coala: Improve the coala CLI

Student info

- Name: Maxim, Alexandru-Sorin
  Github username: @Nosferatul
  Link to my github: https://github.com/Nosferatul
- Email: alexandrus.maxim@gmail.com
- Telephone: +40727569950
- Time Zone: +02:00 GMT, Bucharest, Romania
- GSoC Blog RSS Feed URL:
  https://gsocsite.wordpress.com/2017/03/25/google-summer-of-code/

- Proposal Abstract

In this project we are going to focus on the command line interface(CLI) of coala. Areas and points to focus on are: allow chaining actions in a single input, add a 'select action for all results' operator, reprompt after running actions only if the action says so.

Code Sample

- My commits to the **coala** repository are here:
https://github.com/coala/coala/commits/master?author=Nosferatul
https://github.com/coala/coala/pull/2778
https://github.com/coala/coala/pull/2015
- My commits to the **coala-bears** repository:
https://github.com/coala/coala-bears/pull/732
- My commits to the **coala-documentation** repository:
https://github.com/coala/documentation/commits/master?author=Nosferatul
- My commits to the **coala-html** repository:
https://github.com/coala/coala-html/pull/39

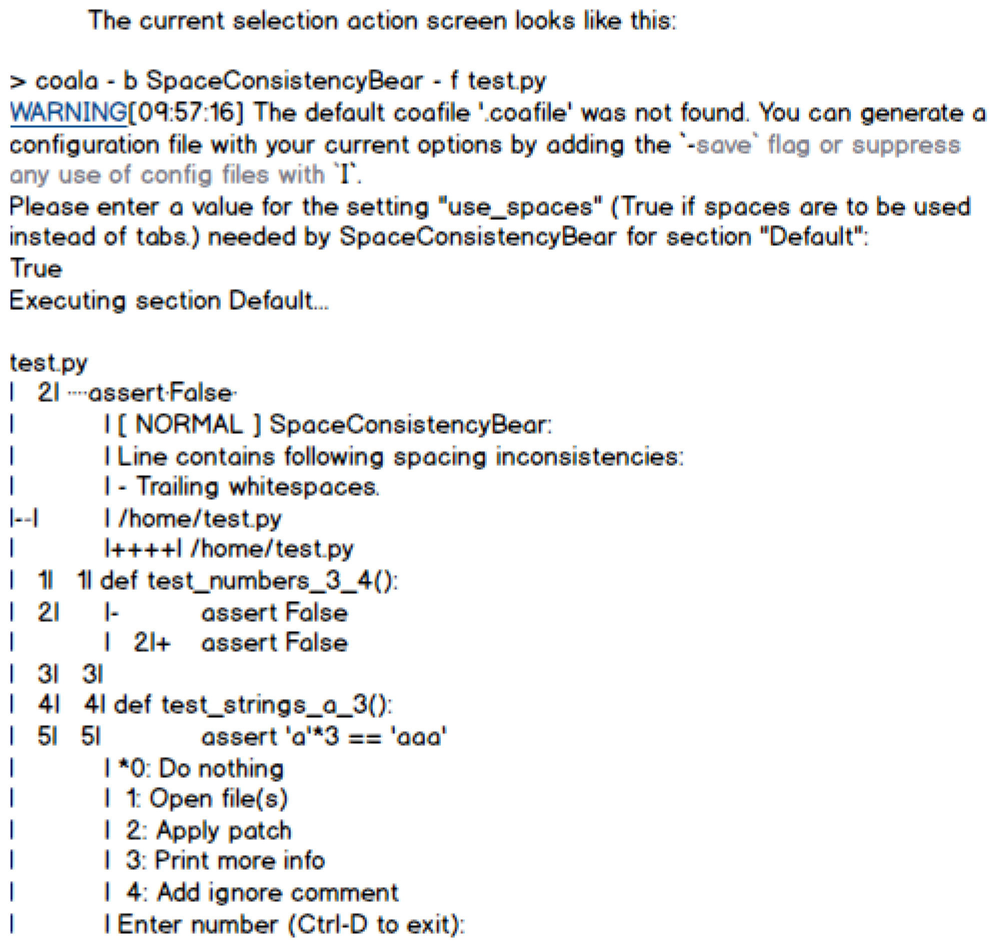
Project info

- **Which project from http://projects.coala.io are you applying for?**
  Improve the coala CLI:
  http://projects.coala.io/#/projects?project=improve_the_coala_cli

- **How is your project helping coala and its community?**

  Make coala easier to use, hence the project is going to rely heavily upon mockups and feedback from the community. Adds new functionality so user could have multiple options when uses coala. The user interface would help coala become much more user friendly and intuitive.

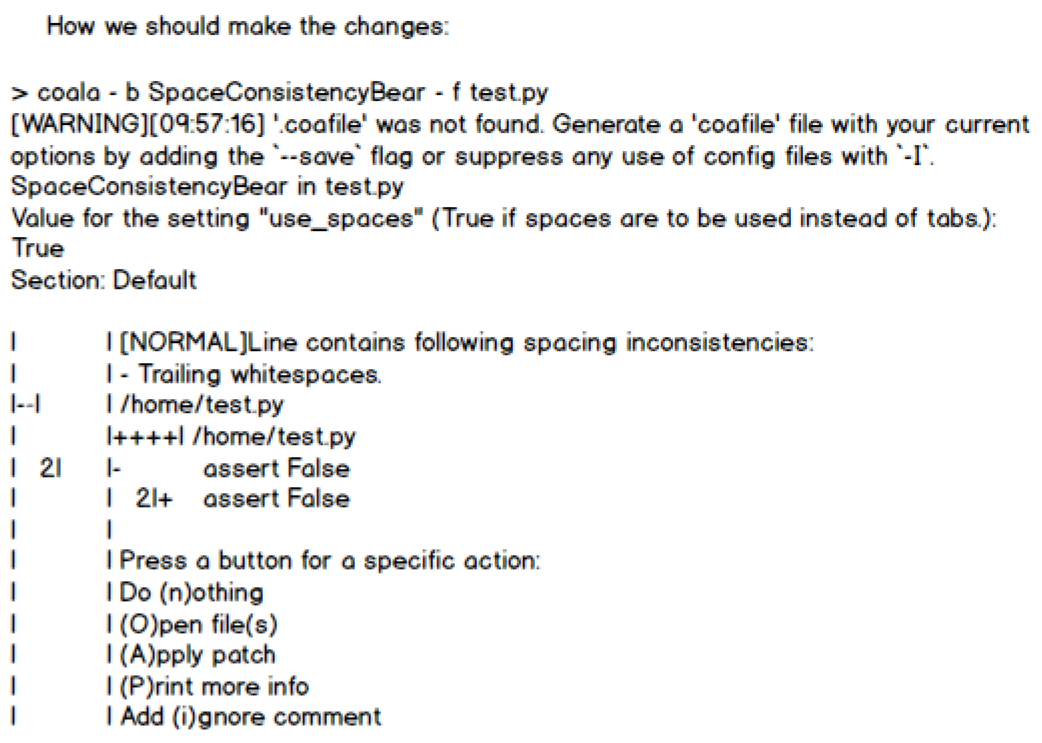
  Mockups:



The current selection action screen looks like this:

```
> coala - b SpaceConsistencyBear - f test.py
WARNING[09:57:16] The default coafile '.coafile' was not found. You can generate a
configuration file with your current options by adding the `-save` flag or suppress
any use of config files with `I`.
Please enter a value for the setting "use_spaces" (True if spaces are to be used
instead of tabs.) needed by SpaceConsistencyBear for section "Default":
True
Executing section Default...

test.py
|   2| ····assert·False·
|        | [ NORMAL ] SpaceConsistencyBear:
|        | Line contains following spacing inconsistencies:
|        | - Trailing whitespaces.
|--|     | /home/test.py
|        |++++| /home/test.py
|   1|   1| def test_numbers_3_4():
|   2|     |-      assert False
|        |   2|+    assert False
|   3|   3|
|   4|   4| def test_strings_a_3():
|   5|   5|        assert 'a'*3 == 'aaa'
|        | *0: Do nothing
|        |  1: Open file(s)
|        |  2: Apply patch
|        |  3: Print more info
|        |  4: Add ignore comment
|        | Enter number (Ctrl-D to exit):
```

After 1 week:



How we should make the changes:

```
> coala - b SpaceConsistencyBear - f test.py
[WARNING][09:57:16] '.coafile' was not found. Generate a 'coafile' file with your current
options by adding the `--save` flag or suppress any use of config files with `-I`.
SpaceConsistencyBear in test.py
Value for the setting "use_spaces" (True if spaces are to be used instead of tabs.):
True
Section: Default

|        | [NORMAL]Line contains following spacing inconsistencies:
|        | - Trailing whitespaces.
|--|     | /home/test.py
|        |++++| /home/test.py
|  2|    |-        assert False
|        |  2|+    assert False
|        |
|        | Press a button for a specific action:
|        | Do (n)othing
|        | (O)pen file(s)
|        | (A)pply patch
|        | (P)rint more info
|        | Add (i)gnore comment
```

- **What is the final goal for this project? What would make it a total and perfect success?**
  Adds functionality alongside with a better interaction with the user. The final goal will be when coala CLI has a new “face”: more friendly, easy to read, more functionality depending on what user wants!

- **Are you already engaged with the project's possible mentors and do you have any preference for a particular mentor?**
  I reviewed https://github.com/coala/cEPs/pull/29 with possible solutions for different aspects of coala CLI.
  No, I don’t have preference for a particular mentor.

- **What parts of coala do you have to work with in order to complete this project? What else are you planning on using?**
  I’ll work with coala core and CLI, especially with ConsoleInteraction and Processing.

- **Why are you the right person to work on this project?**
  I am very familiar with the core code, most of my patches are in coala repository. Also I implemented Ignore Action, so I know well the bounding between files from the core and CLI.

- **How do you plan to achieve completion of your project?**
  Working hard. Split each task into subtasks and properly solve them. Constantly analyzing the changes made and keep track of the optimization made so far. Also tests, and documentation will be added so that the code doesn't break other functionalities.
  Milestones:
    - Remove the unnecessary text from the action selection screen.
    - User letters instead of numbers for selecting actions. In this way, the selection is more user-friendly.
    - Allow chaining actions in a single input so that user could choose between applying multiple actions for different sections.
    - Make the implementation of ApplyPatchAction log the applied patch
    - Reprompt after running multiple actions.
    - Implement "Interactive patches"
    - Provide alternative diffs if it's possible

- ## Proposal Detailed Description/Timeline

Community Bonding (May 4 - May 30)

The four major aspects that will be covered in this period are:

1. Update cEP-0004, I already reviewed it with my ideas on how can we improve the CLI.
2. I plan to use a form to get feedback from the community explaning how the CLI should look like. Adds functionality alongside with a better interaction with the user.
3. Make changes based on the feedback. At last, merge cEP-0004 with all tehnical details.
4. I want to start by improving the coala Diff handler. Why? Because we will need some functionalities like: provide alternative diffs(if it's possible), we will need later on when I'll start working on "Interactive patches".

Working on the project (May 30 - August 29)

- **Week 1 (May 30 - 5 June)**
  In Week 1 I am planning on using the knowledge accumulated so far on starting by improving text output from the action selection screen based on the feedback. From now on, we will use letters instead of numbers for selecting actions. In this way, the selection will be more user-friendly. Also, all of the above will be updated every time when I'm merging on master branch. This is not so hard we will shrink the

messages/errors from the “result” directory. As I said, the user could just press the key for an action to happen. We could use “pynput.keyboard.Listener” (http://pythonhosted.org/pynput/keyboard.html) and just make a function that “convert” a pressed button to an int for “actions” in ConsoleInteration.

- **Week 2 (June 6 - 13 June)**
  In Week 2 I want to start implementing “interactive patches”. If the user wants, we will propose some parameters(e.g. variable renaming, code indentation) depending on the output of Diff that doesn’t matter of bears that user is using. Also, we could have some dependecies. Let’s say that a user wants ‘indentation= 4 spaces’ and wants to be applied on a file that has a line at 78 without indentation. First apply the indentation variable and then use our bear to repare this. If a bear from ones that we want to run by default, generates a diff, propose a param and apply it.

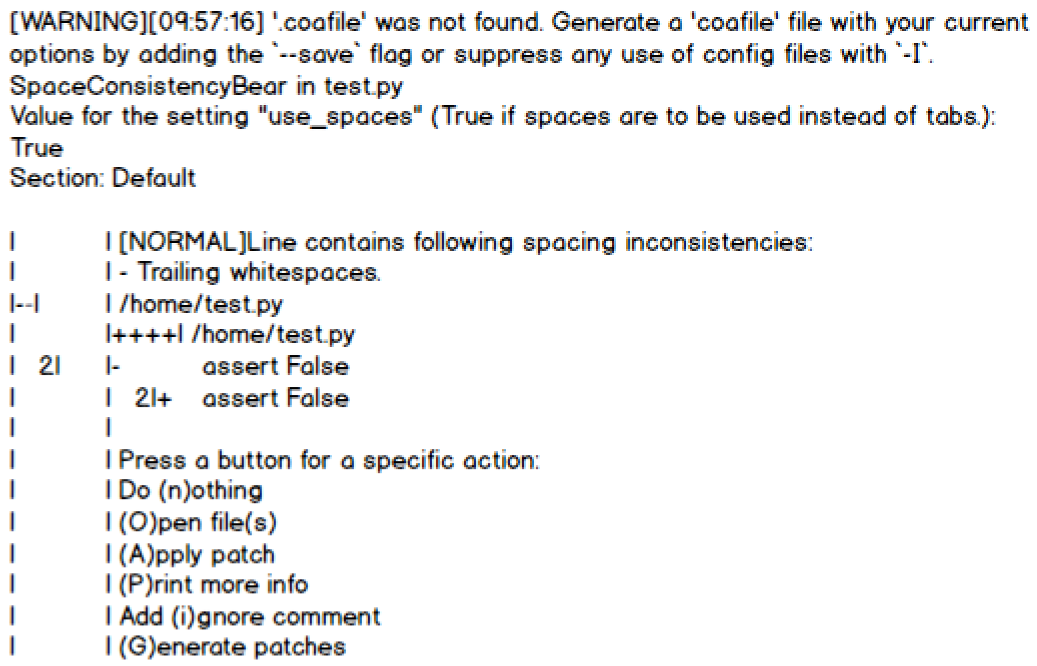
- **Week 3 (June 14 - 21 June)**

  In this week I’ll finish implementing “interactive patches”. For this I want to create “GenerateActions.py”. All of our actions have “is_applicable” and “apply” methods. Because we want to generate more than 1 action every time, we will propose the parameters and with the bears, will apply that parameter. Let’s say that a default interactive action will be ‘trailing whitespaces’. If the user wants this, SpaceConsistencyBear will make the changes automatically(like a use of ApplyPatchAction with SpaceConsistencyBear by default on the specific file). So, we could use ApplyPatchAction multiple times over a file.

- **Week 4 (June 14 - 21 June)**

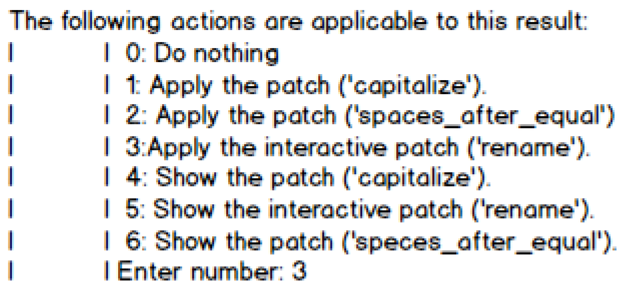
  In Week 4 I’ll start to implement “interactive patches” for the bears that we could use to generate the actions(not by default). We can generate more actions by using the bears(for the specific type of programming language present in the file). We will try to run the bears on that file, depending on what bears yield a diff to be made, we generate specific actions. Also I’ll make a new action “(G)enerate actions”. We could make bears to generate actions. So, if XBear provide a diff, it will generate an Xaction available for user.

Mockups



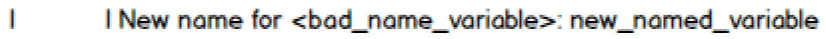
```
[WARNING][09:57:16] '.coafile' was not found. Generate a 'coafile' file with your current
options by adding the `--save` flag or suppress any use of config files with `-I`.
SpaceConsistencyBear in test.py
Value for the setting "use_spaces" (True if spaces are to be used instead of tabs.):
True
Section: Default

|        | [NORMAL]Line contains following spacing inconsistencies:
|        | - Trailing whitespaces.
|--|     | /home/test.py
|        |++++| /home/test.py
|   2|   |-        assert False
|        |   2|+   assert False
|        |
|        | Press a button for a specific action:
|        | Do (n)othing
|        | (O)pen file(s)
|        | (A)pply patch
|        | (P)rint more info
|        | Add (i)gnore comment
|        | (G)enerate patches
```

Let's say that user, in this case press "G" button. The next selection screen will look like:



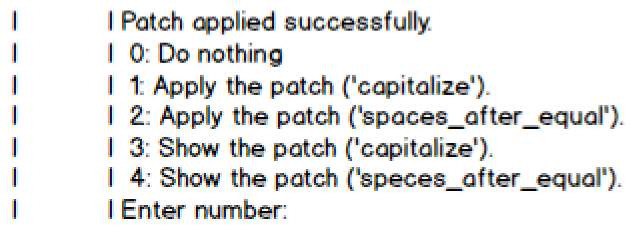
```
The following actions are applicable to this result:
|        |  0: Do nothing
|        |  1: Apply the patch ('capitalize').
|        |  2: Apply the patch ('spaces_after_equal')
|        |  3:Apply the interactive patch ('rename').
|        |  4: Show the patch ('capitalize').
|        |  5: Show the interactive patch ('rename').
|        |  6: Show the patch ('speces_after_equal').
|        | Enter number: 3
```

The user wants to apply "interactive patch('rename')". Interactive patch means that the user will be asked to introduce a value. I want to use numbers because we could have many applicable actions. For this interactive patch, the user is asked to enter a new variable name for the bad one.



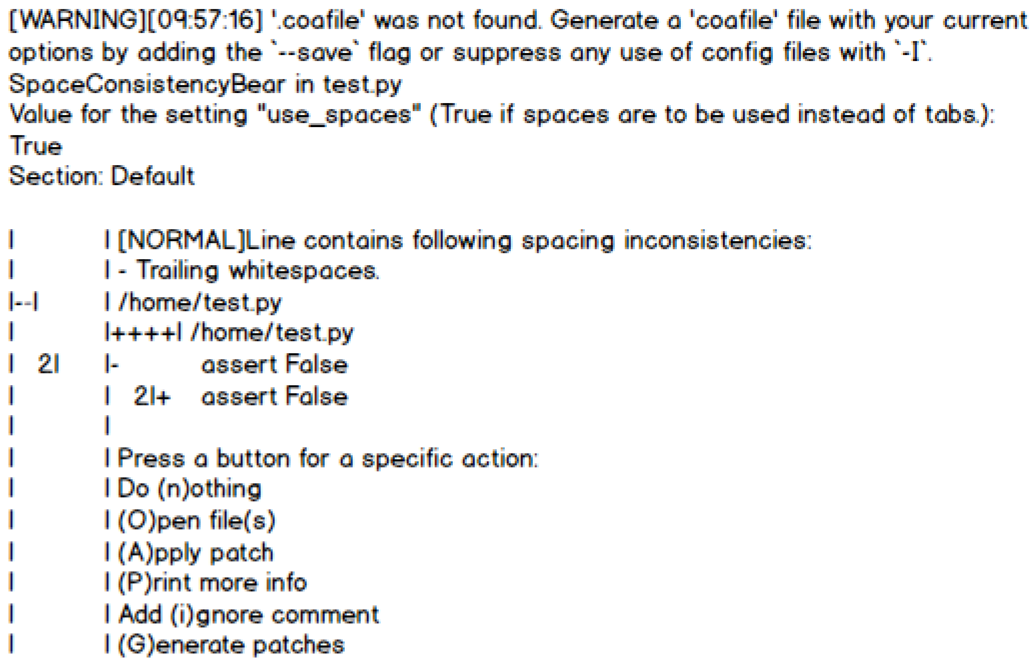
```
|        | New name for <bad_name_variable>: new_named_variable
```

After the patch is applied:



```
|          | Patch applied successfully.
|          |  0: Do nothing
|          |  1: Apply the patch ('capitalize').
|          |  2: Apply the patch ('spaces_after_equal').
|          |  3: Show the patch ('capitalize').
|          |  4: Show the patch ('speces_after_equal').
|          | Enter number:
```

If the user enter “0” we will reprompt the possible actions:



```
[WARNING][09:57:16] '.coafile' was not found. Generate a 'coafile' file with your current
options by adding the `--save` flag or suppress any use of config files with `-I`.
SpaceConsistencyBear in test.py
Value for the setting "use_spaces" (True if spaces are to be used instead of tabs.):
True
Section: Default

|          | [NORMAL]Line contains following spacing inconsistencies:
|          | - Trailing whitespaces.
|--|       | /home/test.py
|          |++++| /home/test.py
|  2|      |-         assert False
|          |  2|+    assert False
|          |
|          | Press a button for a specific action:
|          | Do (n)othing
|          | (O)pen file(s)
|          | (A)pply patch
|          | (P)rint more info
|          | Add (i)gnore comment
|          | (G)enerate patches
```

Also, we will have default “patches” that will be generated at every run. In the first phase, I will focus on the implementation and structure of this new concept, it will be added just few patches in the beginning.

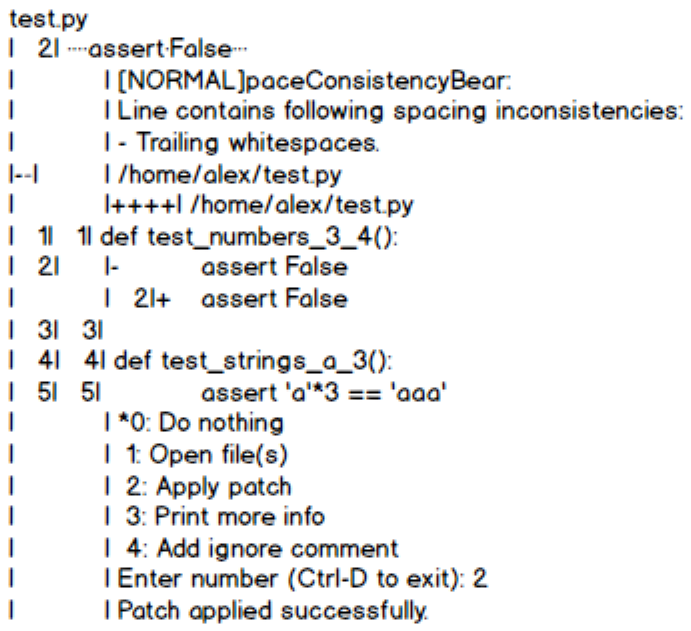
- **Week 5 (June 30 - 6 July)**

  I plan in Week 5 being a buffer week, where I will go over the code written so far, fixing any minor issues encountered. This is a **milestone** week, as I am planning on merging my work into the master branch until now.

- **Week 6 (July 7 - 14 July)**

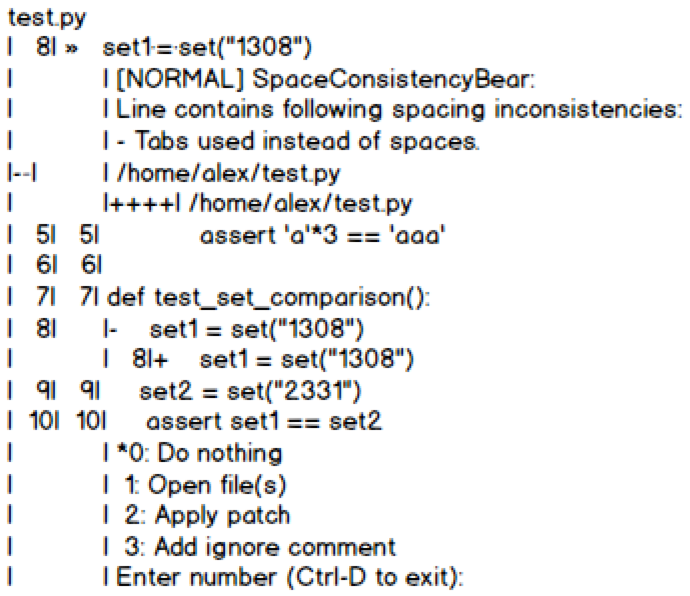
  In Week 6 I’m planing to start working on “reprompt after running actions”. This will work with “interactive patches”. As I explained with mockups, after the user press “0” in “Generate actions”, the last selection screen will be prompted with the applied patch(es). Here will use “reprompt after an actions”. We could just use call the “choose_action” until the user wants to exit.

- **Week 7 (July 15 - 21 July)**
  In Week 7 I want to implement the “select action for all result” operator. Will work like this: if a user wants to apply a patch over a file, will choose this option. In this scenario, a patch means changes made by a single bear over a file.
  Now, coala works like this: a file have multiple issues with (e.g) spacing inconsistencies but when “Apply patch” is used, the bear solve only the first issue encountered. With this option, it will solve all the issues in a file.
  Mockup. How works now (first apply):



```
test.py
|   2| ····assert·False···
|         | [NORMAL]paceConsistencyBear:
|         | Line contains following spacing inconsistencies:
|         | - Trailing whitespaces.
|--|      | /home/alex/test.py
|         |++++| /home/alex/test.py
|   1|   1| def test_numbers_3_4():
|   2|     |-        assert False
|         |   2|+    assert False
|   3|   3|
|   4|   4| def test_strings_a_3():
|   5|   5|          assert 'a'*3 == 'aaa'
|         | *0: Do nothing
|         |  1: Open file(s)
|         |  2: Apply patch
|         |  3: Print more info
|         |  4: Add ignore comment
|         | Enter number (Ctrl-D to exit): 2
|         | Patch applied successfully.
```

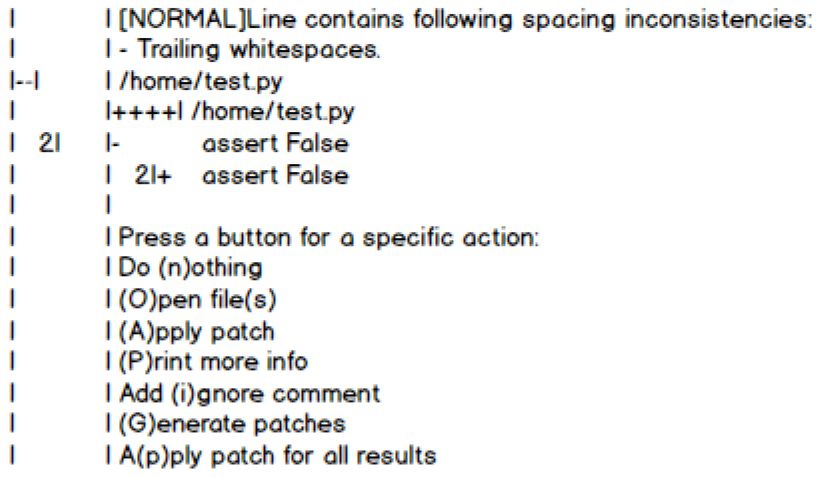
Second apply:



```
test.py
|   8| »   set1·=·set("1308")
|         | [NORMAL] SpaceConsistencyBear:
|         | Line contains following spacing inconsistencies:
|         | - Tabs used instead of spaces.
|--|      | /home/alex/test.py
|         |++++| /home/alex/test.py
|   5|   5|          assert 'a'*3 == 'aaa'
|   6|   6|
|   7|   7| def test_set_comparison():
|   8|     |-    set1 = set("1308")
|         |   8|+    set1 = set("1308")
|   9|   9|    set2 = set("2331")
|  10|  10|    assert set1 == set2
|         | *0: Do nothing
|         |  1: Open file(s)
|         |  2: Apply patch
|         |  3: Add ignore comment
|         | Enter number (Ctrl-D to exit):
```

The 1st apply, change only the line 2 and the second apply, line 8. We want something like this:

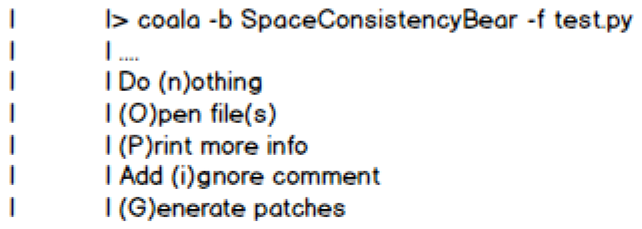
First run:



```
|          | [NORMAL]Line contains following spacing inconsistencies:
|          | - Trailing whitespaces.
|--|       | /home/test.py
|          |++++| /home/test.py
|   2|     |-          assert False
|          |   2|+    assert False
|          |
|          | Press a button for a specific action:
|          | Do (n)othing
|          | (O)pen file(s)
|          | (A)pply patch
|          | (P)rint more info
|          | Add (i)gnore comment
|          | (G)enerate patches
|          | A(p)ply patch for all results
```

User choose “Apply patch for all results”. This will apply all changes yield by the bear used.

Second run:



```
|          |> coala -b SpaceConsistencyBear -f test.py
|          | ....
|          | Do (n)othing
|          | (O)pen file(s)
|          | (P)rint more info
|          | Add (i)gnore comment
|          | (G)enerate patches
```

The changes are applied over all content of a file.

How? We have function “ask_for_action_and_apply” in the “ConsoleInteraction.py”. We could call it for a single fil until “file_diff_dict” is empty for that file.

- **Week 8 (July 22 - 29 July)**

In Week 8 I will test everything written so far, and fix the possible mistakes or bugs that I may have encountered. This way, by the end of Week 7 we will get clear running code and automated packages. This is another **milestone** week, where my tests will be committed and merged to the master branch. Also, I'll make changes based on the feedback from the form and members.

- **Week 9 (July 30 - 5 August)**

  In this week I'll make "Apply patch for all results" to work for interactive patches. In the first instance it will make all changes by default. So when a user choose an action from the generated ones, that action will be applied over the file, by default. Depending on the feedback from the community, I'll change that by asking the user every time chooses an generated action, if wants to be chained.

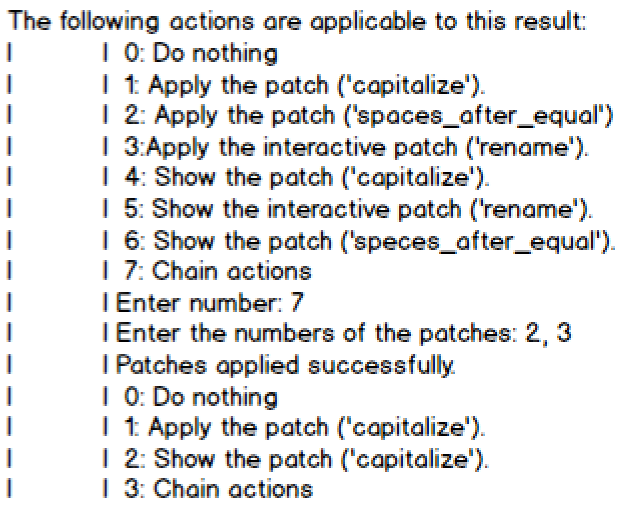
- **Week 10 (August 6 - 12 August)**

  In Week 10 I want to add more interactive patches, the user will have more options to choose from. Also, I'll write tests to check all possible combinations of actions, and check if the files are fixed properly. I will test everything so far on both my Windows OS and my Linux distribution.

- **Week 11 (August 13 - 19 August)**

  In this week. I will start working on chaining actions, so users could apply more than one patch in a single input. This will work with interactive patches and simple actions. How? The function from "ConsoleInteraction.py", "choose_action" works only with a single action. We could change that by making "choose_action" to return a string of choices. After, just apply the actions that user wants. Also, the function "ask_for_action_and_apply" will be called for every action in the string.

  Mockup:



```
The following actions are applicable to this result:
|           |  0: Do nothing
|           |  1: Apply the patch ('capitalize').
|           |  2: Apply the patch ('spaces_after_equal')
|           |  3:Apply the interactive patch ('rename').
|           |  4: Show the patch ('capitalize').
|           |  5: Show the interactive patch ('rename').
|           |  6: Show the patch ('speces_after_equal').
|           |  7: Chain actions
|           | Enter number: 7
|           | Enter the numbers of the patches: 2, 3
|           | Patches applied successfully.
|           |  0: Do nothing
|           |  1: Apply the patch ('capitalize').
|           |  2: Show the patch ('capitalize').
|           |  3: Chain actions
```

- **Week 12** (August 20 - August 23)

  Week 12 is also supposed to be a buffer week. On this week I will check all the TODOs marked on my code, and also fix my code-style so that it is easy for others to read in the future. I'll add documentation for my code and make for the last time changes based on the feedback.

Other commitments

- Do you have any other commitments during the main GSoC time period, May 23rd to August 23rd?
  - Do you have exams or classes that overlap with this period?
    Yes, I will have exams until 17th June, I'll need some time to prepare for exams. From 27th May to 17th June I am willing
    to work on the project 30-35 hours per week. Maybe more, depending
    on the difficulty of the exams on that week.
  - Do you plan to have any other jobs or internships during this period?
    No, I am planning on devoting entirely to this project.
  - Do you have any other short term commitments during this period?
    Not yet, however I am willing to go on a 5-7 days trip to the seaside, probably around July or August, but I have not decided the date yet. I will make the other vacation plans in September, so they do not interfere with my work.

- Have you applied with any other organizations? If so, do you have a preferred project/org?
  - No

Extra information

- Link to my resume:
  https://drive.google.com/file/d/0B7N48tqMppTQTjhtcTA1Z0V0ck0/view?usp=sharing

- University info
  - University Name: Faculty of Automatic Control and Computer Science, Polytechnic University of Bucharest

    - Major: Computer Science
    - Current Year and Expected Graduation date: Year 2. Expected Graduation Date is 2019
    - Degree (e.g. BSc, PhD): Bachelor of Science in Computer Science
- Other Contact info:
    - Alternate contact info in case your primary email above stops working: I have an alternate email I'm constantly using, mxm_alexandru@yahoo.com
    - Instant messaging: Im always available on https://www.facebook.com/maxim.alexandru.58
    - Twitter:
- Will you be able to join us for a conference such as EuroPython and present your work if at least partial sponsorship will be provided?

  Yes, I am looking forward to going to EuroPython and other conferences.
- We love having twitter handles so we can tell people publicly about your great project and successes at https://twitter.com/coala_analyzer!

  I have actually been sending links to my coala achievements that were posted on twitter to my friends, and also recommended them to use coala.