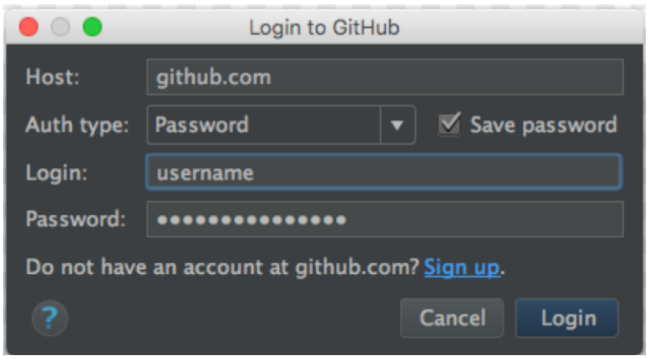
Getting the code onto your computer

- PyCharm can download code from GitHub
- Open PyCharm
- In the **VCS** menu, select **Checkout from Version Control**, and then **Git**



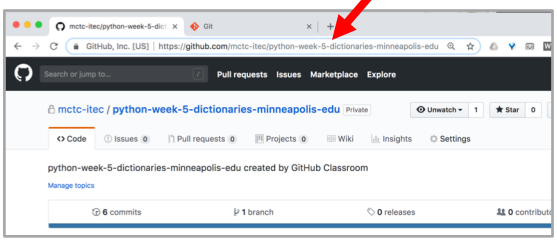
Sign in to GitHub

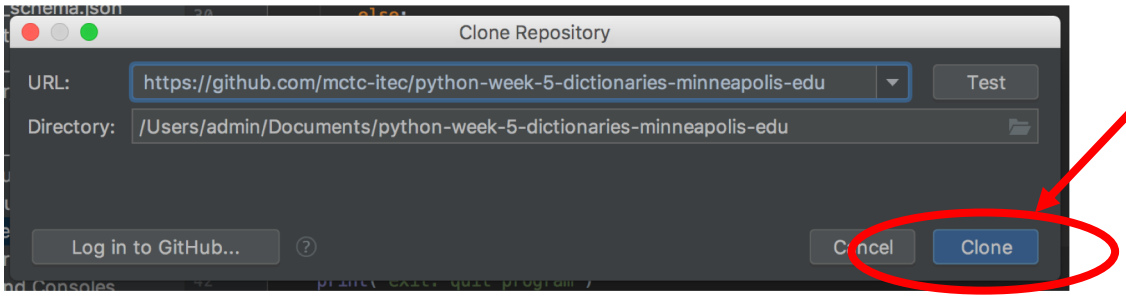
- Make sure the Auth type: is **Password**
- Enter your GitHub login name and password
- Click Login



Clone (download) your repository

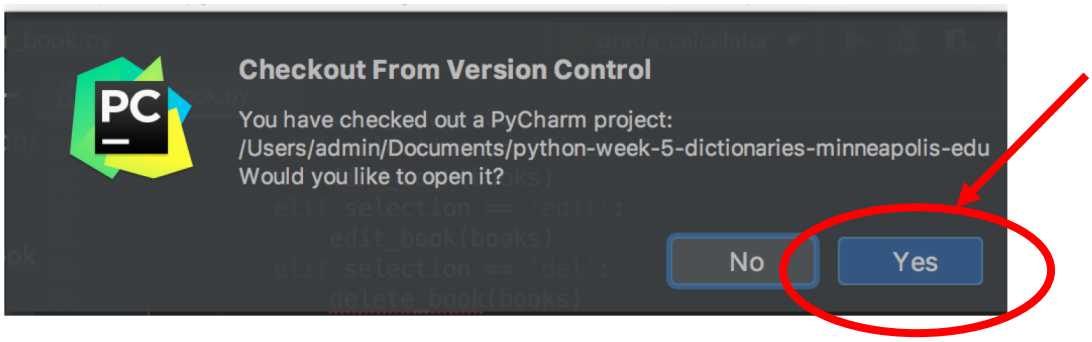
- Paste the URL from your repository into the URL field
- Change the directory location if you like, or keep the suggested location
  - Your code will be downloaded here
- Click the **Clone** button





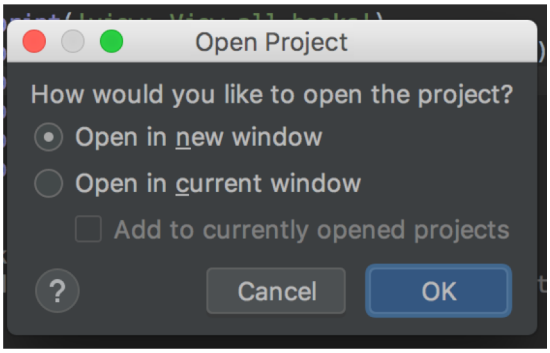
Would you like to open the project?

- Click Yes



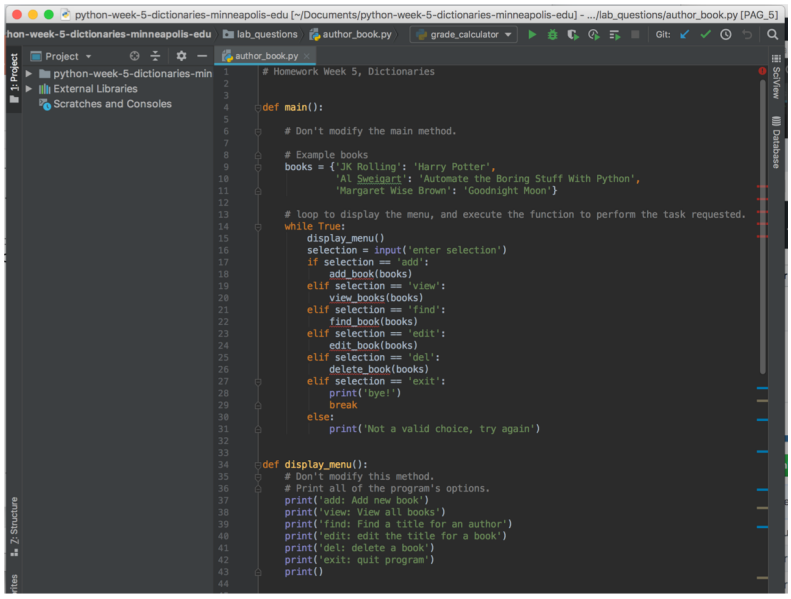
How would you like to open the project?

- If you already have a PyCharm project open, you'll see this window
- Pick either option



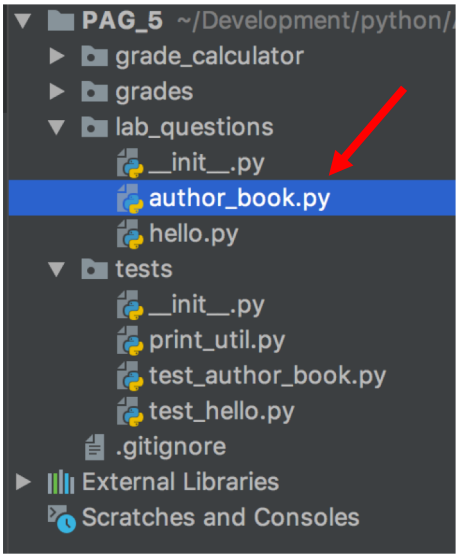
Project code

- Some code is already written for you.
- Your task for this lab is to finish it



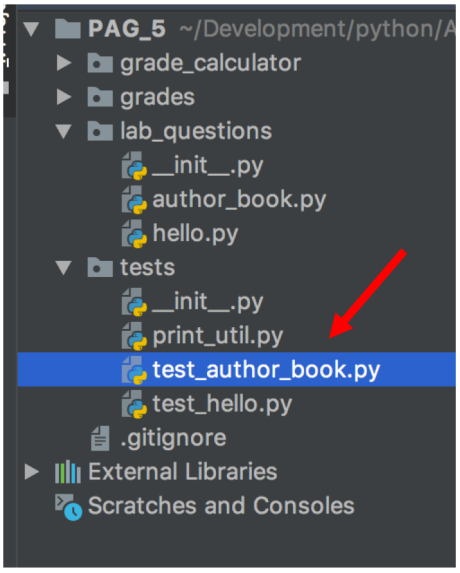
Project code

- One file you will be working with is called **author_book.py** and it is in the **lab_questions** directory
- Find this file and open it, if it's not already open
- Click on the triangles by a directory name to expand them and see the files



Where are the tests?

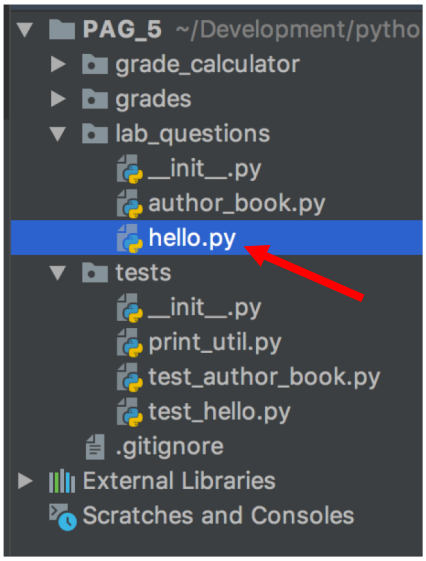
- They are in a file called **test_author_book.py** in the **tests** directory
- You don't need to modify this file,
  - although you can read it
  - and you'll run it, to check your code
- You can ignore all the other files in the project



But first: we'll start with a Hello World example

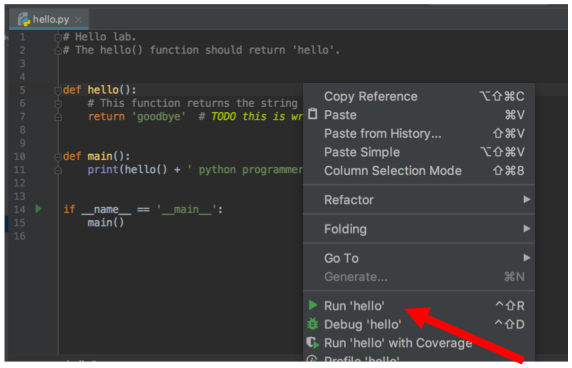
- Open the other python file in **lab_questions** called **hello.py**

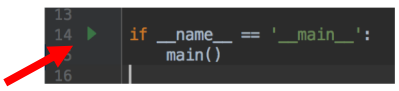
```
hello.py
1  # Hello lab.
2  # The hello() function should return 'hello'.
3
4
5  def hello():
6      # This function returns the string 'Hello'
7      return 'goodbye'  # TODO this is wrong! Please fix.
8
9
10 def main():
11     print(hello() + ' python programmer!')
12
13
14 if __name__ == '__main__':
15     main()
16
```



Run hello.py

- Run hello.py - same was as you've done before with other programs
  - Right click on in the editor for this file and select Run
  - Or right-click on the filename in the project explorer and select Run
  - Or click the green arrow next to the call to main





Program is not working correctly

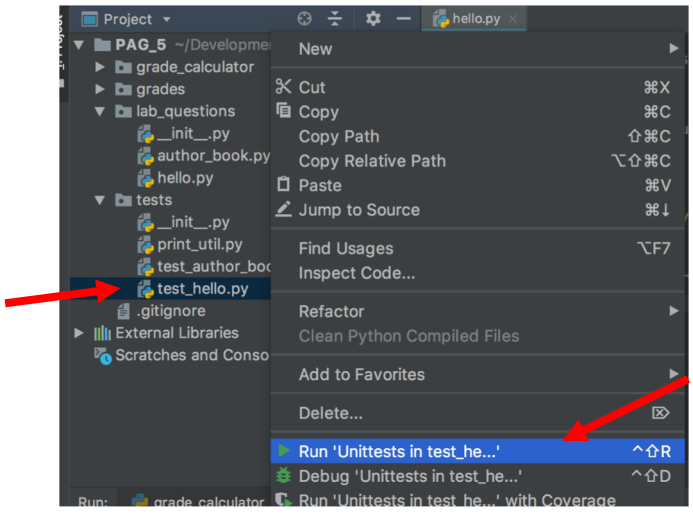
- This program should print 'hello python programmer!'
- What's wrong?

```
# Hello lab.
# The hello() function should return 'hello'.


def hello():
    # This function returns the string 'Hello'
    return 'goodbye'  # TODO this is wrong! Please fix.


def main():
    print(hello() + ' python programmer!')


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

```
Run:  hello
/Library/Frameworks/Python.framework/Versions/3.7/bin/pyth
goodbye python programmer!

Process finished with exit code 0
```

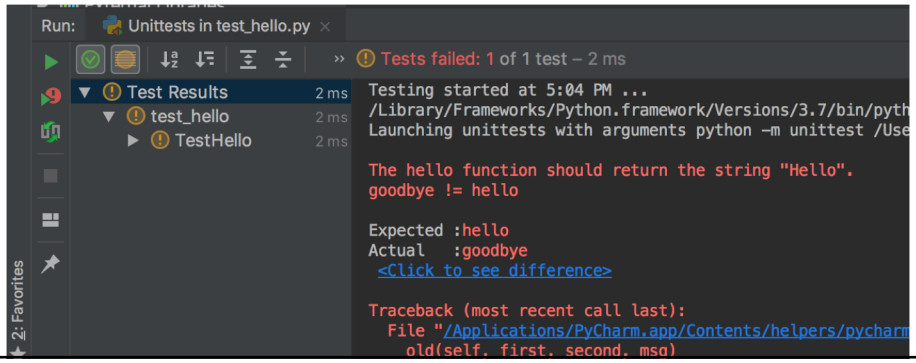
Before you fix it, let's run the test for this file

- Expand the **tests** directory
- Right-click on **test_hello.py**
- Select **Run Unittests in test_hello.py**



The tests check that the hello() function is working

- We know it isn't working, so we expect the tests to fail
- And they do - note the error message, the expected and actual output

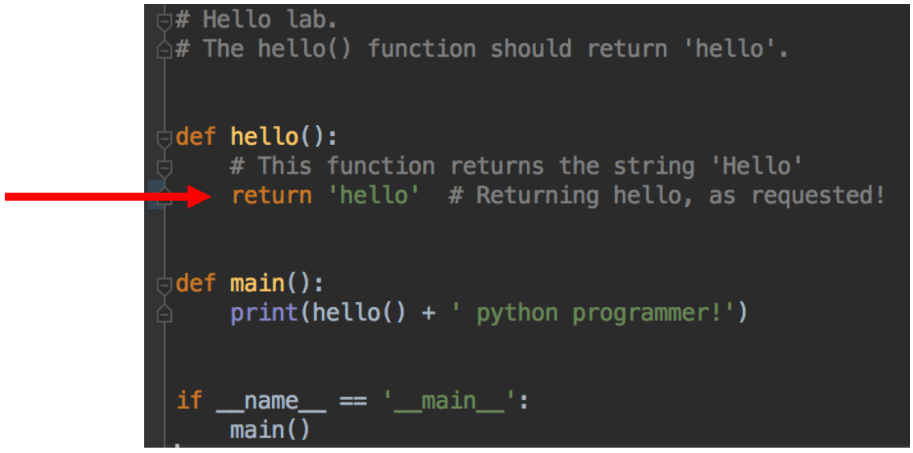


Tests

- There's only one test in tests/test_hello.py
- Other assignment will have more tests to check more complicated functions, and features
- You don't need to modify any of the test files - don't modify anything in the tests directory
  - So don't edit test_hello.py or test_author_book.py
- You only write code in the files in the lab_questions directory

Fix the code - open hello.py

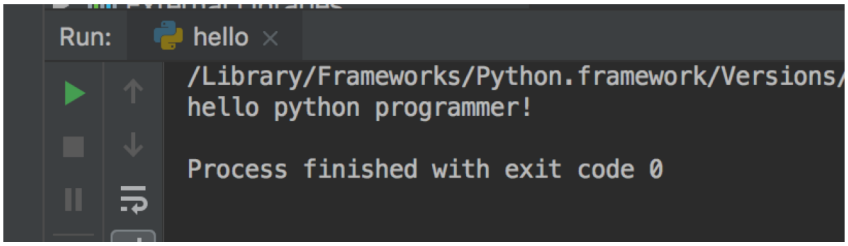
- In hello.py, fix the hello() function



```
# Hello lab.
# The hello() function should return 'hello'.


def hello():
    # This function returns the string 'Hello'
    return 'hello'  # Returning hello, as requested!


def main():
    print(hello() + ' python programmer!')


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

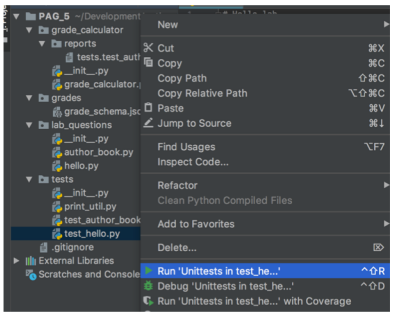
Run hello.py - does it work?

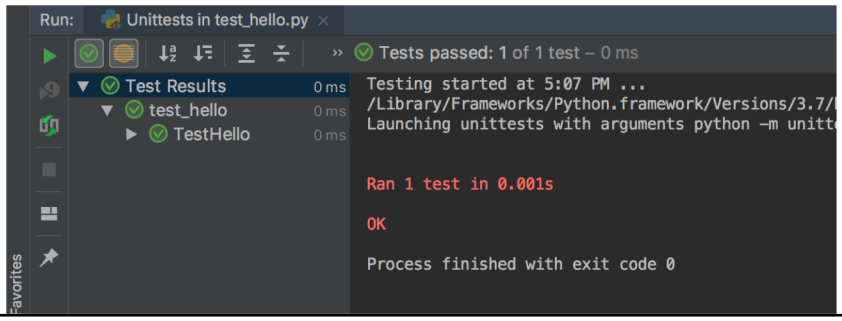
- Check your code by running it as usual
- That looks better!



Run the tests to verify

- As before - right-click on **test_hello.py**, select **Run Unittests in test_hello.py**
- Now the test should pass!
- Note the green checkmark icons by the test name to indicate the test passes





Questions?

- Is your code working?
- Does the test pass?

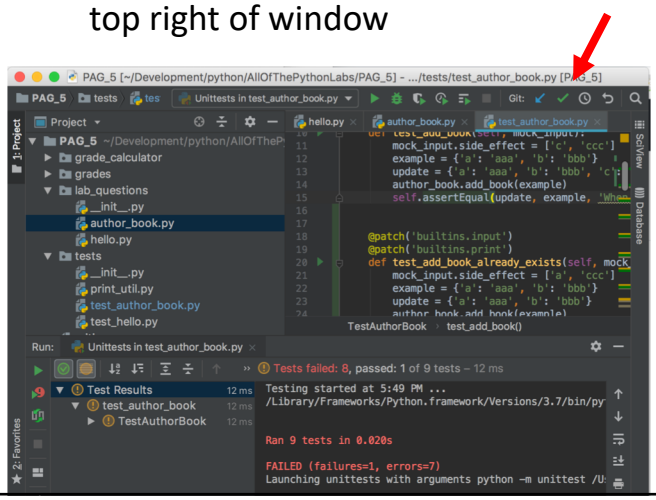
- Please shout out if not!
- Stuck? Something looks weird? Need help? Test still fails? We'll help!

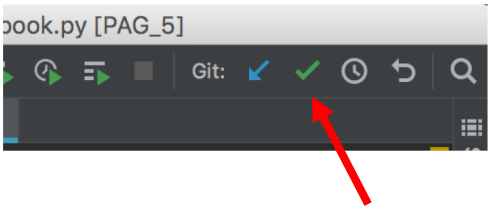
Saving code to GitHub

- PyCharm can upload your code to GitHub
- And then the instructor will be able to review and check your work

Saving code to GitHub

- Look at the PyCharm icon bar for the Commit icon - a green checkmark, top right of window



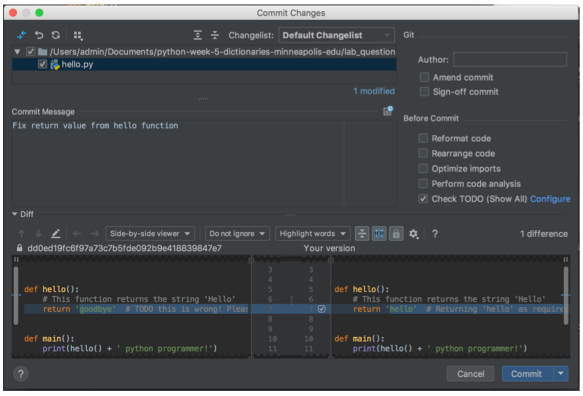


Commit

- '**Commit**' means to save a snapshot of your code
- Useful if you are working on code, and you want to save it's current state
  - For example, you just got a feature working
  - Or you just finished an assignment question
  - Or you have some code working, and you are not sure how to complete the rest of the code. You'd like to save a snapshot so you can try things out, but be able to go back to the working state if things don't go well
- **Commit** is a git operation. You'll use git a lot in 2nd year programming classes

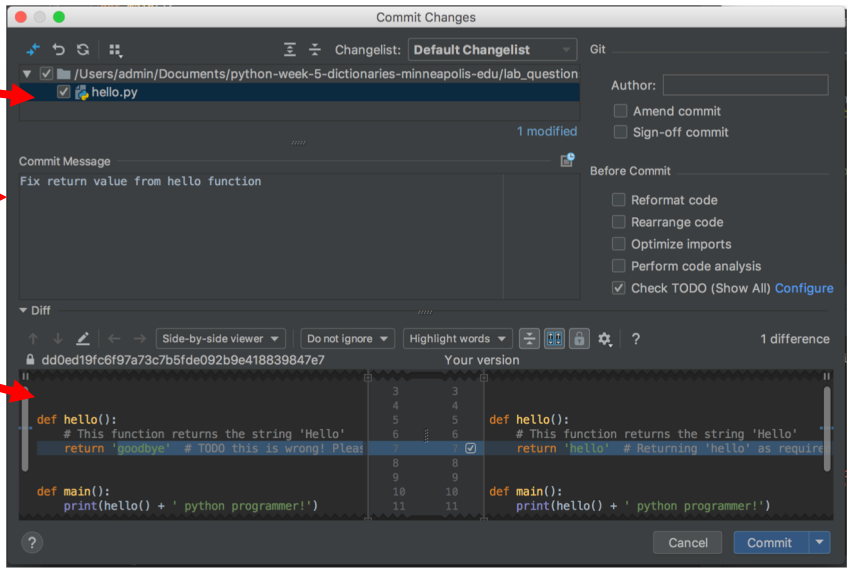
Commit

- Click on the Commit icon (the green check)
- **Commit Changes** window opens



Commit Changes

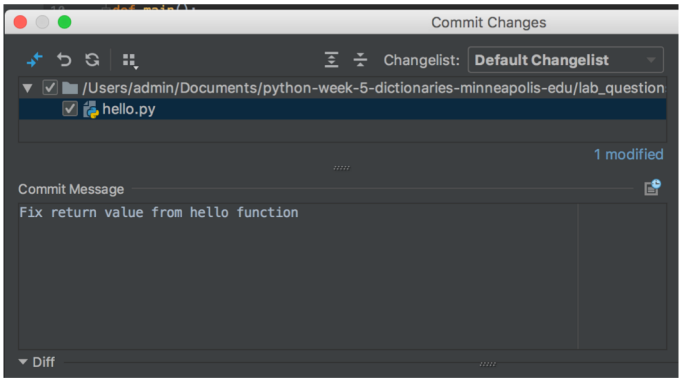
- List of project files that you have changed
- A commit message, that you will write, describing the changes
- The differences between these files
  - Can you see where 'goodbye' was changed to 'hello' ?



Commit Message

- Delete any text in the Commit Message box, and write a short new message describing what you have changed
- For example,

"Fix return value from hello function"



Commit Messages

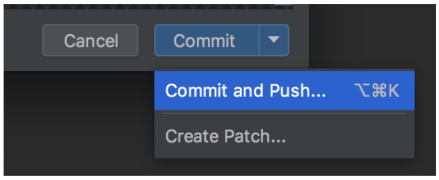
- Write descriptive commit messages
- As a professional developer, commit messages are used by your boss/team leader to track what you've worked on
- They are a very important part of a project's documentation

- More on this in 2nd year programming classes

Commit and Push

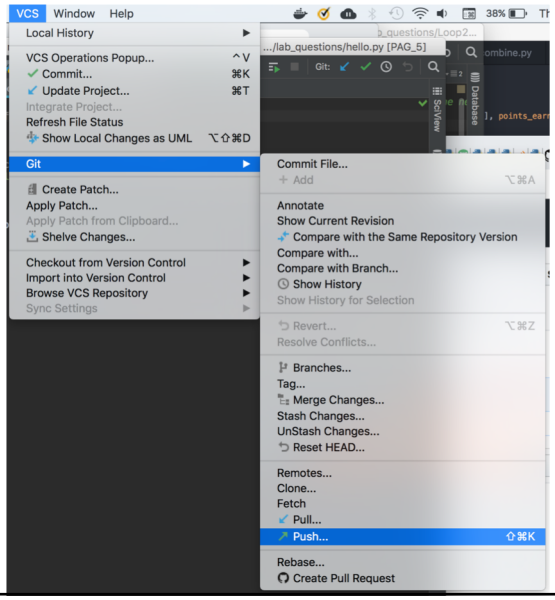
- Commit = save a snapshot
- Push = upload changed project code to GitHub

- Hover your mouse over the Commit button
- A dropdown menu will open
- Select Commit and Push



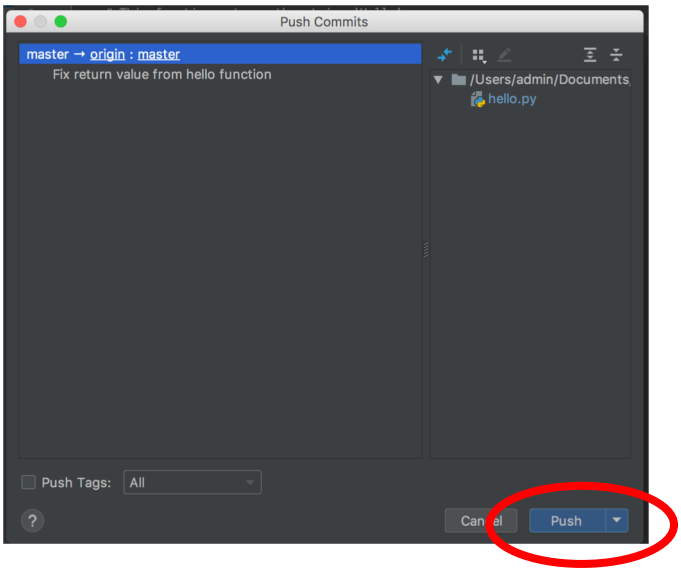
Clicked the Commit button instead of Commit and Push?

- No problem!
- Open the VCS menu, go down to Git, then select Push



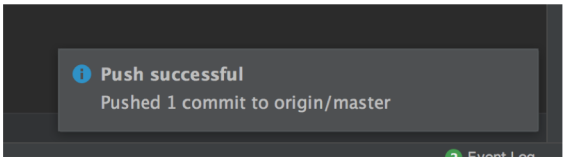
Push changes

- Shows a list of your commits to be pushed (uploaded) to GitHub
- Click the **Push** button



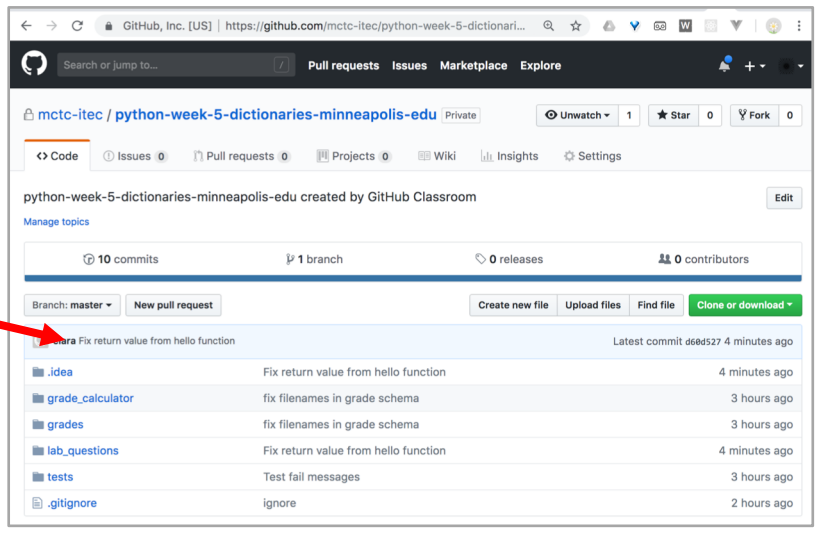
Code uploads

- Success bubble!



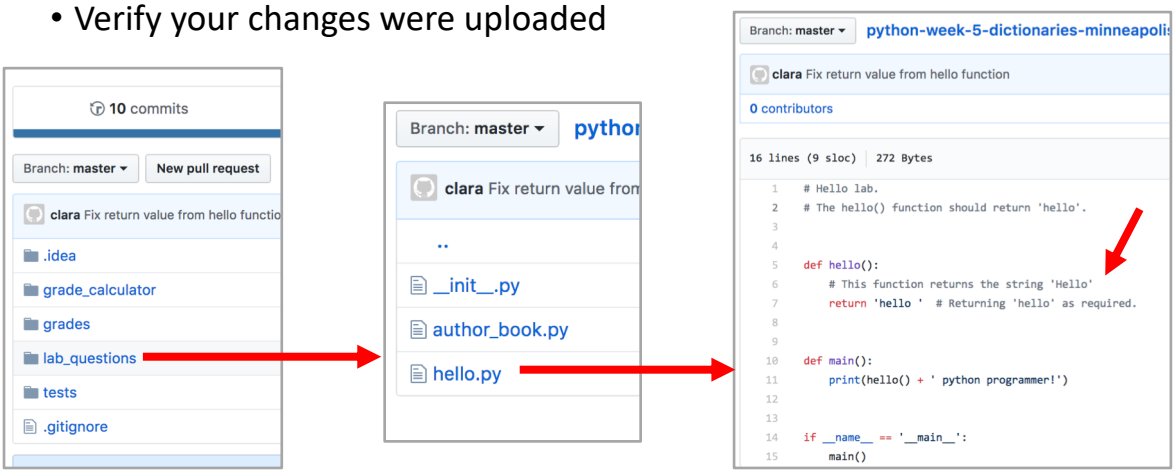
Check at GitHub

- Go to the repository URL you got earlier
- Look, there's your commit message



Check code at GitHub

- Click on lab_questions, and then hello.py
- Verify your changes were uploaded



The instructor has access to your repository

- You and the instructor can both see the code in your repository
- Otherwise, it's private, no-one else can see it

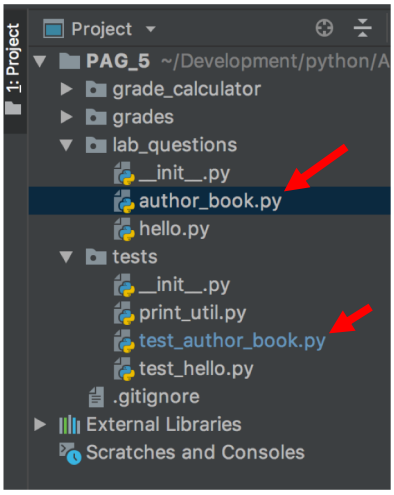
- But, by default, other GitHub repositories are public
- Useful for sharing, or creating a portfolio of your coding work

Questions?

- We've introduced many new systems and tools today
- So it might feel a bit overwhelming now...
- **Please ask if anything isn't working!**
- Don't worry - focus on getting things working today, and you'll have plenty of opportunities to practice in the future
- You'll learn more about these technologies in future programming classes
- Please ask after class if you want to learn more now!

Author and Book Program

- Dictionaries!
- Similar structure to the hello.py program,
  - the **lab_questions/author_book.py** program has code,
  - and **tests/test_author_book.py** has the tests
- **You'll finish the code in lab_questions/author_book.py**
- Remember - don't modify the test code in tests/test_author_book.py



Running the lab code

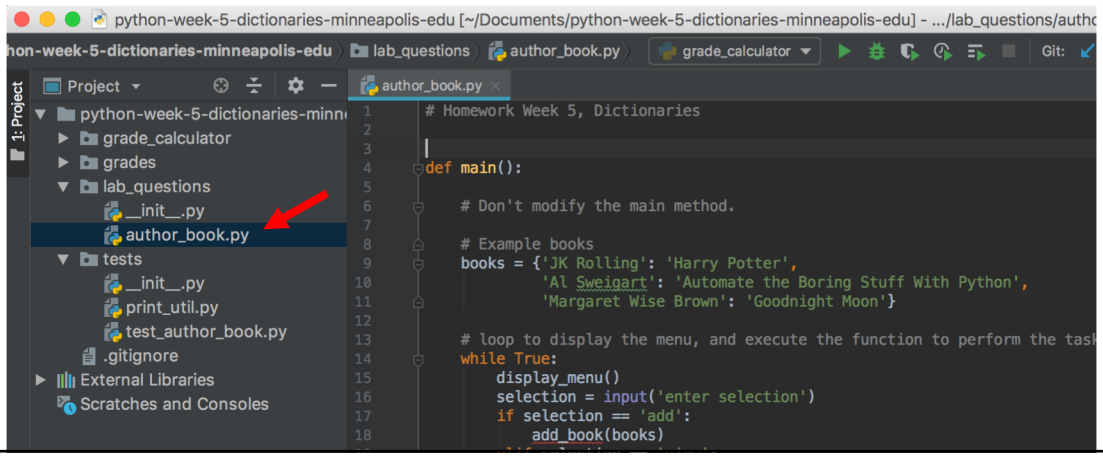
- Just like the hello.py program, the **author_book.py** program has code, and also tests in **tests/test_author_book.py**

- You'll finish the code in **lab_questions/author_book.py**
- Check your work by
    1. ALWAYS running your code and checking manually,
    2. AND running the tests in **tests/test_author_book.py**

It's import to do both! Why?
Hint: can the tests check for every aspect of your code working?
Do you have time to check for every detail every time you change anything?

Run the lab code

- Open the lab_questions/author_book.py file



Instructions

- For this lab, the instructions are in the **lab_questions/author_book.py** file
- Scroll down, and look for the yellow # TODO comments and notes
- # TODO comments will be highlighted in **blue**, if you are using the light color scheme
- TODO comments don't mean anything for your python program, but are useful for programmers

```
    print('add: Add new book')
    print('view: View all books')
    print('find: Find a title for an author')
    print('edit: edit the title for a book')
    print('del: delete a book')
    print('exit: quit program')
    print()


# Make sure you use the exact function names given!
# And, print the exact error messages requested.


# TODO write the add_book function here.
# This function should ask for an author and a title, and add these as a
# If the author is already in the dictionary, print "Already in dictionar


# TODO write the view_books function here. This function should print all
# This function should not ask for any input.
# Print the authors and titles together


# TODO write the find_book function here.
# This function should ask for an author.
# If the author is found in the dictionary, print the title of their book
# If the author is not found, print 'Not Found'


# TODO write the edit_books function here.
# This function should ask for an author, and the new book title.
# If the author is found in the dictionary, Modify the dictionary so the
# If the author is not found, print the string 'Not Found' and don't modi
```

lab_questions/author_book.py

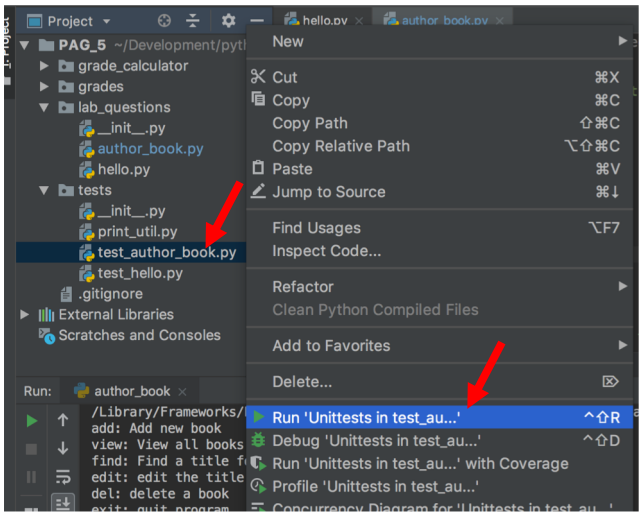
- Try running the code
- It will crash once you enter a menu selection
- Why? The functions it expects to call have not been written yet
- That's your task for this lab!

```
Run:  author_book
/Library/Frameworks/Python.framework/Versions/3.7/bin/python3.7 /Users/
add: Add new book
view: View all books
find: Find a title for an author
edit: edit the title for a book
del: delete a book
exit: quit program

enter selection: add
Traceback (most recent call last):
  File "/Users/admin/Development/python/AllOfThePythonLabs/PAG_5/lab_qu
    main()
  File "/Users/admin/Development/python/AllOfThePythonLabs/PAG_5/lab_qu
    add_book(books)
NameError: name 'add_book' is not defined

Process finished with exit code 1
```

tests/test_author_book.py

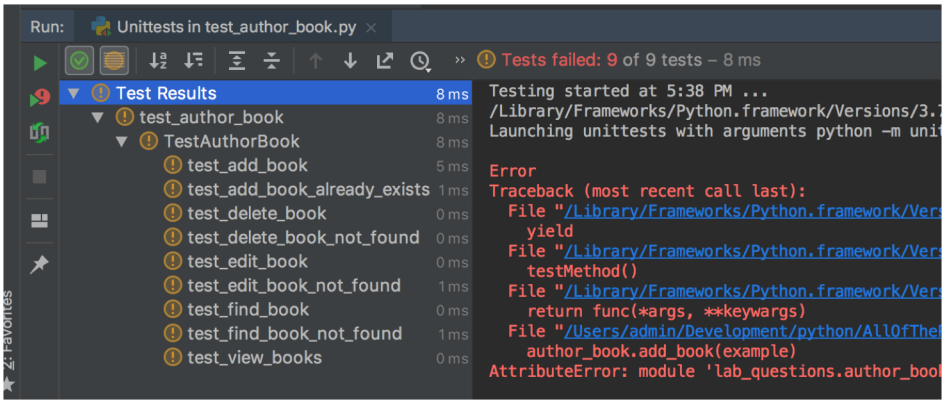
- Let's run the tests
- Open the tests directory, and right-click on **test_author_book.py**
- Select **Run Unittests in test_author_book.py**

- There are nine tests for this program



Test failures

- As expected, all 9 tests fail



Work on code

- Let's write the **add_book** function

- According to the assignment:

"This function should ask for an author, and a title, and add these as a new key:value pair to the books dictionary.
If the author is already in the dictionary, print "Already in dictionary" and don't modify the dictionary. Your program should not crash"

First attempt

- Write the **add_book** function in **author_book.py**

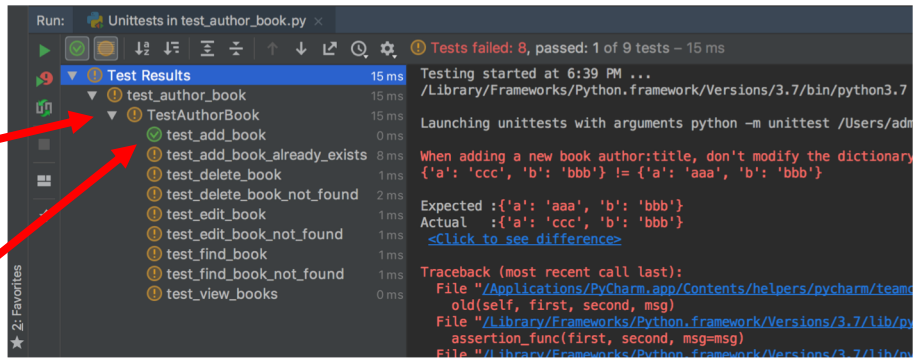
```
48
49
50      # TODO write the add_book function here.
51      # This function should ask for an author and a title, and add these
52      # If the author is already in the dictionary, print "Already in dict
53
54
55      def add_book(books):
56          author = input('Enter author name: ')
57          title = input('Enter title of book: ')
58          books[author] = title
59
60
61      # TODO write the view_books function here. This function should prin
62      # This function should not ask for any input.
```

Run tests in test_author_book.py

- The first test is passing!
- You may need to click on the triangle arrow by TestAuthorBook to see all the tests

Click on the triangle to expand the list of tests

First passing test for this question!



Run tests in test_author_book.py

- Click on the first failing test, **test_add_book_already_exists**
- Look at the errors message on the right
- What's wrong with the code?

Review the instructions

- "... If the author is already in the dictionary, print "Already in dictionary" and don't modify the dictionary. Your program should not crash"

Modify author_book.py add_book() function

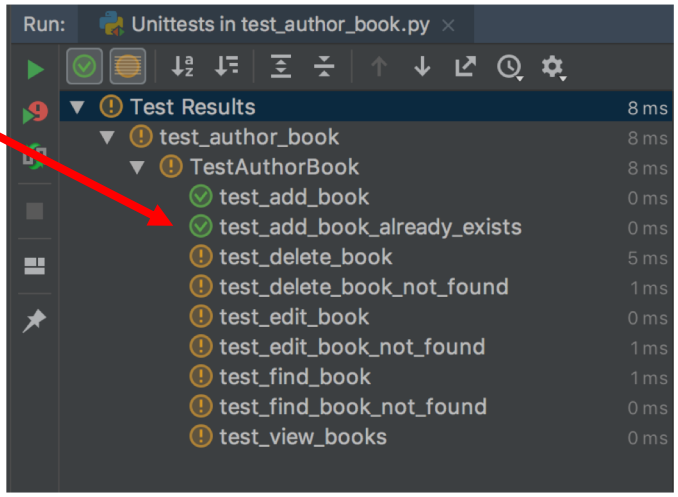
- Check to see if the author is already in the **books** dictionary before adding, and print message for the user

```
def add_book(books):
    author = input('Enter author name: ')
    title = input('Enter title of book: ')
    if author in books:
        print('Already in dictionary')
    else:
        books[author] = title
```

Run the tests again

- Two tests passing!
- **test_add_book_already_exists** is passing

- If not, make sure you print the exact message "Already in dictionary" because the test is looking for that exact output
- Check the logic of your if statement
- **Ask for help if it's still not working**



Help!

- If either/both of your **test_add_book** and **tests_add_book_already_exists** tests are failing:
  - Click on the failing test and look at the error message in the right panel - that should give you a clue
  - Make sure you print the exact message "Already in dictionary" because the test is looking for that exact output
  - Check the logic of your if statement
- **Ask for help if it's still not working**

Right logic, wrong message when author is already in the dictionary

```
def add_book(books):
    author = input('Enter author name: ')
    title = input('Enter title of book: ')
    if author in books:
        print('Book is in dictionary')
    else:
        books[author] = title
```

Error message indicates the problem

```
    raise self.failureException(msg)
AssertionError: 'already in dictionary' not found in 'book is in dictionary' : Print the message "Already in dictionary" if user tries to add an author that is al
```

Your turn!

- Finish the rest of the code
- Read the instructions in author_book.py and write the rest of the functions:
  - view_books
  - find_book
  - edit_book
  - delete_book
- Make sure your function names match the names used
- Read the instructions carefully so you know what is required

Your turn

- As you work on the lab, run your program and check that it's working
- Check it manually - run the code and verify it's working
- And run the tests and make sure the functions are working
- **Remember: it's important to do both to check your code!**

Finishing up - push to GitHub

- When you are finished, or it's the end of the lab, push your code to GitHub
- Click the commit icon 
- Write a descriptive commit message - what did you get done?
  - For example: "Finished author_book program"
  - or "completed add and edit functions"
- Hover over the Commit button and click on Commit and Push
- In the Push window, Click Push

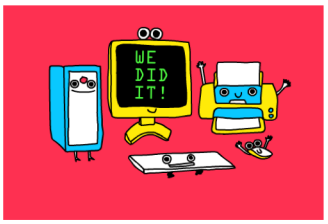
Finishing up - push to GitHub

- Go to your GitHub URL and verify your updates and code is there
- **Tell the instructor what your GitHub ID is so we can find your code**

Well done!

- That's a lot of new things!

- As a professional developer, you'll use GitHub, git, and testing at work
- You'll get a lot more practice in future programming classes
- So don't worry if you don't understand it all now - we covered a lot at a high level



- Questions? Want more detail? Please ask!
- And/or you can email me at clara.james@minneapolis.edu