A bit more on Testing

Admin

Some thoughts on the project

- You are reading data from the web.
  - If you know knowing about http 1.x read at least about get and put
  - Maybe here: https://code.tutsplus.com/tutorials/a-beginners-guide-to-http-and-rest--net-16340
  - Stackoverflow will return xml/structured text
  - Beautiful soup (bs4) or jsoup are good choices to parse
  - Lets look at an example of the format
    - SOQuery = 'https://stackoverflow.com/jobs/feed?l=Boston%2c+MA%2c+United+States&u=Miles&d=50'
    - webResponse = requests.get(SOQuery)
    - print (webResponse.text)

Some suggestions

- In python
  - use the requests package to get the data from the url
  - Then once you tell the result to convery to json you will have a big list with embedded lists in it. Dig through the list manually to find its structure and then extract the bits that you need programmatically.

In Java

- For getting the data, not as much of a one true way – maybe HttpURLConnection
- XML
  - Builtin DOM parser/builder
  - DocumentBuilder
  -

Lets look at an unrelated example

- Alpha vantage (https://www.alphavantage.co/) sells stock info apis
  - Make a sample one available to demo
  - https://www.alphavantage.co/documentation/
  - https://www.alphavantage.co/query?function =TIME_SERIES_DAILY_ADJUSTED&symbol=MSF ikey=demo
  - Returns json with historical data
  -

Hint: be nice

- Under no circumstances should you ever run your http request in a loop!!?!?!!
  - Sites will think you are attacking them and ban your IP
  - You will be sad
  - Bright students have done so in the past.

Lets look at an example in python

- import requests
- Import bs4 as BeautifulSoup
- def demo_json():
-     sample_query = 'https://www.alphavantage.co/query?function=TIME_SERIES_DAILY_ADJUSTED&symbol=MSFT&apikey=demo'
-     web_response = requests.get(sample_query)
-     data = BeautifulSoup(web_response.test)
-     print(data)
-
- Put this python program into your favorite editor and run it. What do you see? (you can grab it off the class website and paste it in)
- How would we grab data from yesterday?

User stories

- A short simple description for a feature from the perspective of a user
  - Eg for an sports betting app:
    - As a user I want to login so that I can access my account
    - As a user I want to place a bet.
    - As a customer service person I want to access an account so I can resolve a complaint
  - General form:
    - As a <type of user> I want to <goal> [so that I can <reason>]
- Usually start our general and then get more specific as you go.

Use Cases

- Use case
  - Describes complete interaction between user and software
    - Or system
  - Is about the behavior that will be built to meet needs
  -

User Stories vs Use Cases

- User stories describe a need
- Use cases describe the behavior to meet that need.
- Perhaps we’ll look at examples later in the class.

Automated Tests

- Automated tests should be executable specifications
  - If the user story says that someone should be able to do X with your code
  - Test that your code does X
  - And Test that it does not do not X
  - And Test that it does not have security errors while doing X
  - And Test that it fails gracefully when the resources needed for X are missing

Tests

- When you are building your tests
  - Try taking the URL as a parameter
  - Then call your function with good and bad – make sure that your code handles response codes other than 200
    - My 4 line example did not.
  - Make sure that you try both good input and get good results
  - And lots of bad input and make sure your program doesn’t barf.

An Example

- I'll work through a testing example using pycharm and pytest
  - If you are using java probably use junit
  - And intellij
  - It should work very similarly.

Install

- Make sure that pytest is installed
  - Pip3 install pytest
  - Or go to your pycharm project interpreter and add the package.

Tell pycharm about pytest

- In pycharm choose settings
  - Then open the tools option and choose python integrated tools.
  - In the default test runner option choose py.test
  - Choose ok and close the dialog.

Best Practices

- For best practices,
  - Have a separate test directory
  - Create a new directory as a subdirectory in your project
  - Lets call it tests.

Lets write a simple test

- In your tests directory in your new test file
  - Make sure it begins with test_
  - Import your real file
  - Import pytest
  - Write a function called
  - test_your_real_function
  - Replace your_real_function with its actual name of course.
  - Use an assert statement in your test function to test something about your production function

Write tests to test for errors

- Also write tests that pass bad data
  - See if your code handles it gracefully.

Lets see it work

- Demo with two tests
    - One that passes
    - One that doesn’t