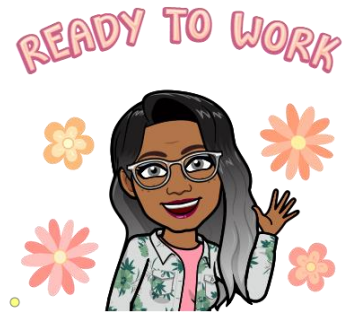
SEO Tech Developer Residency
Week 3: Integration Testing

July 13th, 2022

Presented by:
Dr. Sonia Mitchell

SEO Seizing Every Opportunity



Welcome

Office Hours

Mondays, Wednesdays, Thursdays & Fridays

9a.m. - 10:00a.m. EST

Email: Sonia.Mitchell@seo-usa.org

SEO Seizing Every Opportunity

Learning Objectives

At the end of this lesson, you should be able to:

- Describe integration testing
- Create an integration test and stub for an unimplemented module

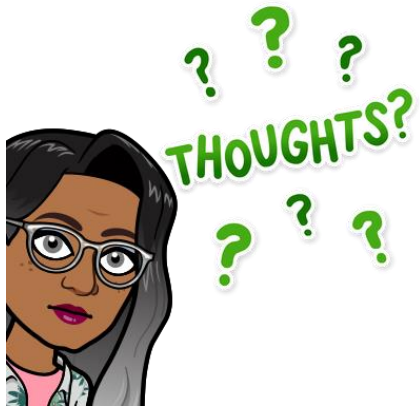
Question of Thought

What do you think Integration Testing could be?

Drop your guesses in the chat!

Or

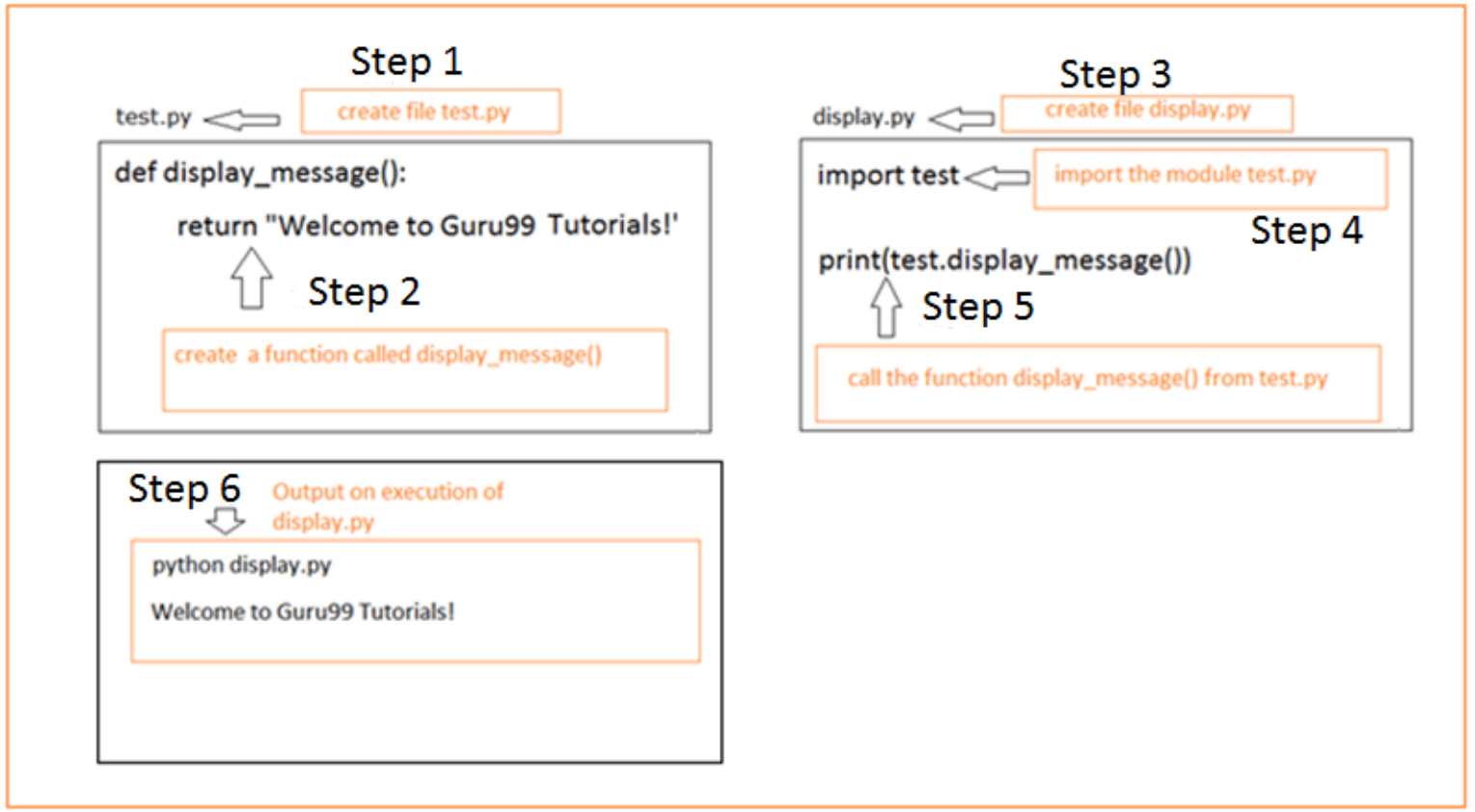
Unmute Quickly, begin to share



What is Integration Testing?

- Integration tests check the interaction **between** modules.
  - **What's a module**?

In Python, Modules are simply **files with the ". py" extension containing Python code that can be imported inside another Python Program**. In simple terms, we can consider a module to be the same as a code library or a file that contains a set of functions that you want to include in your application.



What is integration testing?

- Integration tests check the interaction **between** modules.
  - **What's a module**?

- **SURPRISE!!! You've actually already written an integration test!**
  - When you tested whether your website forms worked – you were testing your main application module, the forms module, the HTML code, and the database connection.
  - For the most part, integration tests are written using the same libraries as unit tests – the big difference is how much of the code is covered in each test.

Why do we care about integration tests?

- If we unit test everything, we should be good right?
- Integration tests are important because:
  - Even if one developer builds an entire module (meaning it passes unit tests), they might have a different understanding than the developers building interacting modules
  - Interfaces between technology, such as modules and a database, are error-prone
  - Lack of handling of exceptions can cause interactions to crash in ways that cannot be discovered in unit testing

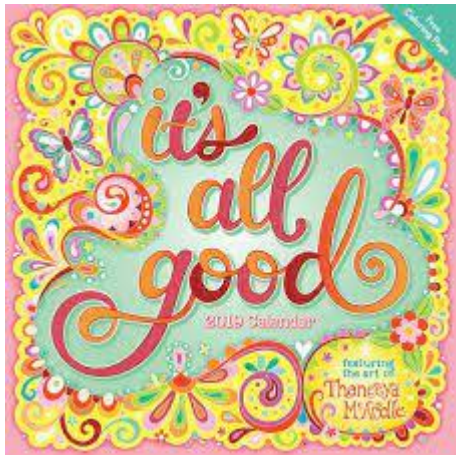
Bottom-up vs Top-Down Integration Testing

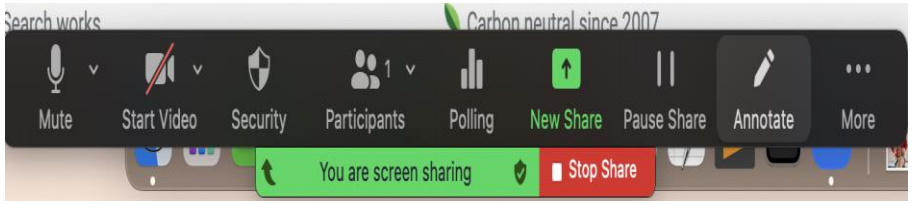
- **Bottom-up integration testing** starts by testing the modules without dependencies on other modules. This makes fault localization easier which is an advantage, but critical, top-level modules are tested last.

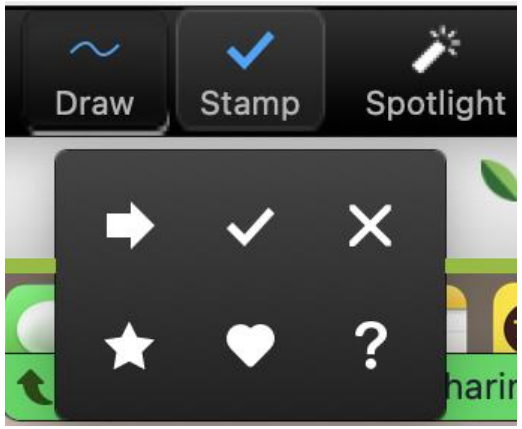
- **Top-down integration testing** follows the control flow of the system. Critical, top-level modules are tested first, which is an advantage, but it requires a lot of stubbing.

LEARNING TEMPERATURE CHECK

Add a stamp to how you are feeling about the lesson so far. You can access the stamp tool by clicking *Annotations* in the top Zoom toolbar, then selecting a stamp.







Testing Terminology Refresher

- A **mock** is a **fake object** that mimics an actual object
- A **stub** can **replace an object** that isn't built yet
  - A stub will **never fail** a unit test, but a mock can.
  - A stub **could be replaced** when the functionality is added.
- A **fake** can refer to either a mock or a stub - any piece of code that is pretending to be fully implemented, production code

Stubs

- There are generally **2 times you stub code out**.
- The first is when you need classes and methods to exist for syntactical correctness, but they don't need to do anything.
- You can handle these cases with the **`pass`** keyword
  - **Docs**: https://docs.python.org/3/tutorial/controlflow.html?highlight=pass#pass-statements

In the docs, you see a couple of examples of minimal classes and functions:

```
class FutureClass:
    pass
```

```
def future_function(*args):
    pass
```

Notice the `*args` which allows you to accept an unspecified number of arguments.

Stubs (cont.)

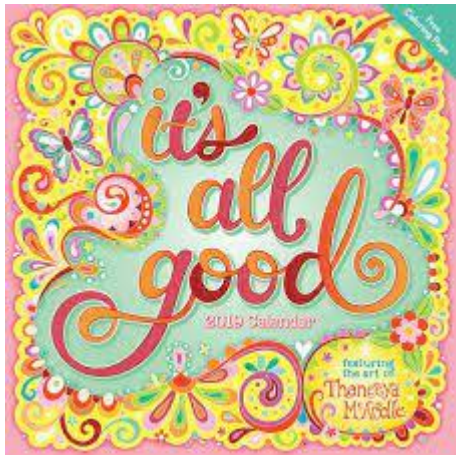
The second type of stubbing requires your yet-to-be-implemented code to do something. In this case, since you know the test cases, you write the minimal code needed to pass the tests. This often looks like a series of conditionals:

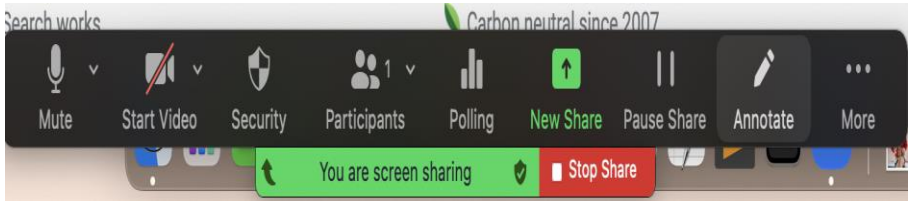
```
def fibonacci(n):
  if n == 1:
    return 1
  elif n == 6
    return 8
  else:
    return 233
```

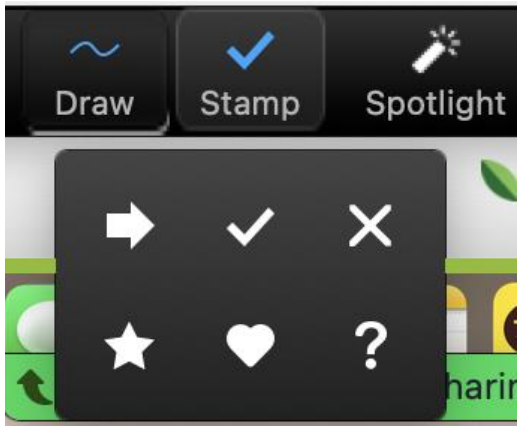
In the example above, because I am writing the test cases, I know that this function will only be tested on this subset of values.

LEARNING TEMPERATURE CHECK

Add a stamp to how you are feeling about the lesson so far. You can access the stamp tool by clicking *Annotations* in the top Zoom toolbar, then selecting a stamp.







Mocks

Mocking is generally useful during unit testing so that external dependencies are no longer a constraint to the unit under test. Often those dependencies may themselves be under development. Without mocking, if a test case fails, it will be hard to know if the failure is due to our code unit or due to dependencies. Mocking therefore speeds up development and testing by isolating failures.*

Other reasons to mock dependencies is to avoid slow network calls or call third-party APIs. Mocking also enables product demos and evaluations. All units of a project can progress in parallel without waiting for everyone to be ready. Thus, testing can start early.*

Code that have side effects should be called only in production. Examples include charging a credit card or sending a notification. Mocking is useful to validate such calls without the side effects.*

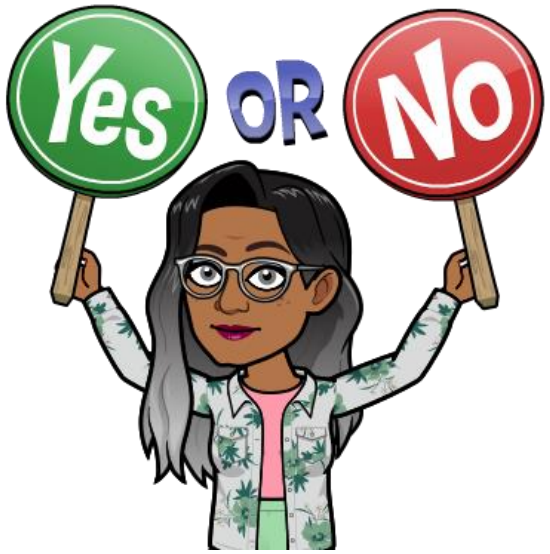
Did we meet our Learning Objectives(LOs)?

Students Will Be Able To

1.Describe integration testing

2.Create an integration test and stub for an unimplemented module

Drop your answers in the chat!



Today's Activity

- Expand your **Web Framework** program by picking something related to testing that is interesting and useful for your project!

- You'll see a few suggestions on the activity page.

Thank You!

Q&A

Office Hours

Mondays, Wednesdays, Thursdays & Fridays

9a.m. - 10:00a.m. EST

Email: Sonia.Mitchell@seo-usa.org

SEO Seizing Every Opportunity