Kishan Patel (kishanlp) and Aliya Khan (khanal)
EECS 481: Software Engineering
16 Apr 2018

HW6b: Contribution

I. Name and Email Ids

Kishan Patel (kishanlp) and Aliya Khan (khanal)

II. Selected Project

We contributed to an open-source messaging application called Zulip. The Zulip website URL is https://zulipchat.com/. The Github project can be found at https://github.com/zulip/zulip. Zulip is a messaging app similar to Slack, and it is mainly geared towards larger communities such as organizations or companies that include many people. The Zulip project is highly trafficked, with developers responding to my questions within hours. Zulip itself is organized into several different smaller projects, one of which is the main Zulip app, and another being the Python Zulip API which includes implementations of several bots that can be run within the Zulip app. For our contribution, we were initially going to contribute to a bug in the main Zulip app, but when this proved too time-consuming, we switched to focusing on tests for the bots in the Python Zulip API.

III. Project Context

Zulip's project context and 'business model' involves a Slack-like interface that allows users who haven't been active for a while to easily catch up on previous conversations. Its open-source competitors include Mattermost, Rocket.chat, and matrix.org, and its main closed-source competitor is Slack. Zulip aims to create a chat application for large communities. On its website, it states "Zulip combines the immediacy of Slack with an email threading model. With Zulip, you can catch up on important conversations while ignoring irrelevant ones" (Zulip). Its main goals are increasing productivity and creating more streamlined updates for community members. While contributing to the project, we had to use Zulip to connect with other contributors. Even after being unable to work on the project for five days, we were very easily able to catch up with the messages sent in the chat and stay updated about information that was relevant to us. This relative ease of use after a period of time being uninformed made the use of Zulip much more painless than Slack, and made it something that we would both consider using for large group chats in the future.

IV. Project Governance

The project uses its own messaging application in order to communicate among contributors. This process is slightly structured in that contributors must use specific channels within the Zulip messaging application that relate the most to the issue that they are contributing to. Additionally, Zulip has a very structured approach to identifying and assigning issues to certain developers, which is outlined below:

A. First, the developer must find an issue with the tag “help wanted”, which indicates that this issue is “up for grabs”, but without the tag “in progress”, which means that someone else has already claimed this issue and is working on it. Additionally, first-time contributors may only claim one issue at a time.

B. After finding an issue, a contributor must comment “@zulipbot claim” on the issue. They will then have 14 days to either submit a pull request or reference it in the GitHub comments for that issue before zulipbot will automatically unassign them from the issue.

C. While fixing the issue, if the developer runs into hurdles, cannot build the program, or has any questions about the code already in the codebase, he or she may ask on the Zulip chat. In this chat, there are multiple channels, similar to Slack, where the developer can ask questions in a channel specific to the issue that the developer is fixing. For example, in our contribution, we used the “test-suite” channel to ask any questions to other developers or admins that may be familiar with the codebase in Zulip. This communication with other developers significantly aided us in being able to make a contribution to Zulip.

D. There are several QA guidelines for Zulip based on which codebase contributors write to, in our case, we were writing to python-zulip-api. For this, we had to first build a virtualenv where we could run the test cases necessary. Then, each time we made a change in the code, we would run the tests locally to make sure they passed before committing them in our pull request.

E. Zulip also has guidelines for new contributors, which are **below in the link**. It emphasizes describing *why* a change was made in prose, which is similar to what was emphasized in lectures regarding similar material, especially requirements elicitation. At first, we did not know about these guidelines and ended up having to change our original commit message in order to fit them before our pull request could be accepted. In addition, the link provides information on how to communicate with other developers or contributors in order to receive help.
Link: https://zulip.readthedocs.io/en/latest/overview/contributing.html

F. After submitting the pull request, Travis CI integration tests are performed on the changes to make sure that they are compatible with the rest of the codebase.

In terms of requirements, while we did know the bare expectations of what the end goal was(to reach 100% coverage), the requirements were still vague about other quality requirements of the test-cases, such as size, speed, and readability. There were no requirements pertaining to design for the specific test-coverage issue that we selected. In addition, although the Zulip project does have guidelines for code style and overall readability of code, there is no overarching design standard that the project uses.

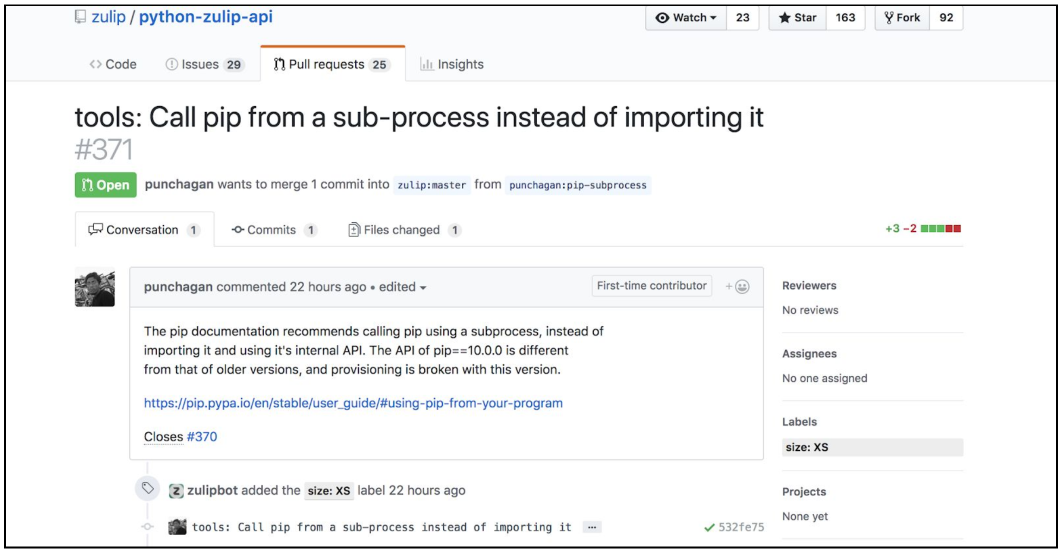
Quality assurance for this project consists of making every contributor pass integration tests before a pull request is passed. This method ensures that any changes that a contributor has made are compatible with the rest of the subsystems and subcomponents of the Zulip project. After our task is completed, we would have to run tests locally in order to test the zulip_bots, but we would also have to run the integration tests on Travis CI. After a pull request is passed on Travis CI, it may be eligible for being merged with master on GitHub. This process significantly decreases the likelihood of a bug or defect being introduced into the codebase by a contributor that may have tested locally inadequately.

V. Task Description

TicTacToe Bot Tests

The task that we were able to officially submit a PR for was a testing task involving increasing test coverage for one of the bots. This bot, called tictactoe.py, allows the user to play tictactoe with the zulipbot through the application. The original test coverage for tictactoe.py was 19%, and we were able to increase it slightly to 20%. However, our main focus when working with this code turned out to be less focused on increasing coverage and more focused on creating a non-buggy implementation of the test file and a precedent for future tests. We figured out that the way that the tests were initially set up initialized the tictactoe board using a function called _get_game_handlers(), which created a tictactoe board model for testing. However, instead of creating an instance of the class, we figured out that it was referencing the class itself, leading to confusion whenever the tests were run. After we were able to fix this issue, with help from one of the developers on the Zulip chat, we had time to create and run several test cases which tested the basic functionality of the bot, which were eventually merged through our pull request.

Call pip from a sub-process instead of importing it (Issue #371)


zulip / python-zulip-api

Watch 23 | Star 163 | Fork 92

Code | Issues 29 | Pull requests 25 | Insights

tools: Call pip from a sub-process instead of importing it #371

Open punchagan wants to merge 1 commit into `zulip:master` from `punchagan:pip-subprocess`

Conversation 1 | Commits 1 | Files changed 1 | +3 −2

punchagan commented 22 hours ago • edited · First-time contributor

The pip documentation recommends calling pip using a subprocess, instead of importing it and using it's internal API. The API of pip==10.0.0 is different from that of older versions, and provisioning is broken with this version.

https://pip.pypa.io/en/stable/user_guide/#using-pip-from-your-program

Closes #370

zulipbot added the size: XS label 22 hours ago

`tools: Call pip from a sub-process instead of importing it` ✓ 532fe75

Reviewers: No reviews

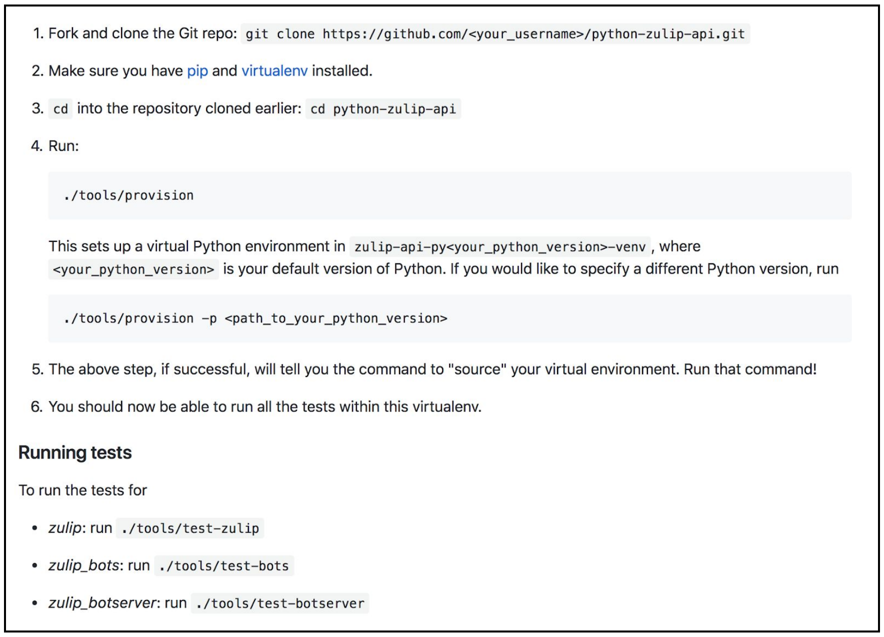
Assignees: No one assigned

Labels: size: XS

Projects: None yet

In addition to working on the test cases for tictactoe.py, we were able to identify and help fix a bug that was not in an official issue request that was creating issues for the entire build of the python-zulip-api. This bug ended up being the focus of the majority of our time, but unfortunately due to the Project Governance, we were unable to submit a PR for this issue. This issue involved directions of how to clone and run tests for the bots (Image 1). After trying to run these simple commands for several days on multiple versions of Mac OSX, Windows, and Ubuntu, and attempting multiple fixes, some of which are shown below, we figured out that there was an issue in the actual code for tools/provision that was preventing us from running the virtual environment. Essentially, each time we tried to run the virtual environment we would receive the error message “pip has no attribute main” (Image 2). While attempting to fix this issue, we were able to find a suggestion through Google that implied that pip version 10.0.0 broke some of the dependencies within the pip library which made it unable to find pip.main. However, we tried changing versions of pip on multiple systems to 9.0.3 and it still did not fix the issue. Finally, we looked into the code for tools/provision and figured out that the file was manually overriding our version of pip with a version above 9.0, therefore forcing the run of pip 10.0.0. After fixing this issue locally so that we could build and run the virtual environment on our local system, we brought it up to the main contributors who then created a new issue request for this to be fixed, and were able to fix it globally throughout the project (we could not contribute to this issue directly because we had already claimed another issue and according to the Project Governance we could only claim 1 issue).

Image 1



1. Fork and clone the Git repo: `git clone https://github.com/<your_username>/python-zulip-api.git`
2. Make sure you have pip and virtualenv installed.
3. `cd` into the repository cloned earlier: `cd python-zulip-api`
4. Run:

   ```
   ./tools/provision
   ```

   This sets up a virtual Python environment in `zulip-api-py<your_python_version>-venv`, where `<your_python_version>` is your default version of Python. If you would like to specify a different Python version, run

   ```
   ./tools/provision -p <path_to_your_python_version>
   ```

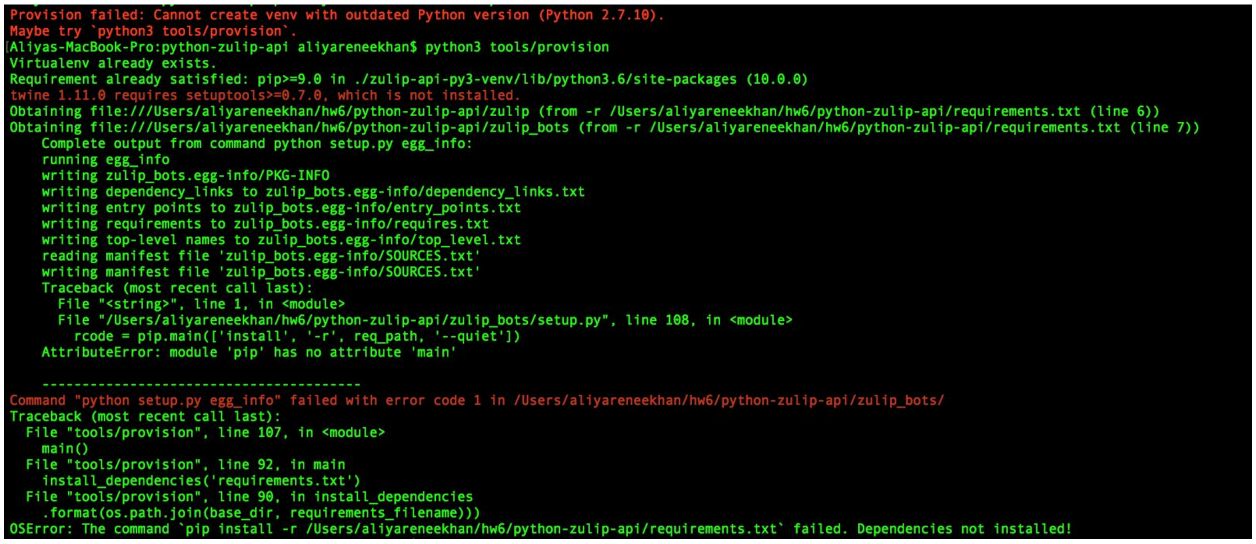
5. The above step, if successful, will tell you the command to "source" your virtual environment. Run that command!
6. You should now be able to run all the tests within this virtualenv.

Running tests

To run the tests for

- *zulip*: run `./tools/test-zulip`
- *zulip_bots*: run `./tools/test-bots`
- *zulip_botserver*: run `./tools/test-botserver`

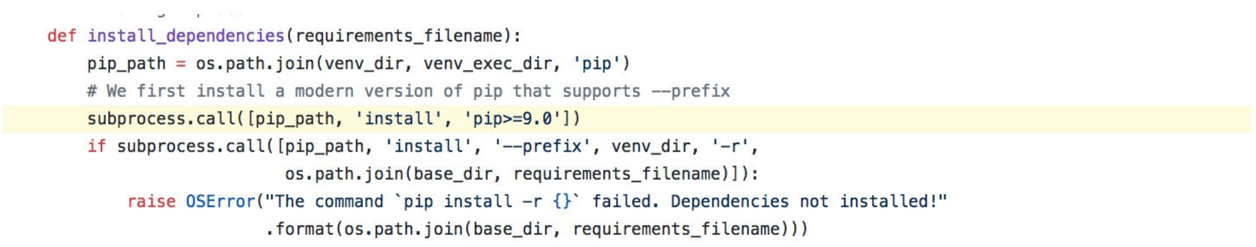
Image 2



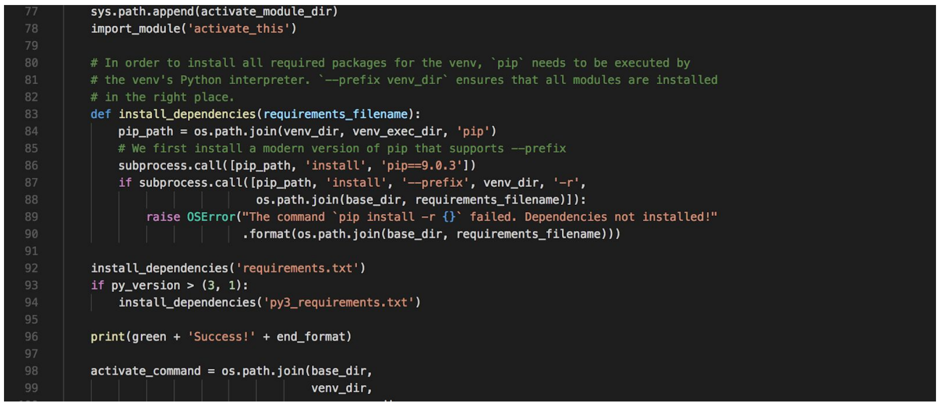
```
Provision failed: Cannot create venv with outdated Python version (Python 2.7.10).
Maybe try `python3 tools/provision`.
[Aliyas-MacBook-Pro:python-zulip-api aliyareneekhan$ python3 tools/provision
Virtualenv already exists.
Requirement already satisfied: pip>=9.0 in ./zulip-api-py3-venv/lib/python3.6/site-packages (10.0.0)
twine 1.11.0 requires setuptools>=0.7.0, which is not installed.
Obtaining file:///Users/aliyareneekhan/hw6/python-zulip-api/zulip (from -r /Users/aliyareneekhan/hw6/python-zulip-api/requirements.txt (line 6))
Obtaining file:///Users/aliyareneekhan/hw6/python-zulip-api/zulip_bots (from -r /Users/aliyareneekhan/hw6/python-zulip-api/requirements.txt (line 7))
    Complete output from command python setup.py egg_info:
    running egg_info
    writing zulip_bots.egg-info/PKG-INFO
    writing dependency_links to zulip_bots.egg-info/dependency_links.txt
    writing entry points to zulip_bots.egg-info/entry_points.txt
    writing requirements to zulip_bots.egg-info/requires.txt
    writing top-level names to zulip_bots.egg-info/top_level.txt
    reading manifest file 'zulip_bots.egg-info/SOURCES.txt'
    writing manifest file 'zulip_bots.egg-info/SOURCES.txt'
    Traceback (most recent call last):
      File "<string>", line 1, in <module>
      File "/Users/aliyareneekhan/hw6/python-zulip-api/zulip_bots/setup.py", line 108, in <module>
        rcode = pip.main(['install', '-r', req_path, '--quiet'])
    AttributeError: module 'pip' has no attribute 'main'

    ----------------------------------------
Command "python setup.py egg_info" failed with error code 1 in /Users/aliyareneekhan/hw6/python-zulip-api/zulip_bots/
Traceback (most recent call last):
  File "tools/provision", line 107, in <module>
    main()
  File "tools/provision", line 92, in main
    install_dependencies('requirements.txt')
  File "tools/provision", line 90, in install_dependencies
    .format(os.path.join(base_dir, requirements_filename)))
OSError: The command `pip install -r /Users/aliyareneekhan/hw6/python-zulip-api/requirements.txt` failed. Dependencies not installed!
```

VI. Submitted Artifacts

Below are screenshots of some of the code that we submitted in our PR. These are very simple test cases, but required extensive amounts of work on dependencies and involved the discovery that the initial implementation of the test cases was buggy. Therefore, we were not able to get the test coverage to 100% but we were able to smooth out the test cases enough to create a precedent for future test cases that could be easily followed. Finally, I have also included a screenshot of the initial bug that we had to solve involving the version of pip, because this is what we spent most of our time discovering and fixing. The following image (Image 1) is pertaining to the bug in tools/provision file at line 86 that prevented us from running Zulip Python API for a long time.



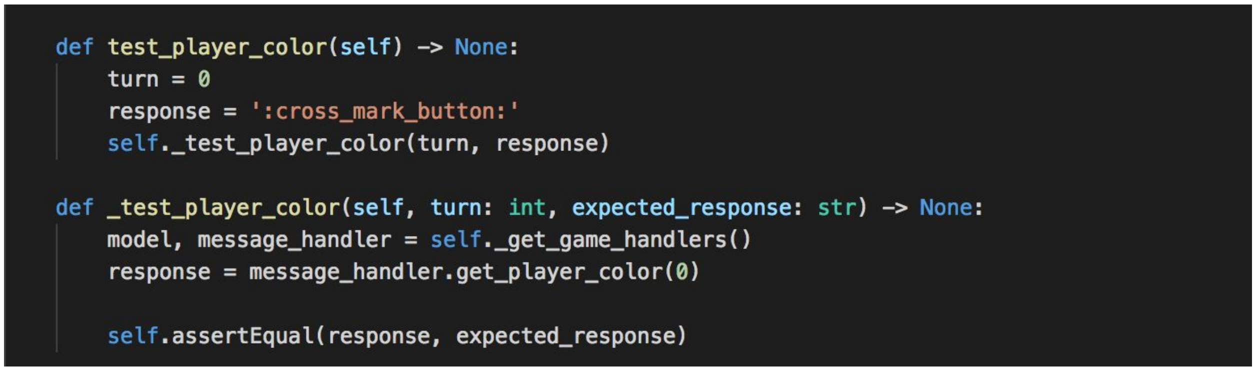
```
def install_dependencies(requirements_filename):
    pip_path = os.path.join(venv_dir, venv_exec_dir, 'pip')
    # We first install a modern version of pip that supports --prefix
    subprocess.call([pip_path, 'install', 'pip>=9.0'])
    if subprocess.call([pip_path, 'install', '--prefix', venv_dir, '-r',
                        os.path.join(base_dir, requirements_filename)]):
        raise OSError("The command `pip install -r {}` failed. Dependencies not installed!"
                      .format(os.path.join(base_dir, requirements_filename)))
```



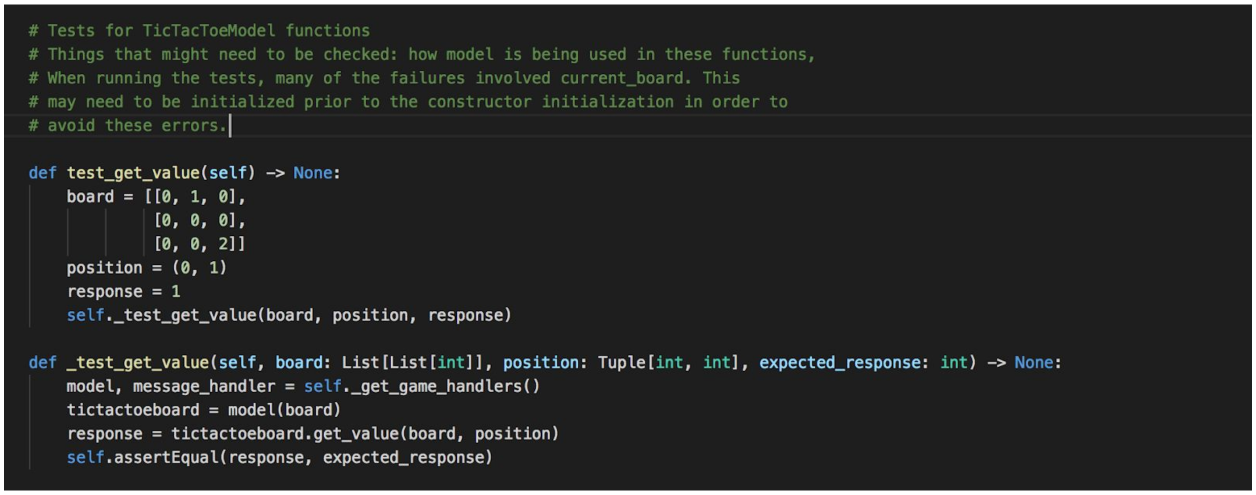
```
77      sys.path.append(activate_module_dir)
78      import_module('activate_this')
79
80      # In order to install all required packages for the venv, `pip` needs to be executed by
81      # the venv's Python interpreter. `--prefix venv_dir` ensures that all modules are installed
82      # in the right place.
83      def install_dependencies(requirements_filename):
84          pip_path = os.path.join(venv_dir, venv_exec_dir, 'pip')
85          # We first install a modern version of pip that supports --prefix
86          subprocess.call([pip_path, 'install', 'pip==9.0.3'])
87          if subprocess.call([pip_path, 'install', '--prefix', venv_dir, '-r',
88                              os.path.join(base_dir, requirements_filename)]):
89              raise OSError("The command `pip install -r {}` failed. Dependencies not installed!"
90                            .format(os.path.join(base_dir, requirements_filename)))
91
92      install_dependencies('requirements.txt')
93      if py_version > (3, 1):
94          install_dependencies('py3_requirements.txt')
95
96      print(green + 'Success!' + end_format)
97
98      activate_command = os.path.join(base_dir,
99                                      venv_dir,
```

Image 2: This is the fixed code at line 86 where the “pip>=9.0” is replaced with “pip==9.0.3”. This was causing only pip version 10.0.0 to be run which happens to be incompatible with the pip.main() package that Zulip uses.

Images 3-4: Test functions that we wrote inside of the test_tictactoe.py file and committed to the Zulip codebase.



```
def test_player_color(self) -> None:
    turn = 0
    response = ':cross_mark_button:'
    self._test_player_color(turn, response)

def _test_player_color(self, turn: int, expected_response: str) -> None:
    model, message_handler = self._get_game_handlers()
    response = message_handler.get_player_color(0)

    self.assertEqual(response, expected_response)
```


```python
# Tests for TicTacToeModel functions
# Things that might need to be checked: how model is being used in these functions,
# When running the tests, many of the failures involved current_board. This
# may need to be initialized prior to the constructor initialization in order to
# avoid these errors.

def test_get_value(self) -> None:
    board = [[0, 1, 0],
             [0, 0, 0],
             [0, 0, 2]]
    position = (0, 1)
    response = 1
    self._test_get_value(board, position, response)

def _test_get_value(self, board: List[List[int]], position: Tuple[int, int], expected_response: int) -> None:
    model, message_handler = self._get_game_handlers()
    tictactoeboard = model(board)
    response = tictactoeboard.get_value(board, position)
    self.assertEqual(response, expected_response)
```

Link to Public Repository of Python-Zulip-API: https://github.com/zulip/python-zulip-api

Integration Testing Link: The link for the Pull Request on Travis CI is as follows. After clicking the link, you will be able to see our commit message as well as the test-suites that we passed. Link: https://travis-ci.org/zulip/python-zulip-api/builds/367024408

Link to first issue pertaining to test-coverage:
https://github.com/zulip/python-zulip-api/issues/122

VII. QA Strategy

With regards to QA, the part of the code that we were writing was part of the QA process for the zulipbot, because we were attempting to increase coverage for the zulipbot. Additionally, we discovered three bugs in the zulip code through our attempts to increase the test coverage. Finally, towards the end of our testing, we had to submit our code to review through Travis CI for integration tests and had to correct many PEP and lint errors found through a static analysis of the code. In order to make test cases that would increase the coverage in tictactoe.py, we made tictactoe “boards” in the test case and passed in the boards as arguments to the functions we were testing in the tictactoe module. Then, we would assert our expected final answer or ‘test oracle’ to the output of the function and check to see if the output was correct. Initially, we found that the function _get_game_handlers was returning the model class instead of an instance of the class. In order to work around this bug that was already in the tictactoe.py file, we had to instantiate an object of the model() class every time we wanted to test a function that required a model of the board.

In addition to fixing or working around any bugs that may already exist in the zulip_bots codebase before we began working on Zulip, we also had to consider maintainability as a QA strategy. Following the coding style and guidelines of the Zulip project would be required in order to pass all Travis CI pull request tests and for our contribution to be accepted into the codebase. Specifically, we made sure to comment our test cases and follow any linting, spacing, or character guidelines for each function we added. After writing the test cases and making sure that they follow the style guidelines of Zulip, we performed unit testing by running only the test_tictactoe.py test cases on tictactoe.py. We were able to reveal any bugs in the test_tictactoe.py that was written prior to us working on it through unit testing because some of the test case functions written by previous contributors included a few bugs including a bug where a class was being returned from _get_game_handler instead of an instance of the class. In order to make this unit testing efficient, we made sure to make each test small so that if an assertion in a specific test case failed, we would know exactly what the bug was. For example, we would test 1 functionality in each test_function which would allow us to pinpoint any bugs should any test cases fail.

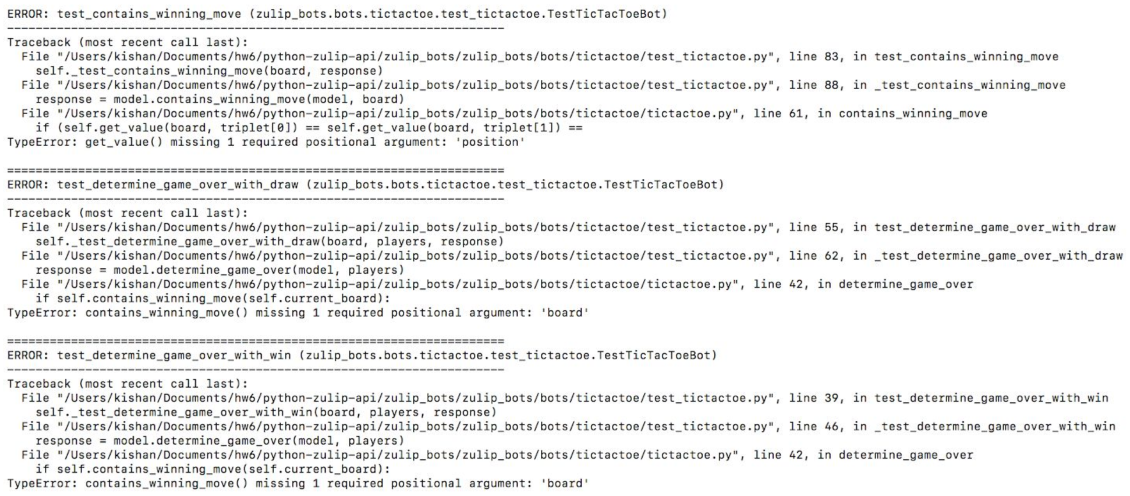
After performing unit testing, we performed integration testing using Travis CI and we would wait until our pull request would pass all integration tests. This step would ensure that the changes/commits we made would not cause the rest of the Zulip codebase to fail.

To generalize the QA strategy, our approach included using line coverage in Python, code maintainability and style guidelines, unit testing, and integration testing to ensure that our contribution would not introduce any new bugs into the Zulip codebase and would successfully test the existing codebase. Relating this back to lecture concepts, line coverage is a form of dynamic analysis and code maintainability relates to static analysis where code execution is not performed.

VIII. QA Evidence

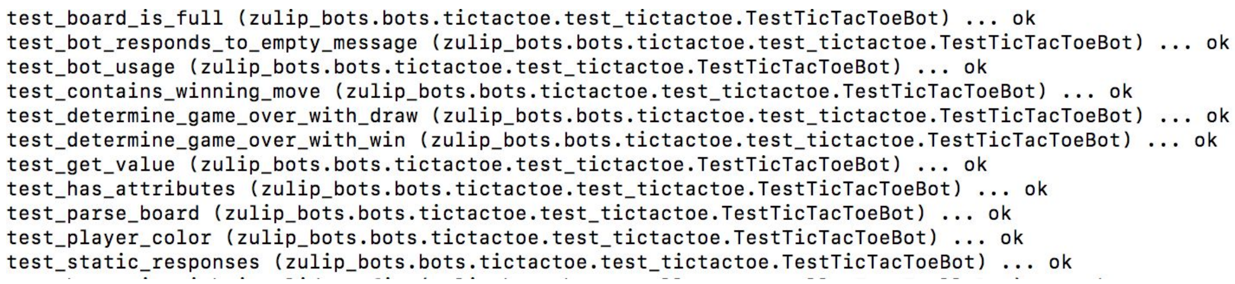
The following evidence includes evidence of the actual changes in code, the line coverage testing, and integration testing using Travis CI. The evidence also includes screenshot of the bug we ran into regarding the issue in tools/provision that prevented running pip version 9.0.3. It also includes an example of static analysis of code by using a python linter. Lastly, it includes proof of running the dynamic analysis tool coverage.py for tictactoe.py.

Image 1: the initial runs of the functions that we wrote failed because of a bad implementation of the object _game_handler(), which we were able to identify and work around.



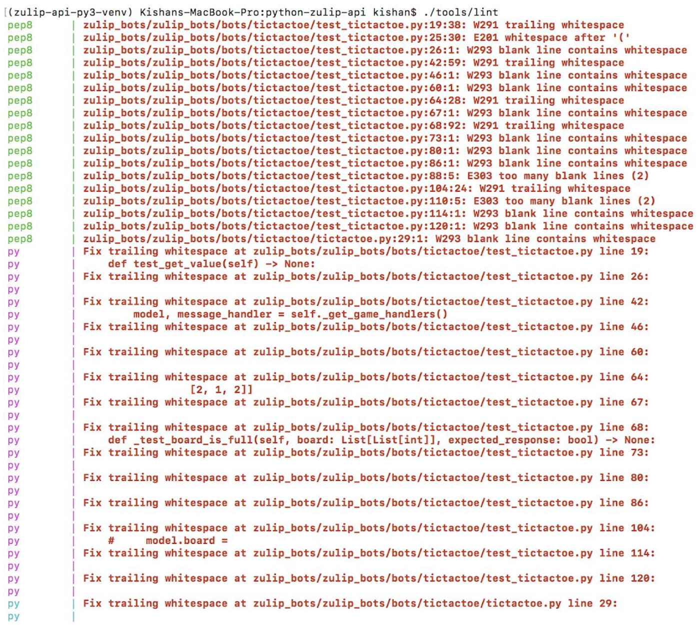
```
ERROR: test_contains_winning_move (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py", line 83, in test_contains_winning_move
    self._test_contains_winning_move(board, response)
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py", line 88, in _test_contains_winning_move
    response = model.contains_winning_move(model, board)
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/tictactoe.py", line 61, in contains_winning_move
    if (self.get_value(board, triplet[0]) == self.get_value(board, triplet[1]) ==
TypeError: get_value() missing 1 required positional argument: 'position'

======================================================================
ERROR: test_determine_game_over_with_draw (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py", line 55, in test_determine_game_over_with_draw
    self._test_determine_game_over_with_draw(board, players, response)
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py", line 62, in _test_determine_game_over_with_draw
    response = model.determine_game_over(model, players)
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/tictactoe.py", line 42, in determine_game_over
    if self.contains_winning_move(self.current_board):
TypeError: contains_winning_move() missing 1 required positional argument: 'board'

======================================================================
ERROR: test_determine_game_over_with_win (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py", line 39, in test_determine_game_over_with_win
    self._test_determine_game_over_with_win(board, players, response)
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py", line 46, in _test_determine_game_over_with_win
    response = model.determine_game_over(model, players)
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/tictactoe.py", line 42, in determine_game_over
    if self.contains_winning_move(self.current_board):
TypeError: contains_winning_move() missing 1 required positional argument: 'board'
```

Image 2: passing runs of the tests after working around the bad implementation of _game_handler() - ok indicates that the test successfully ran with the codebase



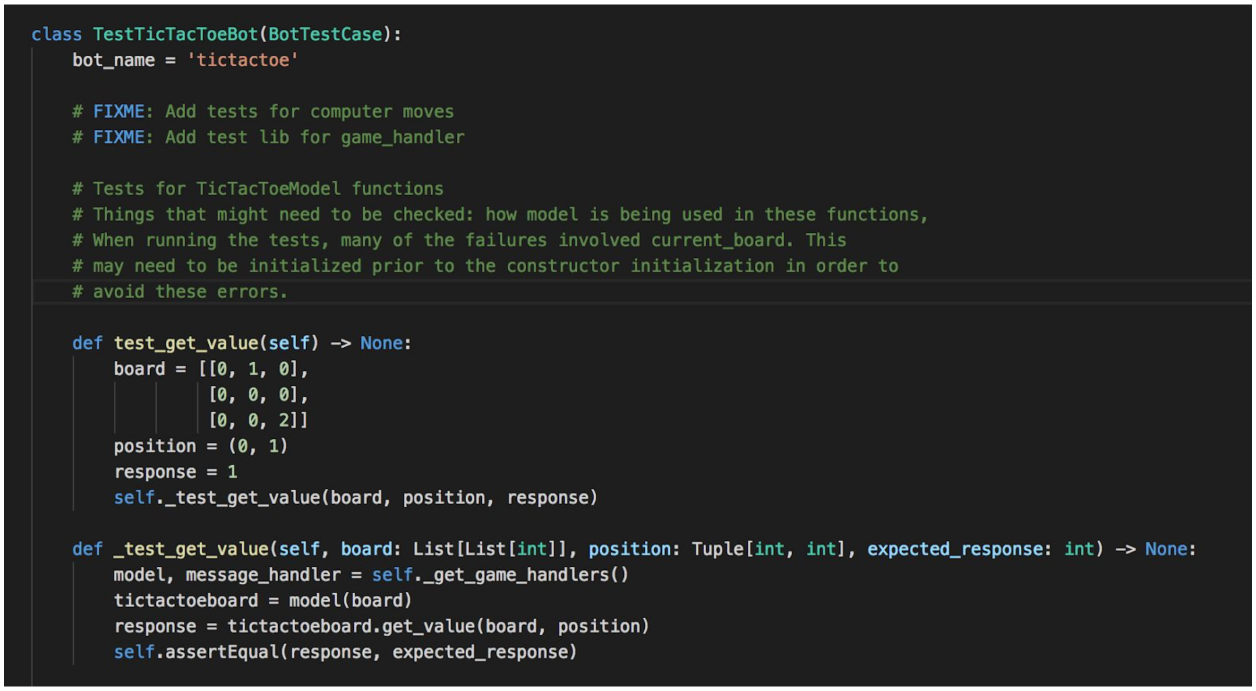
```
test_board_is_full (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot) ... ok
test_bot_responds_to_empty_message (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot) ... ok
test_bot_usage (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot) ... ok
test_contains_winning_move (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot) ... ok
test_determine_game_over_with_draw (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot) ... ok
test_determine_game_over_with_win (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot) ... ok
test_get_value (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot) ... ok
test_has_attributes (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot) ... ok
test_parse_board (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot) ... ok
test_player_color (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot) ... ok
test_static_responses (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot) ... ok
```

Image 3: Screenshot of linter output run locally, example of static analysis



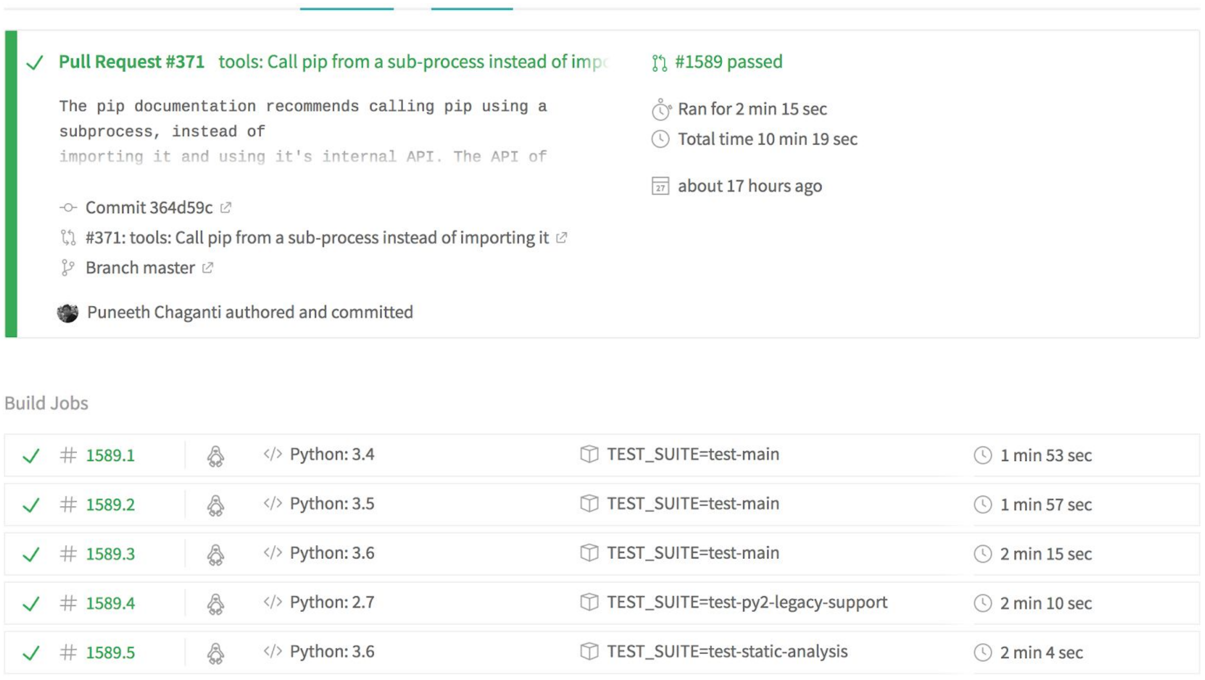
```
[(zulip-api-py3-venv) Kishans-MacBook-Pro:python-zulip-api kishan$ ./tools/lint
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:19:38: W291 trailing whitespace
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:25:30: E201 whitespace after '('
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:26:1: W293 blank line contains whitespace
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:42:59: W291 trailing whitespace
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:46:1: W293 blank line contains whitespace
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:60:1: W293 blank line contains whitespace
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:64:28: W291 trailing whitespace
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:67:1: W293 blank line contains whitespace
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:68:92: W291 trailing whitespace
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:73:1: W293 blank line contains whitespace
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:80:1: W293 blank line contains whitespace
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:86:1: W293 blank line contains whitespace
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:88:5: E303 too many blank lines (2)
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:104:24: W291 trailing whitespace
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:110:5: E303 too many blank lines (2)
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:114:1: W293 blank line contains whitespace
pep8       | zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py:120:1: W293 blank line contains whitespace
pep8       | zulip_bots/zulip_bots/bots/tictactoe/tictactoe.py:29:1: W293 blank line contains whitespace
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py line 19:
py         |     def test_get_value(self) -> None:
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py line 26:
py         |
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py line 42:
py         |         model, message_handler = self._get_game_handlers()
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py line 46:
py         |
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py line 60:
py         |
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py line 64:
py         |                  [2, 1, 2]]
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py line 67:
py         |
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py line 68:
py         |     def _test_board_is_full(self, board: List[List[int]], expected_response: bool) -> None:
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py line 73:
py         |
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py line 80:
py         |
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py line 86:
py         |
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py line 104:
py         |     #     model.board =
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py line 114:
py         |
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py line 120:
py         |
py         | Fix trailing whitespace at zulip_bots/zulip_bots/bots/tictactoe/tictactoe.py line 29:
py         |
```

Image 4: Comment that we wrote in our committed code explaining the changes that we made and what might need to be done in the future



```
class TestTicTacToeBot(BotTestCase):
    bot_name = 'tictactoe'

    # FIXME: Add tests for computer moves
    # FIXME: Add test lib for game_handler

    # Tests for TicTacToeModel functions
    # Things that might need to be checked: how model is being used in these functions,
    # When running the tests, many of the failures involved current_board. This
    # may need to be initialized prior to the constructor initialization in order to
    # avoid these errors.

    def test_get_value(self) -> None:
        board = [[0, 1, 0],
                 [0, 0, 0],
                 [0, 0, 2]]
        position = (0, 1)
        response = 1
        self._test_get_value(board, position, response)

    def _test_get_value(self, board: List[List[int]], position: Tuple[int, int], expected_response: int) -> None:
        model, message_handler = self._get_game_handlers()
        tictactoeboard = model(board)
        response = tictactoeboard.get_value(board, position)
        self.assertEqual(response, expected_response)
```

Image 5: Travis CI Build Output



✓ Pull Request #371 tools: Call pip from a sub-process instead of imp

#1589 passed

The pip documentation recommends calling pip using a subprocess, instead of importing it and using it's internal API. The API of

Ran for 2 min 15 sec

Total time 10 min 19 sec

about 17 hours ago

Commit 364d59c

#371: tools: Call pip from a sub-process instead of importing it

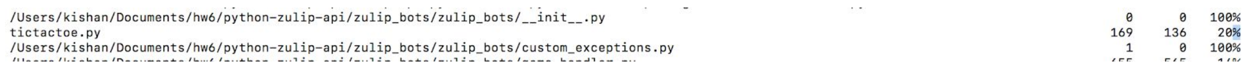
Branch master

Puneeth Chaganti authored and committed

Build Jobs

✓	# 1589.1	Python: 3.4	TEST_SUITE=test-main	1 min 53 sec
✓	# 1589.2	Python: 3.5	TEST_SUITE=test-main	1 min 57 sec
✓	# 1589.3	Python: 3.6	TEST_SUITE=test-main	2 min 15 sec
✓	# 1589.4	Python: 2.7	TEST_SUITE=test-py2-legacy-support	2 min 10 sec
✓	# 1589.5	Python: 3.6	TEST_SUITE=test-static-analysis	2 min 4 sec

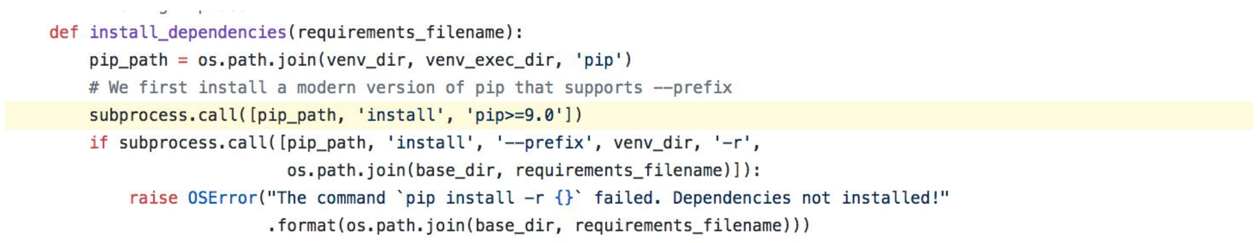
Image 6: This is the output of running coverage on tictactoe.py after we wrote test cases for the file. The coverage increased from 19% to 20%. We were unable to further increase the line coverage because there were bugs inherent in the test_tictactoe.py file from previous contributors that caused us to not further test the coverage on this file. This is an example of running dynamic analysis tools on code to augment/improve testing process.



```
/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/__init__.py                     0      0   100%
tictactoe.py                                                                                     169    136    20%
/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/custom_exceptions.py            1      0   100%
```

IX. Plan Updates

Initially, we planned to solve two bugs, one involving creating buttons for testing the Zulip account creation process and another testing the bots. However, after beginning the bot testing process and realizing that there were many more dependencies than we expected, we had to downsize our ambitions and focus solely on the bot tests. However, through the process of solving the bot tests, we encountered three more bugs that we committed and solved in our final PR. The first bug involved fixing the tools/provision file where line #86 was causing an issue that was preventing us from entering the Zulip virtual environment. The issue was specifically that the provision file would only run or accept pip that is a version greater than 9.0. We tried building and running the python-zulip-api on OS X, Windows, and Unix/Linux, but the bug pertaining to the tools/provision file kept us from running it. We fixed this bug by changing the line to allow pip version 9.0.3. This error can be seen below in the highlighted line which is line 86:



```python
def install_dependencies(requirements_filename):
    pip_path = os.path.join(venv_dir, venv_exec_dir, 'pip')
    # We first install a modern version of pip that supports --prefix
    subprocess.call([pip_path, 'install', 'pip>=9.0'])
    if subprocess.call([pip_path, 'install', '--prefix', venv_dir, '-r',
                        os.path.join(base_dir, requirements_filename)]):
        raise OSError("The command `pip install -r {}` failed. Dependencies not installed!"
                      .format(os.path.join(base_dir, requirements_filename)))
```

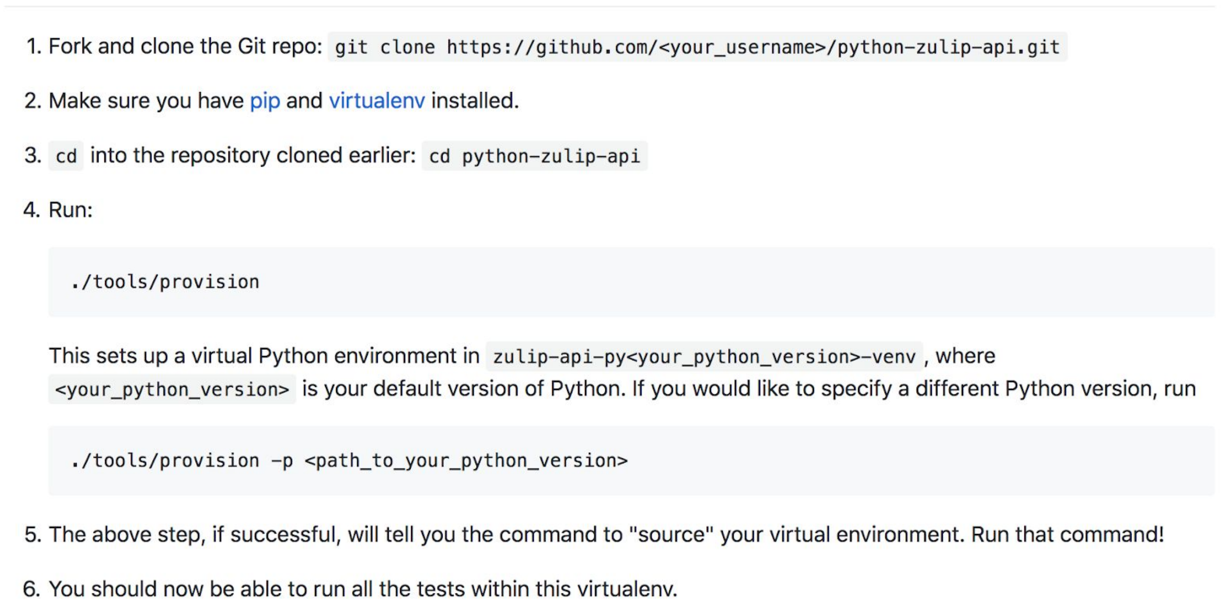
While we both originally scheduled/planned to split up the tasks for fixing button and for improving test coverage, neither of us were able to get the virtual environment for Zulip to run as a result of the existing issue regard pip that is aforementioned. This is an unanticipated risk that was encountered. Prior to our attempts to run Zulip Python API, we did not expect any bugs to exist in the existing Zulip codebase that would prevent us from even building or running the code, let alone fix its issues. Prior to working on the project, we expected the build and run process to be fairly seamless as it is an open source project and chances are that another developer would have already encountered the very same issue we encountered with pip. However, the difference when we started working on the project were the challenges that unexpected risks/defects usually accompany. These unexpected risks cost us valuable time that could have been spent completing the 2nd task of fixing buttons that we originally also intended to contribute. However, because of these unexpected risks we were only able to reserve time to work on the test-coverage task and fix the existing issue of the pip version. In addition, both of us had to contribute time to fix this issue as we were both encountering the same issue during the build process of Zulip, so we were not able to work separately on the different tasks.

X. Experiences and Recommendations

Overall, we had a very difficult experience with this project, mainly because at every step, we faced an unexpected bug in a part of the code that we were not expecting to have to fix. However, it ended up being a good learning experience of how to contribute to an open-source project and the many dependencies that are involved. We were not able to fully solve the issue that we originally claimed because of the unanticipated challenge of not being able to run the Zulip Python API from the beginning, but we were able to identify and solve several bugs which were preventing us from doing so. Therefore, we were able to gain an understanding of how the codebase worked as whole and learned how to tackle unexpected issues.

At the beginning, we tried to work on a bug involving creating buttons for testing. However, the requirements given in the issue were too vague for us to easily figure out where to implement these buttons, and it was difficult to ascertain what the proper way to contribute to this issue would be from inferring from previous code. We tried contacting developers involved to gain some clarity on this issue, but were unable to get advice in a reasonable amount of time.

Following this, we decided to turn to an issue in a different codebase (zulip/python-zulip-api rather than zulip/zulip). This issue involved creating new test cases for the bot tictactoe.py. In order to begin work on these tests, we had to be able to run them locally. The Zulip documentation provided guidelines for how to run these test cases, which initially seemed like a painless sequence of three commands **(Image 1)**.



1. Fork and clone the Git repo: `git clone https://github.com/<your_username>/python-zulip-api.git`
2. Make sure you have pip and virtualenv installed.
3. `cd` into the repository cloned earlier: `cd python-zulip-api`
4. Run:

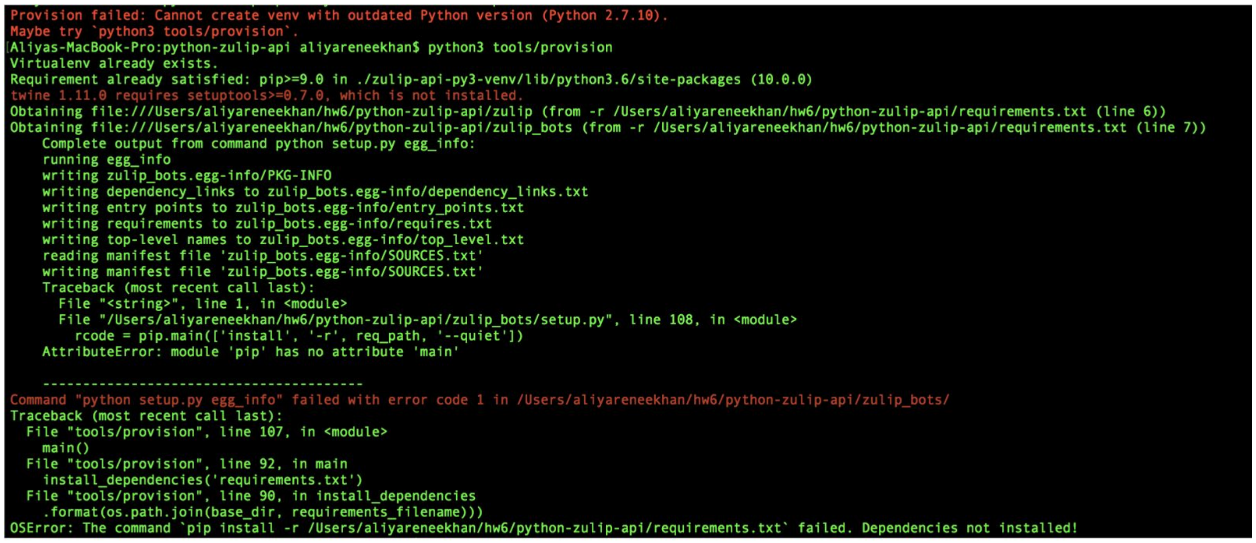
   ```
   ./tools/provision
   ```

   This sets up a virtual Python environment in `zulip-api-py<your_python_version>-venv`, where `<your_python_version>` is your default version of Python. If you would like to specify a different Python version, run

   ```
   ./tools/provision -p <path_to_your_python_version>
   ```

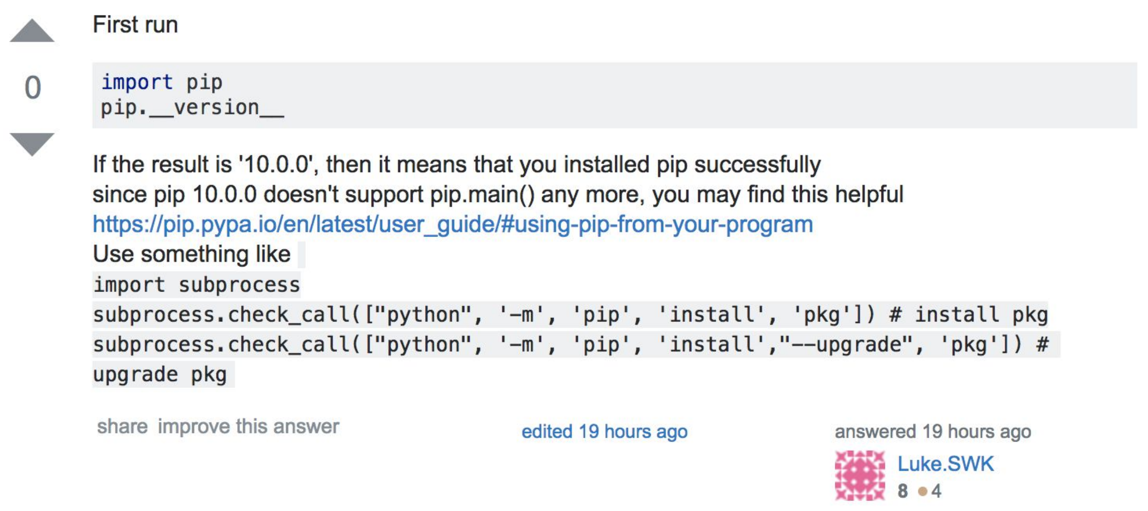
5. The above step, if successful, will tell you the command to "source" your virtual environment. Run that command!
6. You should now be able to run all the tests within this virtualenv.

However, each time that we attempted to run the commands, we came across the bug "pip has no attribute main" **(Image 2).**



```
Provision failed: Cannot create venv with outdated Python version (Python 2.7.10).
Maybe try `python3 tools/provision`.
[Aliyas-MacBook-Pro:python-zulip-api aliyareneekhan$ python3 tools/provision
Virtualenv already exists.
Requirement already satisfied: pip>=9.0 in ./zulip-api-py3-venv/lib/python3.6/site-packages (10.0.0)
twine 1.11.0 requires setuptools>=0.7.0, which is not installed.
Obtaining file:///Users/aliyareneekhan/hw6/python-zulip-api/zulip (from -r /Users/aliyareneekhan/hw6/python-zulip-api/requirements.txt (line 6))
Obtaining file:///Users/aliyareneekhan/hw6/python-zulip-api/zulip_bots (from -r /Users/aliyareneekhan/hw6/python-zulip-api/requirements.txt (line 7))
    Complete output from command python setup.py egg_info:
    running egg_info
    writing zulip_bots.egg-info/PKG-INFO
    writing dependency_links to zulip_bots.egg-info/dependency_links.txt
    writing entry points to zulip_bots.egg-info/entry_points.txt
    writing requirements to zulip_bots.egg-info/requires.txt
    writing top-level names to zulip_bots.egg-info/top_level.txt
    reading manifest file 'zulip_bots.egg-info/SOURCES.txt'
    writing manifest file 'zulip_bots.egg-info/SOURCES.txt'
    Traceback (most recent call last):
      File "<string>", line 1, in <module>
      File "/Users/aliyareneekhan/hw6/python-zulip-api/zulip_bots/setup.py", line 108, in <module>
        rcode = pip.main(['install', '-r', req_path, '--quiet'])
    AttributeError: module 'pip' has no attribute 'main'

    ----------------------------------------
Command "python setup.py egg_info" failed with error code 1 in /Users/aliyareneekhan/hw6/python-zulip-api/zulip_bots/
Traceback (most recent call last):
  File "tools/provision", line 107, in <module>
    main()
  File "tools/provision", line 92, in main
    install_dependencies('requirements.txt')
  File "tools/provision", line 90, in install_dependencies
    .format(os.path.join(base_dir, requirements_filename)))
OSError: The command `pip install -r /Users/aliyareneekhan/hw6/python-zulip-api/requirements.txt` failed. Dependencies not installed!
```

We initially thought that this could be an issue with the way we installed pip. We found a post online regarding the issue which suggested that pip 10.0.0 broke some of the dependencies needed in order to run main **(Image 3).**


0

First run

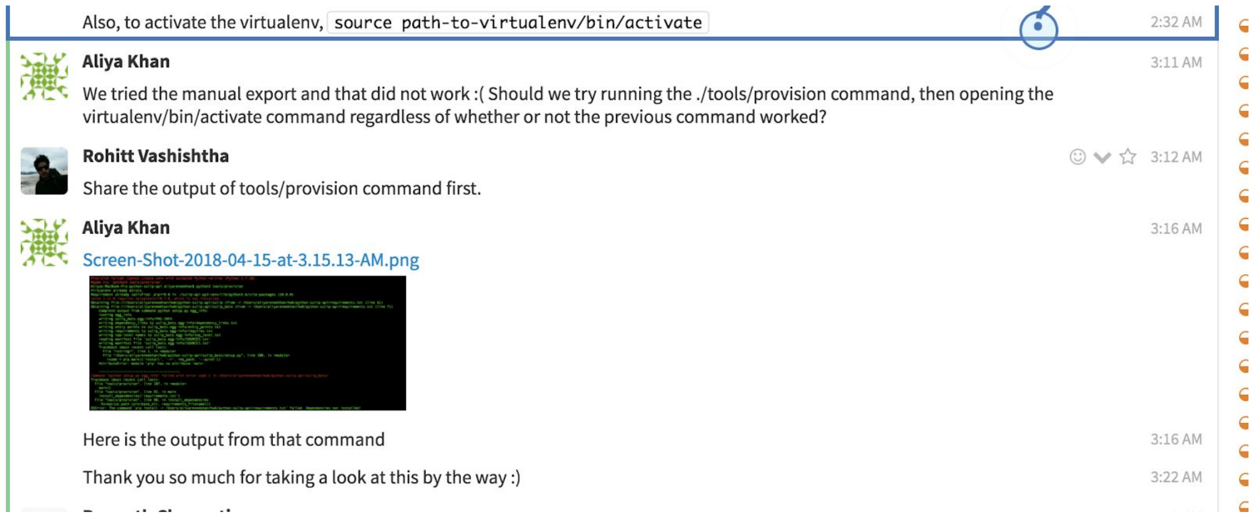
```
import pip
pip.__version__
```

If the result is '10.0.0', then it means that you installed pip successfully
since pip 10.0.0 doesn't support pip.main() any more, you may find this helpful
https://pip.pypa.io/en/latest/user_guide/#using-pip-from-your-program
Use something like

```
import subprocess
subprocess.check_call(["python", '-m', 'pip', 'install', 'pkg']) # install pkg
subprocess.check_call(["python", '-m', 'pip', 'install',"--upgrade", 'pkg']) #
upgrade pkg
```

share improve this answer edited 19 hours ago answered 19 hours ago Luke.SWK 8 ●4

We then tried to uninstall and reinstall pip on our local system, which did not fix the issue. On one computer, we attempted to overwrite the version of Python built in to the Mac (2.7) with the updated version of Python in case this was the issue. This also did not fix the issue. Overall, we tried running the commands on Mac OSX High Sierra 10.12.6 and 10.13.4, as well as using Linux through vagrant and VirtualBox and Windows 10. The commands did not work on any of these operating systems, so we concluded that there was an issue with the file itself. Throughout this process, we also learned how to use the Zulip chat mechanism to communicate with other developers and ask about issues. Below are screenshots from our communication **(Image 4 - multiple images)**.


Also, to activate the virtualenv, `source path-to-virtualenv/bin/activate` 2:32 AM

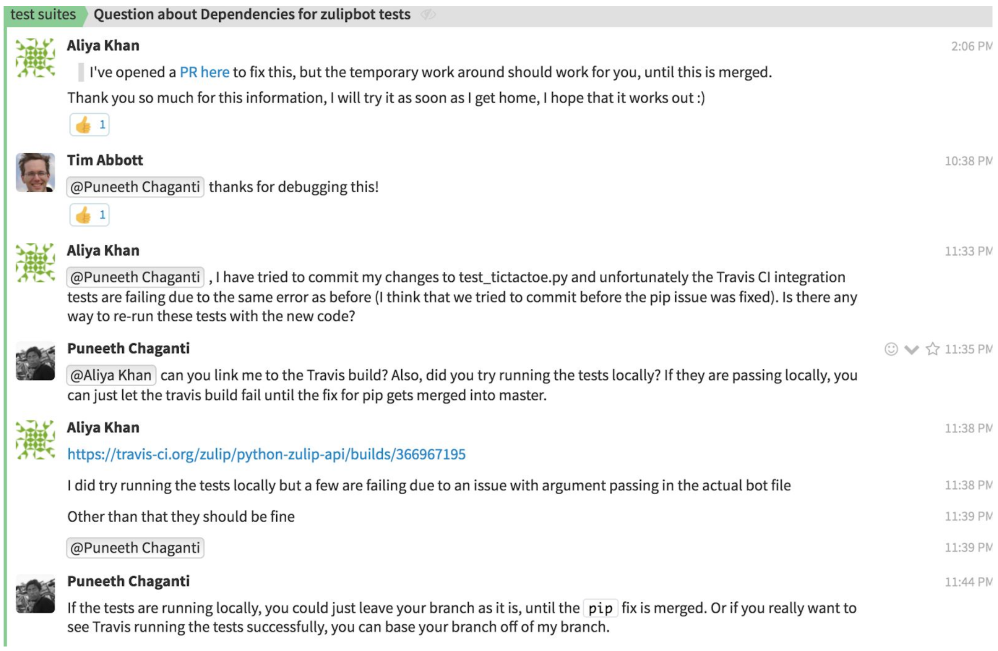
Aliya Khan 3:11 AM
We tried the manual export and that did not work :(Should we try running the ./tools/provision command, then opening the virtualenv/bin/activate command regardless of whether or not the previous command worked?

Rohitt Vashishtha 3:12 AM
Share the output of tools/provision command first.

Aliya Khan 3:16 AM
Screen-Shot-2018-04-15-at-3.15.13-AM.png

Here is the output from that command 3:16 AM

Thank you so much for taking a look at this by the way :) 3:22 AM



test suites | Question about Dependencies for zulipbot tests

Aliya Khan 2:06 PM

> I've opened a PR here to fix this, but the temporary work around should work for you, until this is merged.

Thank you so much for this information, I will try it as soon as I get home, I hope that it works out :)

👍 1

Tim Abbott 10:38 PM

@Puneeth Chaganti thanks for debugging this!

👍 1

Aliya Khan 11:33 PM

@Puneeth Chaganti , I have tried to commit my changes to test_tictactoe.py and unfortunately the Travis CI integration tests are failing due to the same error as before (I think that we tried to commit before the pip issue was fixed). Is there any way to re-run these tests with the new code?

Puneeth Chaganti 11:35 PM

@Aliya Khan can you link me to the Travis build? Also, did you try running the tests locally? If they are passing locally, you can just let the travis build fail until the fix for pip gets merged into master.

Aliya Khan 11:38 PM

https://travis-ci.org/zulip/python-zulip-api/builds/366967195

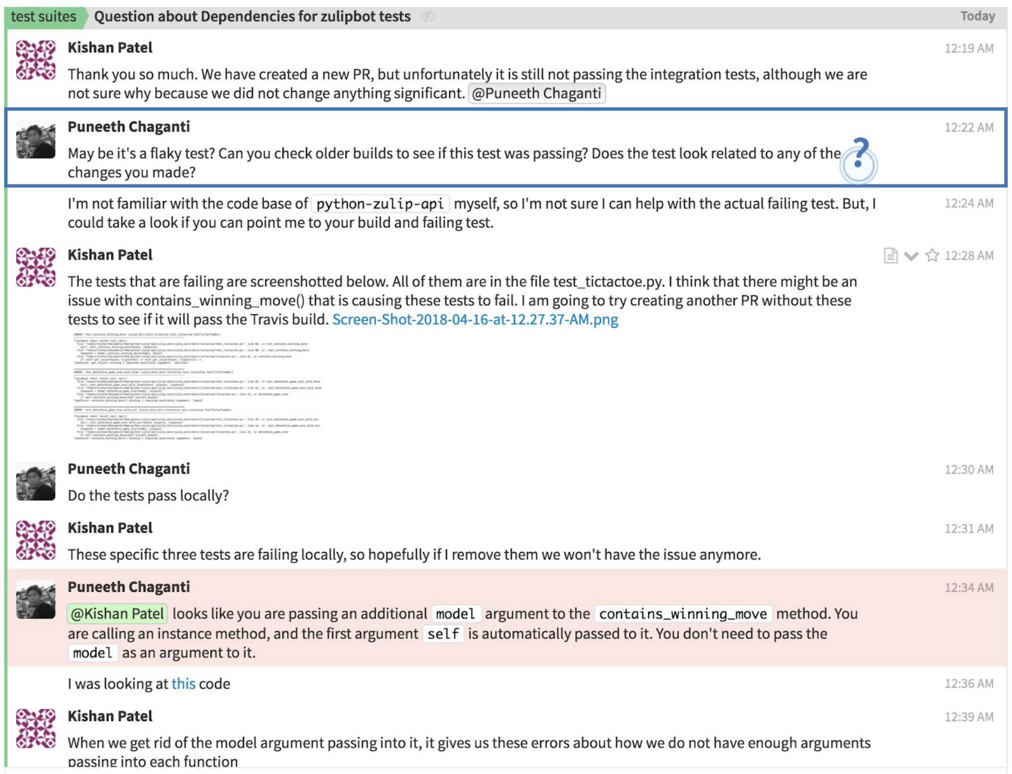
I did try running the tests locally but a few are failing due to an issue with argument passing in the actual bot file 11:38 PM

Other than that they should be fine 11:39 PM

@Puneeth Chaganti 11:39 PM

Puneeth Chaganti 11:44 PM

If the tests are running locally, you could just leave your branch as it is, until the `pip` fix is merged. Or if you really want to see Travis running the tests successfully, you can base your branch off of my branch.



test suites | Question about Dependencies for zulipbot tests | Today

Kishan Patel 12:19 AM

Thank you so much. We have created a new PR, but unfortunately it is still not passing the integration tests, although we are not sure why because we did not change anything significant. @Puneeth Chaganti

Puneeth Chaganti 12:22 AM

May be it's a flaky test? Can you check older builds to see if this test was passing? Does the test look related to any of the changes you made?

I'm not familiar with the code base of `python-zulip-api` myself, so I'm not sure I can help with the actual failing test. But, I could take a look if you can point me to your build and failing test. 12:24 AM

Kishan Patel 12:28 AM

The tests that are failing are screenshotted below. All of them are in the file test_tictactoe.py. I think that there might be an issue with contains_winning_move() that is causing these tests to fail. I am going to try creating another PR without these tests to see if it will pass the Travis build. Screen-Shot-2018-04-16-at-12.27.37-AM.png

Puneeth Chaganti 12:30 AM

Do the tests pass locally?

Kishan Patel 12:31 AM

These specific three tests are failing locally, so hopefully if I remove them we won't have the issue anymore.

Puneeth Chaganti 12:34 AM

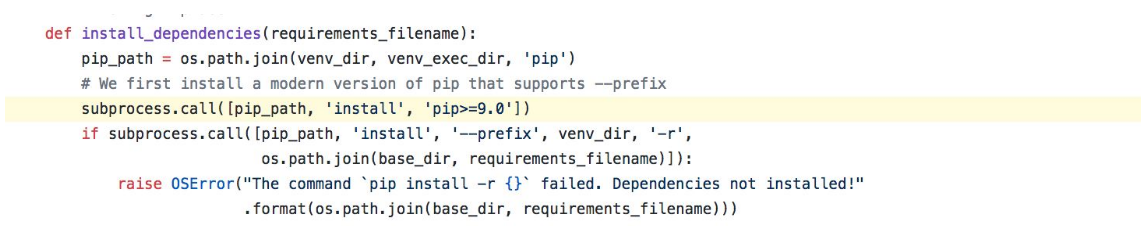
@Kishan Patel looks like you are passing an additional `model` argument to the `contains_winning_move` method. You are calling an instance method, and the first argument `self` is automatically passed to it. You don't need to pass the `model` as an argument to it.

I was looking at this code 12:36 AM

Kishan Patel 12:39 AM

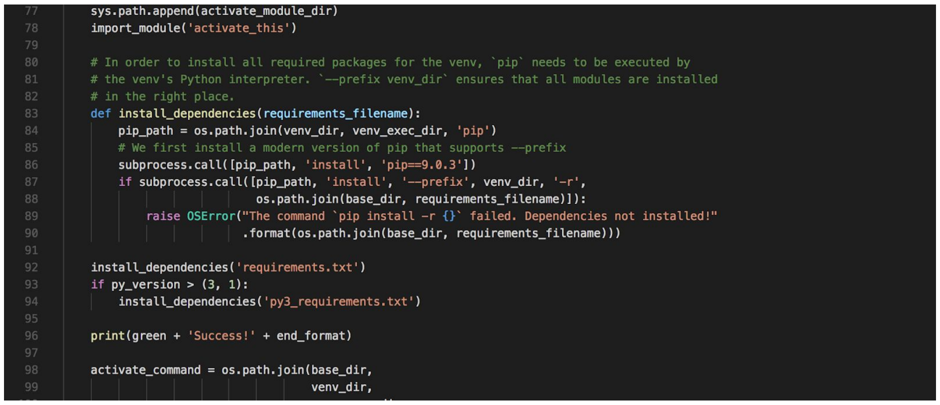
When we get rid of the model argument passing into it, it gives us these errors about how we do not have enough arguments passing into each function

Upon inspecting tools/provision, we figured out that there was a line where the pip version was set to >= 9.0 **(Image 5)**.

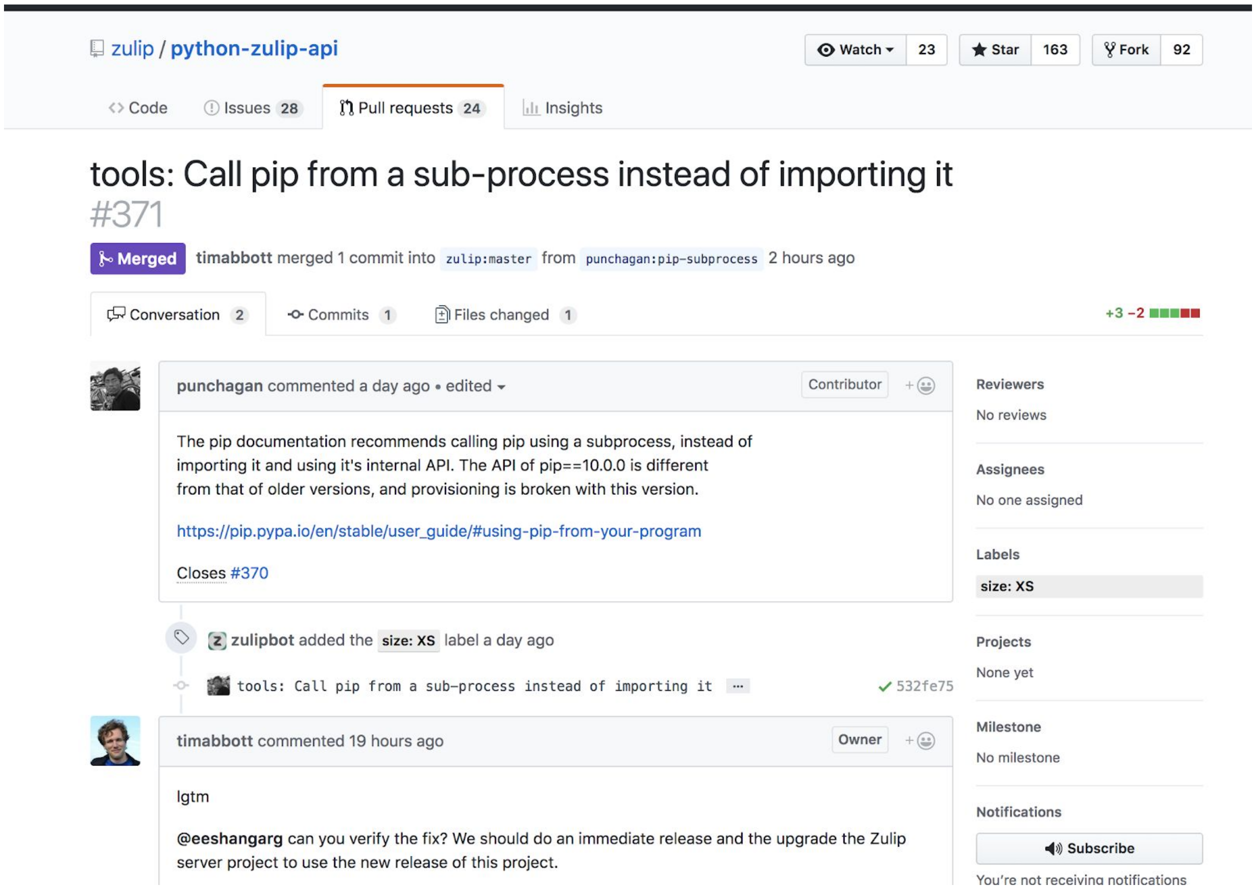


```
def install_dependencies(requirements_filename):
    pip_path = os.path.join(venv_dir, venv_exec_dir, 'pip')
    # We first install a modern version of pip that supports --prefix
    subprocess.call([pip_path, 'install', 'pip>=9.0'])
    if subprocess.call([pip_path, 'install', '--prefix', venv_dir, '-r',
                        os.path.join(base_dir, requirements_filename)]):
        raise OSError("The command `pip install -r {}` failed. Dependencies not installed!"
                      .format(os.path.join(base_dir, requirements_filename)))
```

We realized that this might override our local version of pip with whichever version this line decided to choose, most likely the recent version 10.0.0 which broke the dependencies required to run main. Therefore, we changed this line to set the version equal to 9.0.3, and notified the developers so that they could fix this issue in the overall codebase **(Image 6)**.


```
    sys.path.append(activate_module_dir)
    import_module('activate_this')

    # In order to install all required packages for the venv, `pip` needs to be executed by
    # the venv's Python interpreter. `--prefix venv_dir` ensures that all modules are installed
    # in the right place.
    def install_dependencies(requirements_filename):
        pip_path = os.path.join(venv_dir, venv_exec_dir, 'pip')
        # We first install a modern version of pip that supports --prefix
        subprocess.call([pip_path, 'install', 'pip==9.0.3'])
        if subprocess.call([pip_path, 'install', '--prefix', venv_dir, '-r',
                            os.path.join(base_dir, requirements_filename)]):
            raise OSError("The command `pip install -r {}` failed. Dependencies not installed!"
                          .format(os.path.join(base_dir, requirements_filename)))

    install_dependencies('requirements.txt')
    if py_version > (3, 1):
        install_dependencies('py3_requirements.txt')

    print(green + 'Success!' + end_format)

    activate_command = os.path.join(base_dir,
                                    venv_dir,
```

They then created a separate pull request for this issue and fixed it (**Image 7)**.



tools: Call pip from a sub-process instead of importing it #371

Merged timabbott merged 1 commit into zulip:master from punchagan:pip-subprocess 2 hours ago

punchagan commented a day ago • edited Contributor

The pip documentation recommends calling pip using a subprocess, instead of importing it and using it's internal API. The API of pip==10.0.0 is different from that of older versions, and provisioning is broken with this version.

https://pip.pypa.io/en/stable/user_guide/#using-pip-from-your-program

Closes #370

zulipbot added the size: XS label a day ago

tools: Call pip from a sub-process instead of importing it ✓ 532fe75

timabbott commented 19 hours ago Owner

lgtm

@eeshangarg can you verify the fix? We should do an immediate release and the upgrade the Zulip server project to use the new release of this project.

Reviewers: No reviews
Assignees: No one assigned
Labels: size: XS
Projects: None yet
Milestone: No milestone
Notifications: Subscribe
You're not receiving notifications

After we were finally able to run the test cases, we began working on the file test_tictactoe.py with hopes of increasing the coverage of this file to 100%. However, as we soon found, the test cases had no good precedents for style or structure, and we ended up having to do a lot more work internally in order to make the test cases run properly with the system, because there was an issue with the way that the tests were being initialized that were not allowing any of the tests we tried to write to run properly. At first, we attempted to write tests for the large function computer_move (**Image 8**). However, we soon found that several smaller functions that were being called from within the computer_move function were untested such as "contains_winning_move", and this was making it difficult to create an initialized "tic tac toe board" to test the computer move function with. We then noticed that there were no functions in TicTacToeModel that had any test cases. Therefore, we decided to shift to trying to test some of the smaller functions, including get_value, determine_game_over, board_is_full, and contains_winning_move **(Image 9).** We hoped to create a precedent for later test cases that would be written in the same file.

One of the first issues we ran into was that the test functions were unable to access the local variable current_board from the TicTacToeModel class. We tried explicitly declaring this variable outside of the constructor within the tictactoe.py function, and this seemed to fix the issue (Image 10). We noted this issue and fix when we submitted our pull request.

Additionally, we ran into several issues with the way that arguments were passed into functions (**Image 11**). At first, we tried to hard code these into our test cases, but issues still arose when functions would call other functions within themselves. Therefore, we tried to simplify the way that we were testing by initializing our own instance of the class TicTacToeModel() ourselves so that we would not depend on any other files that might be initializing the model incorrectly. Previously to this, initialization of the bot was done by the function _get_game_handlers(), but for some reason the model created by this function would not correctly save and update the tictactoe board (**Image 12**). Therefore, we tried to create our own instance by importing the tictactoe.py file directly into the test file using from . import tictactoe.

This created an issue where the module tictactoe could not be found. At this point, we turned to the Zulip chat in order to try to get help from other developers, in case they had more information about Zulip standards or dependencies that we might be missing. We learned that we should the _get_game_handlers() function in order to initialize the model, but that the implementation of_get_game_handlers() was buggy such that it was getting the entire class TicTacToeModel instead of just creating an instance of it. Therefore, we changed our test cases so that it used _get_game_handlers() but within each case initialized a new instance of the model **(Image 13).** Overall, the general structure of our test cases would include two functions per case: one would initialize the board for that specific test and the test oracle, which was the expected response for the function with that test case. We then called another function which would initialize an instance of the TicTacToeModel() with the tic-tac-toe board we created in the first function. It then would run the function being tested with the given parameters and store the response in a variable called "response." This response was then compared with the test oracle, saved in the variable "expected_response", to make sure that the function performed as expected in this specific instance. After smoothing out our new code using a linter so that it would pass the Zulip static analysis test, we were able to submit a pull request **(Image 14)** that passed the Travis CI Integration tests and therefore was accepted by the maintainers into the codebase **(Image 15).**

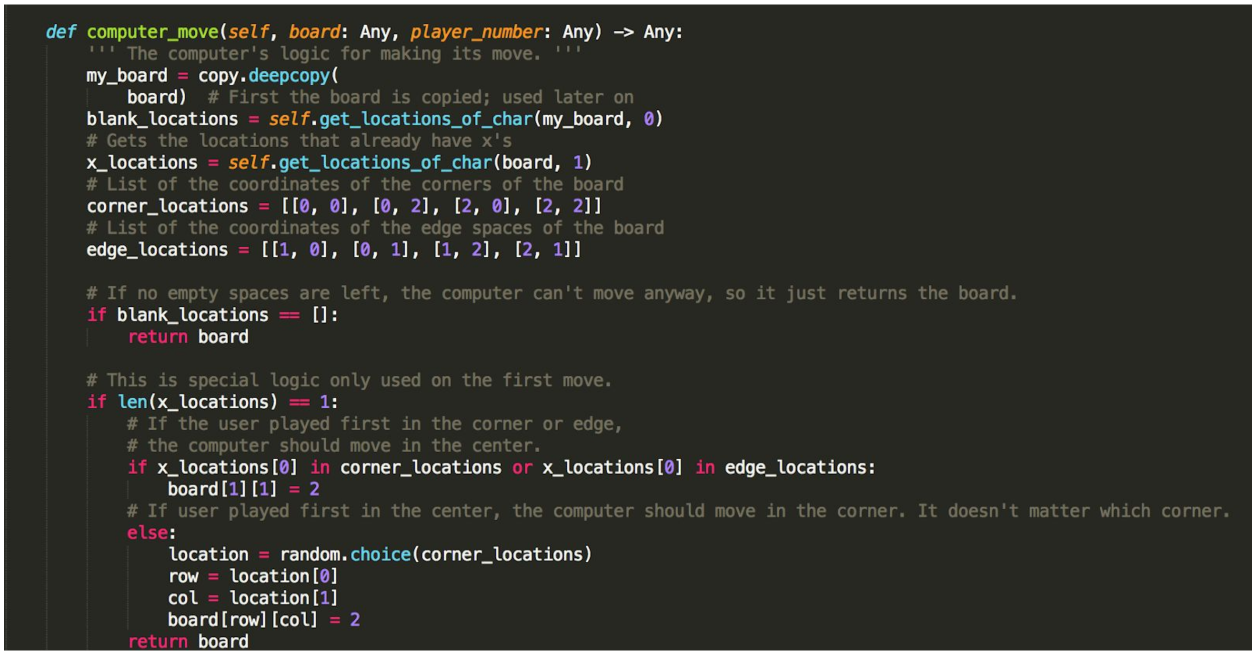
If we were to start an open-source project from scratch, we would make sure to add a note on each tracked issue in order to make sure that any new contributor that claims the specific issue has some idea of common problems that previous contributors have had in relation to running, building, or debugging code related to the respective issue. This would help to avoid the issue we ran into where pip version 10.0.0 does not allow for an import of main package to pip module. If a previous developer who may have run into this issue already were to write a note on the page responsible for the specific issue, then it would have been easier to fix the bug without using more time. Additionally, there was no prerequisite or standard available for how to write test cases for the bots, and as a result we ended up spending a lot of our time debugging previous implementations of the test cases. Including some sort of standard in the documentation for the project would have been really helpful for creating better tests, and if we were provided this, we might have been able to spend more time creating more complex test cases that would substantially increase the coverage.

In this project, the communication between developers and contributors is essential to ensuring a fast rate of fixes and we experienced this help when we used the Zulip Chat to receive help from someone more familiar with the codebase. While this aspect of the project was good, the documentation provided was severely inadequate. Specifically, the test cases we would be producing were not held to any specific coding style standard; therefore, we had to produce the test cases without following any specific readability or coding style standard. While this does not affect the overall Zulip project currently, it may begin to affect the Zulip project codebase if no documentation or coding style standard is specified. The test cases that were previously in the test_tictactoe.py from previous contributors had no comments identifying what the test cases covered.

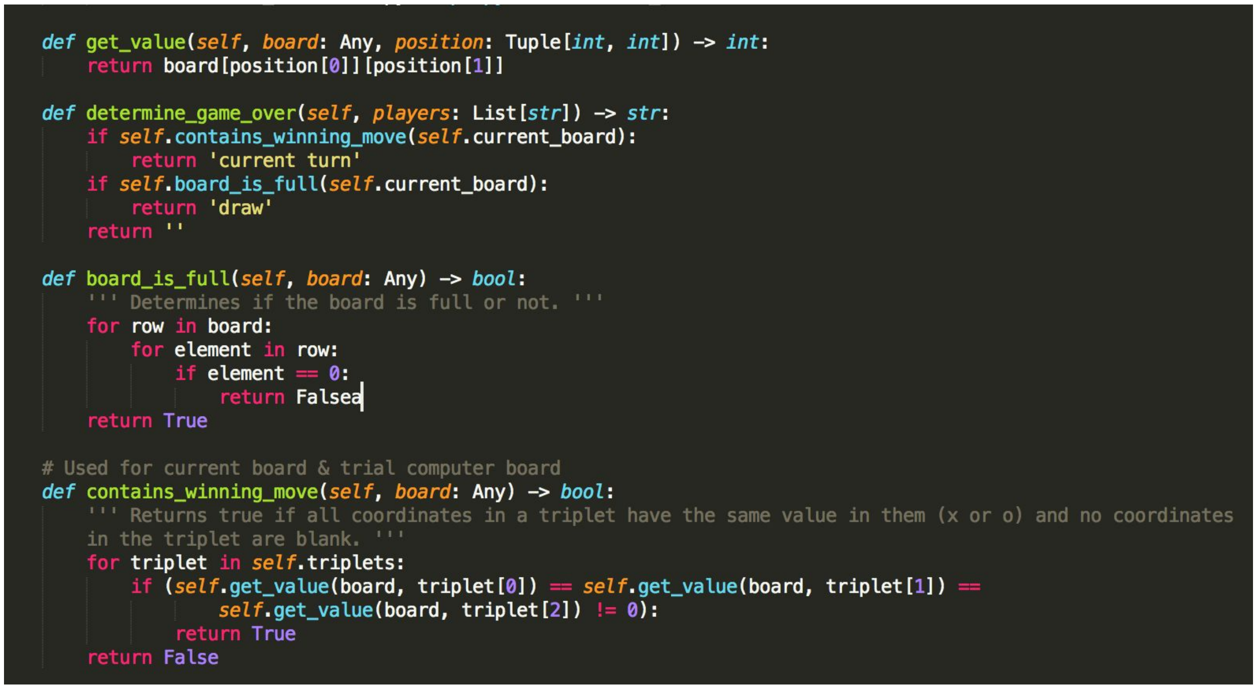
Throughout this project, we applied several of the concepts covered in this class. Regarding the tests that we wrote, we used code coverage when we ran coverage.py against the test cases we wrote and test oracles when we implemented the test cases and compared the results of each function with the expected result. Additionally, we ran a linter and unit tests locally before submitting our code for a pull request. We also informally reviewed each others code and for many parts of the development process used a pair programming approach where one of us would be writing the code and the other would be checking for errors. We also had to report a defect that we found in the code while developing (the issue with the version of pip), and were able to observe the process that Zulip used to address and fix this issue. We also had an initial issue with requirements elicitation because the requirements given to us in the first bug we tried to fix were too ambiguous for us to effectively tackle the issue. Finally, after we submitted the pull request, our code was subjected to integration tests and further static analysis. We also were able to make an improvement in the overall maintainability of the code by creating a standard for the test cases for this bot which can be used for future testers. Overall, we experienced the heavy focus on reading code, requirements elicitation, and communicating with developers versus actually writing code firsthand. Most of our time was spent reading the codebase, identifying dependencies that could be creating issues, and communicating with other Zulip developers. While this was a very challenging experience, it was a good conclusion to this class because we were able to apply concepts learned in almost every single lecture.

Our main recommendation would be to find someone else within the open-source project who knows the code that you are working on very well - this proved very useful to us as we became discouraged by bugs in the later stages of our development process. The person who helped us with the project readily helped us learn many dependencies and intricacies of the style of the project that would have been much more tedious to learn otherwise. Additionally, at the beginning of our development process we did not expect to encounter other bugs in the code while we were developing, so accounting for this ahead of time when planning out how to tackle the issues would be another recommendation.

Image 8: Beginning of the function computer_move that we initially tried to test.

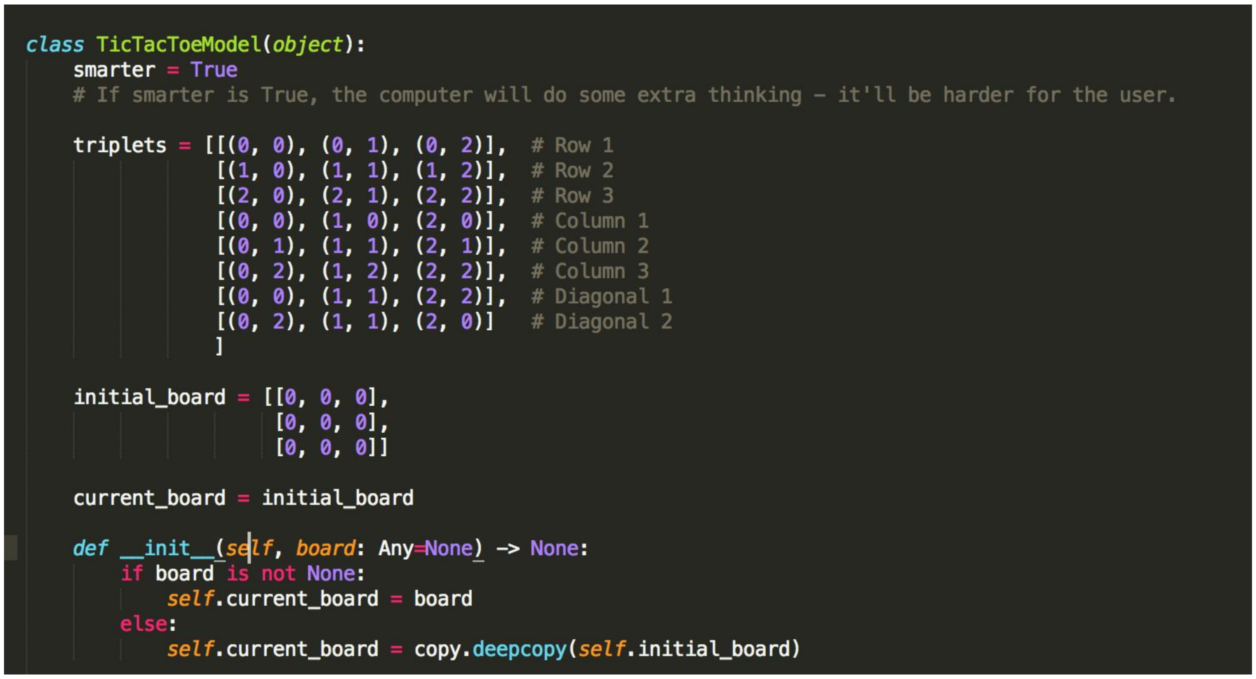


```
def computer_move(self, board: Any, player_number: Any) -> Any:
    ''' The computer's logic for making its move. '''
    my_board = copy.deepcopy(
        board)  # First the board is copied; used later on
    blank_locations = self.get_locations_of_char(my_board, 0)
    # Gets the locations that already have x's
    x_locations = self.get_locations_of_char(board, 1)
    # List of the coordinates of the corners of the board
    corner_locations = [[0, 0], [0, 2], [2, 0], [2, 2]]
    # List of the coordinates of the edge spaces of the board
    edge_locations = [[1, 0], [0, 1], [1, 2], [2, 1]]

    # If no empty spaces are left, the computer can't move anyway, so it just returns the board.
    if blank_locations == []:
        return board

    # This is special logic only used on the first move.
    if len(x_locations) == 1:
        # If the user played first in the corner or edge,
        # the computer should move in the center.
        if x_locations[0] in corner_locations or x_locations[0] in edge_locations:
            board[1][1] = 2
        # If user played first in the center, the computer should move in the corner. It doesn't matter which corner.
        else:
            location = random.choice(corner_locations)
            row = location[0]
            col = location[1]
            board[row][col] = 2
        return board
```



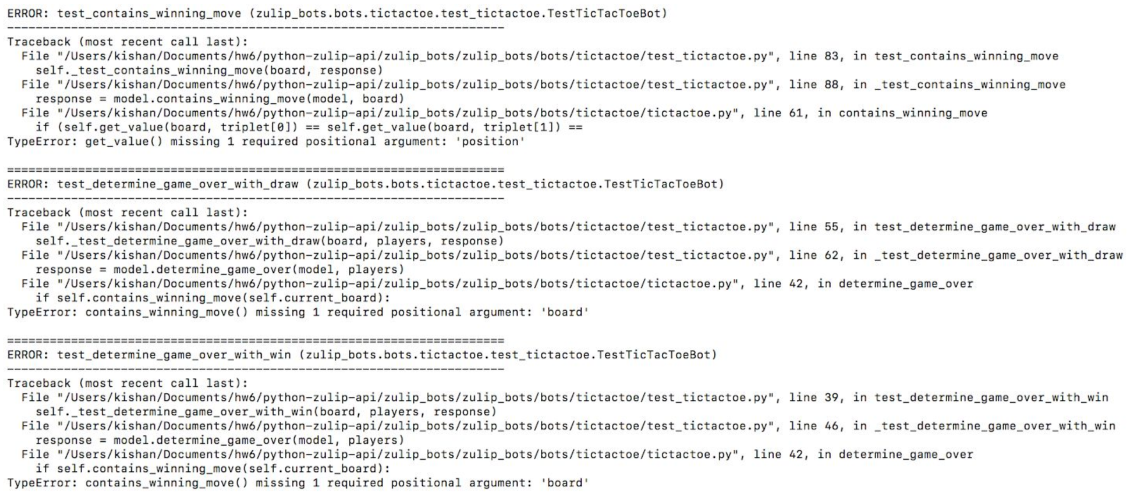
```python
def get_value(self, board: Any, position: Tuple[int, int]) -> int:
    return board[position[0]][position[1]]

def determine_game_over(self, players: List[str]) -> str:
    if self.contains_winning_move(self.current_board):
        return 'current turn'
    if self.board_is_full(self.current_board):
        return 'draw'
    return ''

def board_is_full(self, board: Any) -> bool:
    ''' Determines if the board is full or not. '''
    for row in board:
        for element in row:
            if element == 0:
                return Falsea
    return True

# Used for current board & trial computer board
def contains_winning_move(self, board: Any) -> bool:
    ''' Returns true if all coordinates in a triplet have the same value in them (x or o) and no coordinates
    in the triplet are blank. '''
    for triplet in self.triplets:
        if (self.get_value(board, triplet[0]) == self.get_value(board, triplet[1]) ==
                self.get_value(board, triplet[2]) != 0):
            return True
    return False
```

Image 9: Smaller TicTacToeModel functions that we ended up testing.
Image 10: Change that we made in tictactoe.py. Initially, current_board was only referenced in the constructor. However, this created difficulties when the variable current_board was referenced because the variable could not be found. Therefore, we added the line "current_board = initial_board" in order to fix this issue.



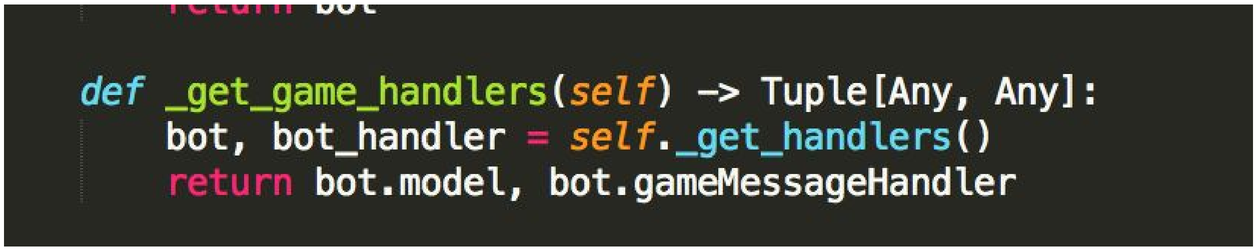
```python
class TicTacToeModel(object):
    smarter = True
    # If smarter is True, the computer will do some extra thinking - it'll be harder for the user.

    triplets = [[(0, 0), (0, 1), (0, 2)],  # Row 1
                [(1, 0), (1, 1), (1, 2)],  # Row 2
                [(2, 0), (2, 1), (2, 2)],  # Row 3
                [(0, 0), (1, 0), (2, 0)],  # Column 1
                [(0, 1), (1, 1), (2, 1)],  # Column 2
                [(0, 2), (1, 2), (2, 2)],  # Column 3
                [(0, 0), (1, 1), (2, 2)],  # Diagonal 1
                [(0, 2), (1, 1), (2, 0)]   # Diagonal 2
                ]

    initial_board = [[0, 0, 0],
                     [0, 0, 0],
                     [0, 0, 0]]

    current_board = initial_board

    def __init__(self, board: Any=None) -> None:
        if board is not None:
            self.current_board = board
        else:
            self.current_board = copy.deepcopy(self.initial_board)
```

Image 11: Function missing one required positional argument error



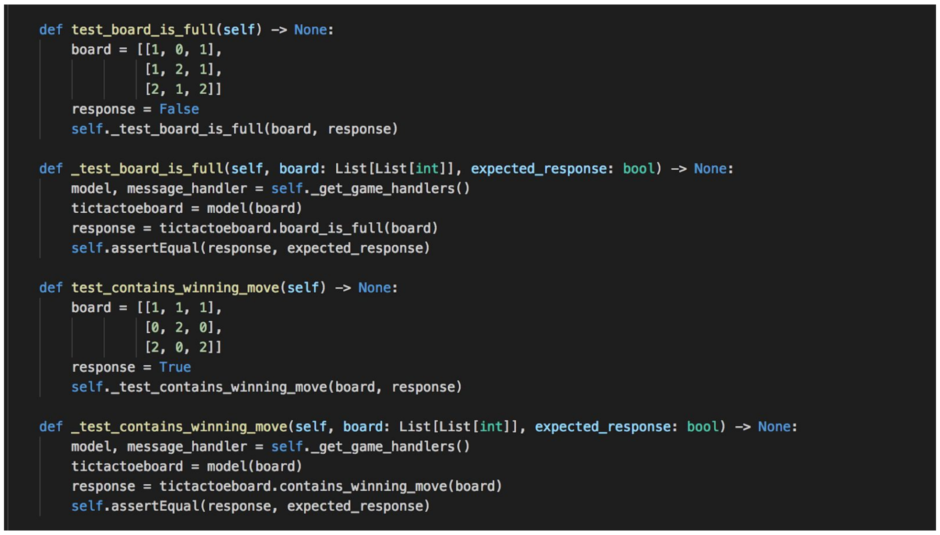
```
ERROR: test_contains_winning_move (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py", line 83, in test_contains_winning_move
    self._test_contains_winning_move(board, response)
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py", line 88, in _test_contains_winning_move
    response = model.contains_winning_move(model, board)
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/tictactoe.py", line 61, in contains_winning_move
    if (self.get_value(board, triplet[0]) == self.get_value(board, triplet[1]) ==
TypeError: get_value() missing 1 required positional argument: 'position'

======================================================================
ERROR: test_determine_game_over_with_draw (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py", line 55, in test_determine_game_over_with_draw
    self._test_determine_game_over_with_draw(board, players, response)
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py", line 62, in _test_determine_game_over_with_draw
    response = model.determine_game_over(model, players)
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/tictactoe.py", line 42, in determine_game_over
    if self.contains_winning_move(self.current_board):
TypeError: contains_winning_move() missing 1 required positional argument: 'board'

======================================================================
ERROR: test_determine_game_over_with_win (zulip_bots.bots.tictactoe.test_tictactoe.TestTicTacToeBot)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py", line 39, in test_determine_game_over_with_win
    self._test_determine_game_over_with_win(board, players, response)
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/test_tictactoe.py", line 46, in _test_determine_game_over_with_win
    response = model.determine_game_over(model, players)
  File "/Users/kishan/Documents/hw6/python-zulip-api/zulip_bots/zulip_bots/bots/tictactoe/tictactoe.py", line 42, in determine_game_over
    if self.contains_winning_move(self.current_board):
TypeError: contains_winning_move() missing 1 required positional argument: 'board'
```

Image 12: _get_game_handlers() function



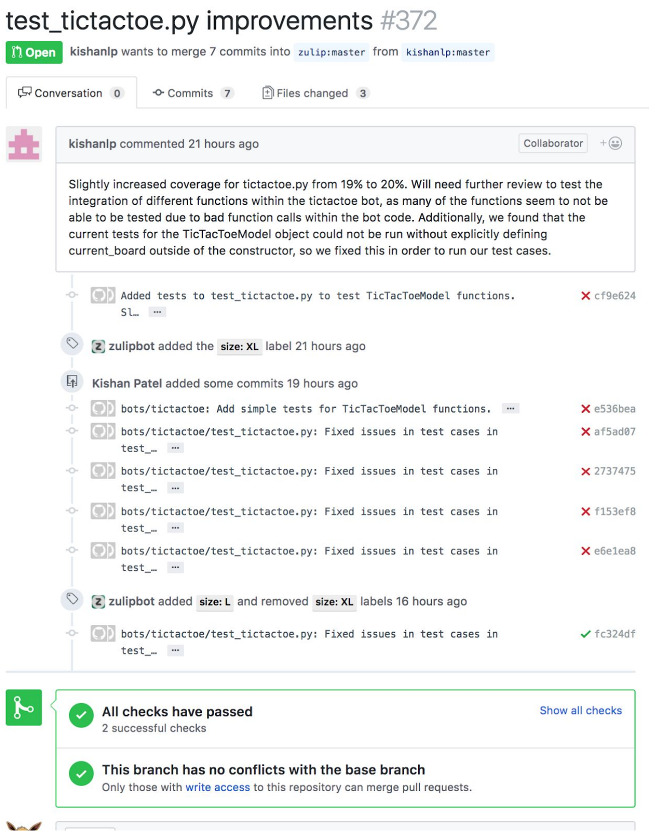
```python
    def _get_game_handlers(self) -> Tuple[Any, Any]:
        bot, bot_handler = self._get_handlers()
        return bot.model, bot.gameMessageHandler
```

Image 13: Some of our finalized test cases that we wrote that initialized an instance of model in order to create an individual TicTacToeModel for each unit test. These were the tests that were submitted in our pull request.



```python
    def test_board_is_full(self) -> None:
        board = [[1, 0, 1],
                 [1, 2, 1],
                 [2, 1, 2]]
        response = False
        self._test_board_is_full(board, response)

    def _test_board_is_full(self, board: List[List[int]], expected_response: bool) -> None:
        model, message_handler = self._get_game_handlers()
        tictactoeboard = model(board)
        response = tictactoeboard.board_is_full(board)
        self.assertEqual(response, expected_response)

    def test_contains_winning_move(self) -> None:
        board = [[1, 1, 1],
                 [0, 2, 0],
                 [2, 0, 2]]
        response = True
        self._test_contains_winning_move(board, response)

    def _test_contains_winning_move(self, board: List[List[int]], expected_response: bool) -> None:
        model, message_handler = self._get_game_handlers()
        tictactoeboard = model(board)
        response = tictactoeboard.contains_winning_move(board)
        self.assertEqual(response, expected_response)
```

Image 14: A screenshot of our pull request and commit message, and evidence that all checks have passed, so the


test_tictactoe.py improvements #372

Open kishanlp wants to merge 7 commits into `zulip:master` from `kishanlp:master`

Conversation 0 · Commits 7 · Files changed 3

kishanlp commented 21 hours ago — Collaborator

Slightly increased coverage for tictactoe.py from 19% to 20%. Will need further review to test the integration of different functions within the tictactoe bot, as many of the functions seem to not be able to be tested due to bad function calls within the bot code. Additionally, we found that the current tests for the TicTacToeModel object could not be run without explicitly defining current_board outside of the constructor, so we fixed this in order to run our test cases.

```
Added tests to test_tictactoe.py to test TicTacToeModel functions.       ✗ cf9e624
Sl…
```

zulipbot added the size: XL label 21 hours ago

Kishan Patel added some commits 19 hours ago

```
bots/tictactoe: Add simple tests for TicTacToeModel functions.           ✗ e536bea
bots/tictactoe/test_tictactoe.py: Fixed issues in test cases in          ✗ af5ad07
test_…
bots/tictactoe/test_tictactoe.py: Fixed issues in test cases in          ✗ 2737475
test_…
bots/tictactoe/test_tictactoe.py: Fixed issues in test cases in          ✗ f153ef8
test_…
bots/tictactoe/test_tictactoe.py: Fixed issues in test cases in          ✗ e6e1ea8
test_…
```

zulipbot added size: L and removed size: XL labels 16 hours ago

```
bots/tictactoe/test_tictactoe.py: Fixed issues in test cases in          ✓ fc324df
test_…
```

All checks have passed
2 successful checks — Show all checks

This branch has no conflicts with the base branch
Only those with write access to this repository can merge pull requests.

XI. Advice for Future Students

In order to succeed in this class, you will have to actively apply the concepts you learn in lecture to the projects and properly plan your course of action when tackling how to debug extremely large codebases in a timely manner.

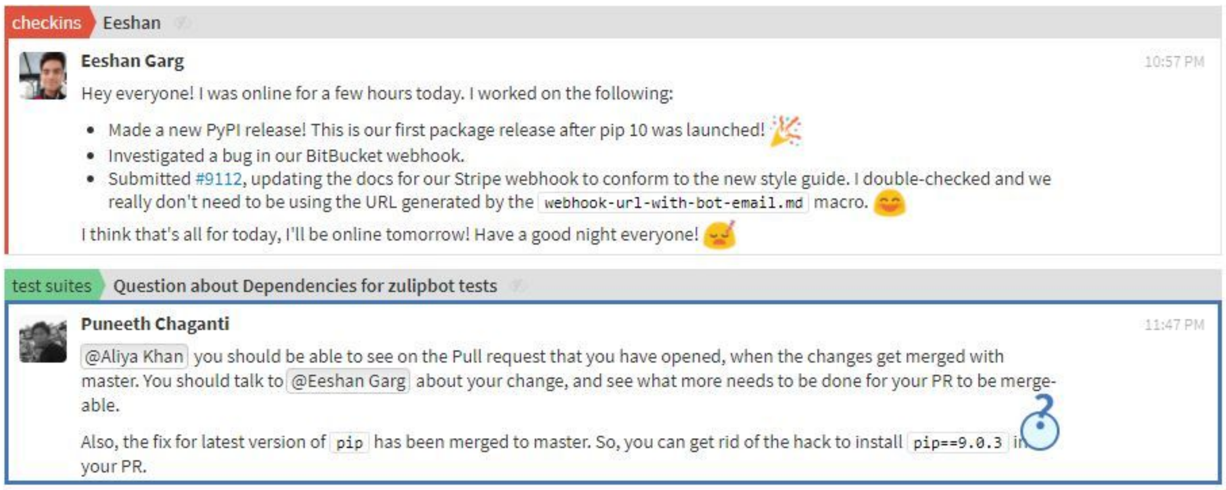
In order to succeed in other aspects of the separate from projects, do the assigned readings before every lecture and take notes as this is an excellent supplement to the lecture material and will make it easier for you to study for and succeed on the exam.

(You may use either of our materials in the future semesters)

XII. Accepted Changes

The bug that we found and fixed locally before telling another developer about the issue was submitted as a separate issue. Below is the associated PR, and this was merged into the master branch. We were not allowed to claim this issue because as new contributors, we are only allowed to claim 1 issue at a time. Even though we found how to fix this, we were not able to author the commit that ultimately changed this code. However, we believe that if we were allowed to claim more than 1 issue to contribute to, then we could have submitted and merged into master. In addition, our current pull request has passed the PR tests and is being reviewed by developers to be merged into the codebase. \

Image: this is proof that our suggested fix for the pip version has been accepted and that our PR for test-coverage is waiting to be merged into the codebase



checkins Eeshan

Eeshan Garg 10:57 PM

Hey everyone! I was online for a few hours today. I worked on the following:

- Made a new PyPI release! This is our first package release after pip 10 was launched!
- Investigated a bug in our BitBucket webhook.
- Submitted #9112, updating the docs for our Stripe webhook to conform to the new style guide. I double-checked and we really don't need to be using the URL generated by the `webhook-url-with-bot-email.md` macro.

I think that's all for today, I'll be online tomorrow! Have a good night everyone!

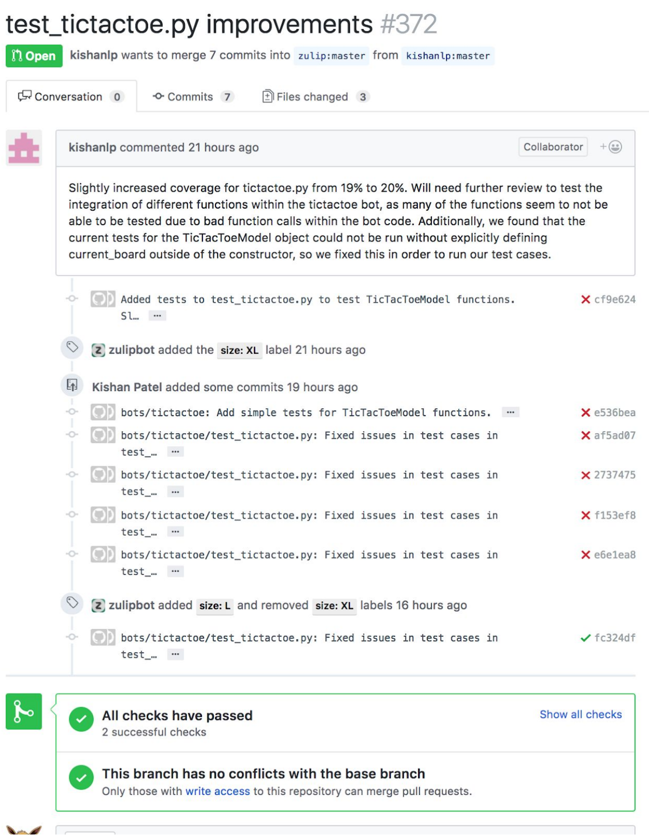
test suites Question about Dependencies for zulipbot tests

Puneeth Chaganti 11:47 PM

@Aliya Khan you should be able to see on the Pull request that you have opened, when the changes get merged with master. You should talk to @Eeshan Garg about your change, and see what more needs to be done for your PR to be merge-able.

Also, the fix for latest version of `pip` has been merged to master. So, you can get rid of the hack to install `pip==9.0.3` in your PR.

Image: Below is the image that shows that the issue we reported and fixed was submitted by another developer on the Zulip project because we are only allowed to contribute to 1 issue at a time. This is why we were not able to submit this issue below.



test_tictactoe.py improvements #372

Open kishanlp wants to merge 7 commits into `zulip:master` from `kishanlp:master`

Conversation 0 · Commits 7 · Files changed 3

kishanlp commented 21 hours ago · Collaborator

Slightly increased coverage for tictactoe.py from 19% to 20%. Will need further review to test the integration of different functions within the tictactoe bot, as many of the functions seem to not be able to be tested due to bad function calls within the bot code. Additionally, we found that the current tests for the TicTacToeModel object could not be run without explicitly defining current_board outside of the constructor, so we fixed this in order to run our test cases.

```
Added tests to test_tictactoe.py to test TicTacToeModel functions.      ✗ cf9e624
Sl… …
```

zulipbot added the **size: XL** label 21 hours ago

Kishan Patel added some commits 19 hours ago

```
bots/tictactoe: Add simple tests for TicTacToeModel functions. …        ✗ e536bea
bots/tictactoe/test_tictactoe.py: Fixed issues in test cases in         ✗ af5ad07
test_… …
bots/tictactoe/test_tictactoe.py: Fixed issues in test cases in         ✗ 2737475
test_… …
bots/tictactoe/test_tictactoe.py: Fixed issues in test cases in         ✗ f153ef8
test_… …
bots/tictactoe/test_tictactoe.py: Fixed issues in test cases in         ✗ e6e1ea8
test_… …
```

zulipbot added **size: L** and removed **size: XL** labels 16 hours ago

```
bots/tictactoe/test_tictactoe.py: Fixed issues in test cases in         ✓ fc324df
test_… …
```

All checks have passed · Show all checks
2 successful checks

This branch has no conflicts with the base branch
Only those with write access to this repository can merge pull requests.