Debug and Profile Bears

Student Info

- Name : Meet Mangukiya
- GitHub Username: @meetmangukiya
- Alternative-/Nickname: Freenode IRC: mngkya
- Email: meet123mangukiya@gmail.com
- Which country will you reside in during this project?
    - India
- Which city, region or geographical boundary?
    - Mumbai, Maharashtra, India
- Time Zone: GMT +5:30
- GSoC blog RSS feed URL: https://meetmangukiya.github.io/index.xml

Code Sample

coala	Cool: https://github.com/coala/coala/pull/2895 https://github.com/coala/coala/pull/3610 - Yet to be merged All Merged: https://github.com/coala/coala/commits/master?author=meetmangukiya
coala-bears	Cool: https://github.com/coala/coala-bears/pull/877 https://github.com/coala/coala-bears/pull/1356 - Yet to be merged All Merged: https://github.com/coala/coala-bears/commits/master?author=meetmangukiya
Open PRs	https://github.com/pulls?utf8=%E2%9C%93&q=is%3Aopen+is%3Apr+author%3Ameetmangukiya+user%3Acoala
Other than coala	I have a patch at go-gitter https://github.com/sromku/go-gitter/pull/13

Project Info

1. **Which project from https://projects.coala.io are you applying for?**

   Debug And Profile Bears

2. **How is your project helping coala and its community?**

   After the completion of this project:

   a. Debugging bears will become as easy as writing new bears.
   b. Debugging bears is one of the inevitable tasks of developing new bears. But debugging is hard, so to aid our developers and prevent them from dropping an ongoing development of bear out of frustration, a new pdb adapted to coala is needed.
   c. This project aims at building a debugger for coala-bears, and hence relieving coalaians of all the bugs that'd be rather difficult and time-consuming to spot and fix.

3. **What is the final goal for this project? What would make it a total and perfect success?**

   The final goal of this project is to create a debugger for coala bears that is easy to use. It is integrated in coala, so the debugger is just one argument away. Getting new developers to use the debugger and make them like debugging would make this project a total and perfect success. Broadly speaking, the python debugger (pdb) will be integrated and adapted for coala.

   The debugger will be capable of debugging functions in a way that:

   - Only arguments and return values of called function is captured.
   - Debug step by step into a function.
   - Create a report that'll contain the return values and arguments passed to the functions. The user wouldn't have to interact with the debugger in

this case. This can be used to get to know where the things are going wrong.

How does one use it?

Multiple ways:

- `coala --debug-bear -b bear -f file -S ...` and the debugger will be started with an instantiated bear, and with the files and settings provided.
- `coala --debugger` to launch the debugger and provide all the required arguments for instantiation interactively.
- `coala --debug-report -b bear -f files -S ...` and get a debug report(report mode is described in Implementation of Debugger section) for that particular bear with provided settings.

4. Are you already engaged with the project's possible mentors and to you have any preference for a particular mentor?

I've discussed with Mischa(@Makman2) about the project, so I'd like to have him as primary mentor.

5. What parts of coala do you have to work with in order to complete this project? What else are you planning on using?

- This project will require work in the coala core repository, i.e. coala/coala.
- This project will also use the pdb/bdb standard library module.

6. Why are you the right person to work on this project?

I've been part of the coala community since September 2016 and been actively involved since then. I've helped newcomers to get aboard. I've reviewed PRs. And I am fairly familiar with the code base. I have also written cobot scripts, I love to

play with it. Apart from all this, I am really interested in this project, creating an actual debugger.
I haven't done any python debugging project in the past. But I've developed bears and while writing tests, I had to do debugging and I had to use pdb and it took me time to get acquainted with pdb.

I know the difficulties new developers face and would like to make debugging the bears better.

https://github.com/coala/coala-bears/pull/1356 is a type/bug that I fixed, which is yet to be merged.

https://github.com/coala/coala-bears/pull/877 is about adding a new feature to an existing bears.

https://github.com/coala/coala-bears/pull/1068 This is the PR of a bear that I authored.

These PRs show my ability to debug and fix bugs.

Also, I can use pdb and went through the documentation.

7. How do you plan to achieve completion of your project?

Abstract

1. **To be able to inspect the settings passed to the bear.**
   The debugging environment has to be setup internally, if an argument like --debug-bear is passed.
2. **To be able to inspect the results yielded by the bear.**
   Ability to play with the result object.
3. **To be able to inspect the arguments passed to a function and the return values from functions. To be able to debug any function normally (like pdb).**
   This is what I am calling a superficial mode. When executing, if a function is called, user is provided with two options:

a. If he wishes to debug it superficially, i.e. just retrieve the return values and arguments passed to the functions.
b. If he wishes to debug it completely, i.e. normally how it is debugged into pdb.

4. <u>To be able to create a report of return values and arguments passed to all the function calls(excluding stdlib functions) in a single step.</u>

   This can be considered to determine, if debugging is required. I'll first try to create the report. Since the report will have the return values of all the functions, it will be easy to spot which function is not behaving correctly and has to be debugged.

5. <u>To be able to profile run time of a bear.</u>

   Here, we will be using the inbuilt python profiler to profile `run()` method of bears provided the arguments gathered from the CLI.

Implementation

We are aiming for a modular design of debugger, so we can provide different ways in which it can be used.

1. Debugger will be called from command line arguments.
2. Collect and import the bear being debugged.
3. Collect the settings(S) for the given bear from the coafile or the command line arguments.
4. Guesser will guess the settings *needed* for instantiation of bear(i.e. Args to passed to __init__ method of the bear.). Note that these settings are different from the settings collected in step 3(S). `S` are the settings that are passed as kwargs to the run method.
5. Call Instantiator with the settings and the bears collected which will return an Instantiated Bear and the settings for run method.
6. Debug method is called, with the instantiated bear and the settings, which will call `bear.run` with those settings in debug mode.

<u>Debugger</u>

- <u>Instantiator</u>: # No User Interaction
  - This function will instantiate a given bear with given settings.
- <u>Collector</u>: # No User Interaction
  - This function will collect settings from the CLI or coafile, collect the bear being debugged and pass those to the Instantiator.
- <u>Guesser</u>: # Somewhat User Interaction
  - This function will guess sane defaults for settings that are not provided for a given bear. To be used when a setting is not obtained from collector but is

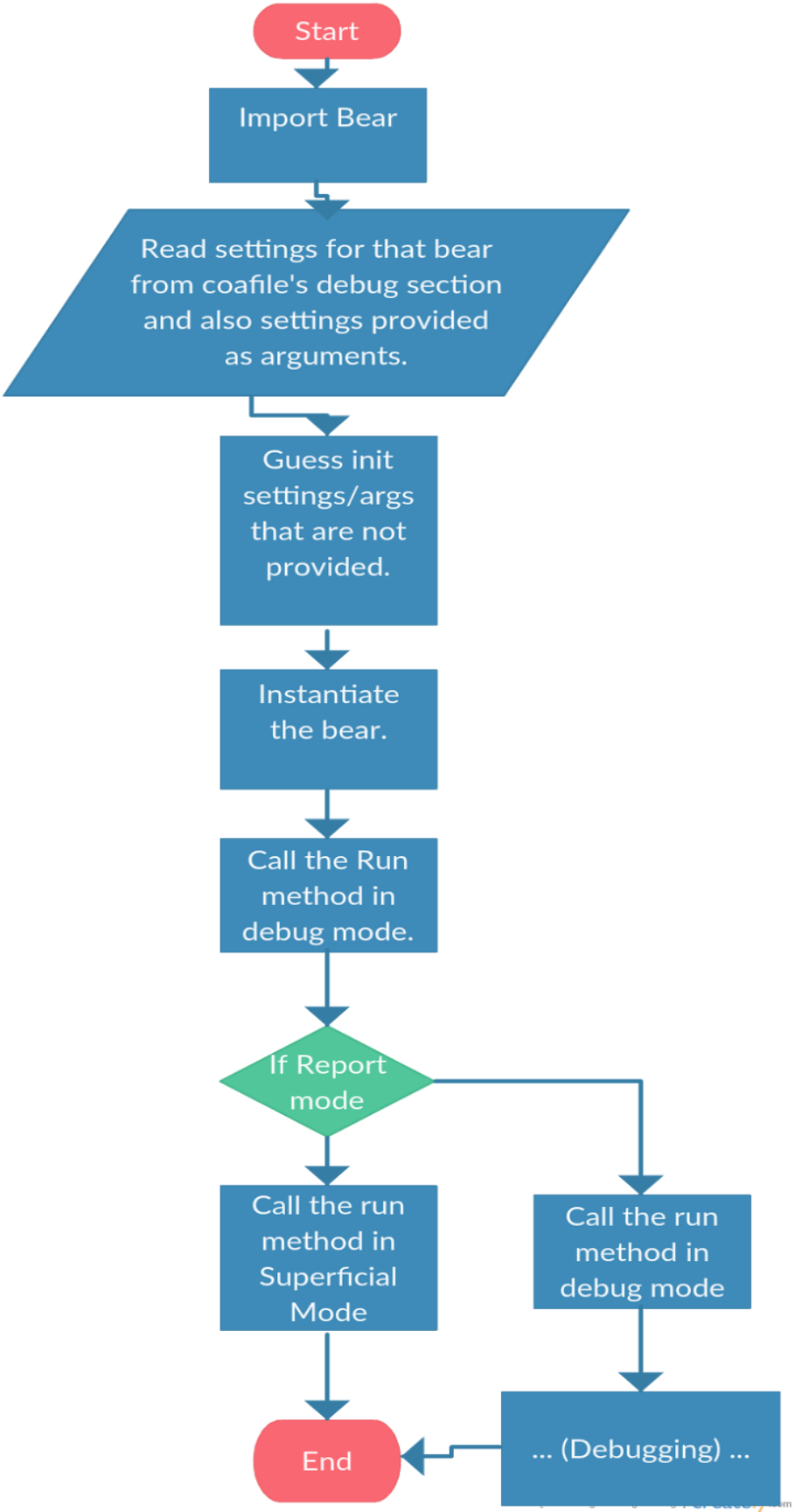
necessary for instantiation. If the Guesser is not able to guess a setting it will prompt the user to provide the setting.

- Debug: # High User Interaction
    - To start the debugging process, by calling the run method in either of the:
        - Debug mode:
          Debug mode allows to start debugging from that line forward the file as if it was being debugged in pdb.
            - If this is selected, then the bear will be run with a normal call to `pdb.run()`
        - Superficial mode:
          Superficial mode gives the information of arguments and return values of a function call.
            - If the run method itself is ran in superficial mode, then the results are yielded.
            - Else, the run method is ran in debug mode, and at all the subsequent function calls, the user is again given two options, superficial or debug.
        - Report mode:
          Report mode gives the information of all the function calls i.e. the return values and arguments provided.(excluding third party functions).
            - Here, the run method will be run in debug mode(silently) and subsequent function calls are run in superficial mode. Then the function name, function args and function return values are printed in form of a beautiful report.
            - Here, we need to implement a `judger` to judge whether a given function should be captured, since we need to escape the std library function calls or even third party function calls.
            - This is more like a consumer of superficial mode with added intelligence.

Existing functionality of coala can be used for the Guesser and the Collector with required modifications for creating the debugging environment.

This API of debugger will allow it be called in the following ways:

- `coala --debug-bear Bear -f file -S ...`
  Here, the Collector and Guesser will be used to get required information and the bear will be instantiated(Instantiator) and the debugging session will be started(Debug).
- `coala --debugger`
  This will start the debugger, and interactively get the settings, the files, etc. from the user(using the bear metadata) and pass on those settings to the Instantiator and debugging session will be started(Debug).


Start
Import Bear
Read settings for that bear from coafile's debug section and also settings provided as arguments.
Guess init settings/args that are not provided.
Instantiate the bear.
Call the Run method in debug mode.
If Report mode
Call the run method in Superficial Mode
Call the run method in debug mode
... (Debugging) ...
End

Profiler

Here, we are going to implement basic profiling.

Here, we are interested to profile runtime of the bears. We will be using the inbuilt python profiler(this can be changed/discussed during community bonding period) or the timeit module.

The profiling will start by profiling the `run()` method of the bears, which is where the logic of the bear lies and where the wrapped linter is run also. The advantage of profiling just the `run()`method of the bear is that, the time coala spends on loading the files, collecting the settings, etc. are not considered. Rather the performance of the bear individually is considered and formed the report of.

Usage:

`coala --profile -b Bear -f Files -S ...`

This will get the files, load the bears, collect the settings, instantiate the bears. Then the profiling starts at the call of the `run()` method.

Timeline

Community Period

Days	Milestones	Timeline Check
May 5 - May 18	- Formulate a cEP, get reviews, and get it merged. - Get Familiar with the processing and collecting part of coala. - Assess the alternatives for profiling and select one.	A merged cEP
May 19 - May 22	If some change has to be introduced in the proposed timeline (since I'll have more insight after the cEP is merged) then discuss it with the mentors and do the appropriate changes.	A new timeline with incorporated changes.
May 5 - May 30	Try to increase test coverage of pdb module and do some bug fixes in other python debuggers and get	

	better insight in the `inspect` and `pdb` module of python standard library.	

Coding Period

Week 1: May 30 - June 5	Debug mode: - Start execution of pdb from the given line of execution.
Week 2: June 6 - June 12 Week 3: June 13 - June 19	Superficial mode: - Identify and capture the arguments, also the name of the arguments.
	- Capture the return value of the function or the next function call.
Week 4: June 20 - June 26 Week 5: June 27 - July 3	Report mode: - Call the run method with debug mode. - And on the subsequent function calls in the run method should be called with the superficial mode. - Store the return value, argument info and funciton name. - Print the report beautifully in a tabular form.
	Write tests. Document it. Rigorous testing of the debugger.
Week 6: July 4 - July 10	Instantiator: - Instantiate a given bear, with the given settings(bear settings like Section, Queue etc.) - Return the instantiated bear. This function will not be directly called, it is more of a helper function. Collector: - Collect and load the bear provided in the CLI. - Collect the settings relevant to the bear from the coafile. - Collect the settings passed in the CLI.
Week 7: July 11 - July 17	Guesser:

	- If not all the information, required for bear instantiation is provided, and if sane settings can be guessed for the required setting, it will be guessed. Add the `--debug` argument. - This argument will spawn the debugger. - The user will be interactively be asked for required information to start the debugging process.
Week 8: July 18 - July 24	Add `--debug-bear` argument: - Call the Collector, Guesser and Instantiate the bear and call Debug on it.
Week 9: July 25 - July 31	Add `--profile` argument. - Instantiate the bear and profile the `run` method. Write tests. Document the Usage.
Week 10: Aug 1 - Aug 7	A better interface to inspect the results. - Override the existing `repr` of results to meet the debugging environment. - And use that `repr` for representing the results beautifully and completely.
Week 11: Aug 8 - Aug 14	More of a buffer period. Get reviews from fellow students about the UI and UX and bring about appropriate changes.
Week 12: Aug 15 - Aug 21	Create asciinemas demonstrating debugging of a bear. Inspection of results. Demonstrating debug, superficial, report mode of the debugger. Demonstrating the profile argument to inspect and optimise the bears.

Stretch Goals

1. Create a separate debugger package, so that everyone can benefit from the Debug Mode, Superficial Mode and the Report Mode.

Other Commitments

1. **Do you have any other commitments during the GSoC period, May 8th to August 29th?**
    a. **Do you have exams or classes that overlap with this period?**
        i. I will have my exams in the month of May.
        ii. I will have my vacations in the month of June and I'll try to work more in this period, including weekends. I hope to get most of the stuff done in this month.
        iii. My college will probably start from second or third week of July.
    b. **How do I plan to dedicate the time GSoC demands?**
        i. I'd like to start working on the project from the last two weeks of bonding period of GSoC if possible, since I've already been around for quite some time and know the community as well as coala.
        Let's crunch some numbers:
            1. There are 84 days between May 30 and Aug 21.
            2. That is equivalent to 12 weeks.
            3. One has to work 5 days a week, that means 60 days

        So if I start from may 18th(suppose) then I get 12 days headstart.

        If I work June(all days), that'll be 30 days.

        Assuming college starts in second week of July, that'll be 12 days.

        That adds up to 54 days, and still almost 6 weeks is left. So even If I work on weekends for next weeks, then it will give 12 days, an extra one week.

        But all this was given considering that I work each and every day from May 18 and not working on weekdays from July 3rd week. But that is obviously not true, I'll be working on weekdays as well and distribute the work little over the remaining period increasing productivity.

        So, given the above numbers it is approximately safe to assume that the possibilities about me getting time to complete my project is pretty good. But I'd require 12 days of headstart in bonding period.

        ii. In the week of July, I can work on GSoC after returning from my college, and if I find that I'm not getting enough time, then I will

skip not-so-important classes and the semester would've just been started, so it'll take time to settle.
And also I'll be working extensively on the weekends(college holiday).

c. Do you have plans to have any other job or internship during this period?

   No.

d. Do you have any other short term commitments during this period?

   No.

2. Have you applied to any other organizations? If so, to whom and do you have a preferred project/org?

   No.

Extra information

OpenHub Account	https://www.openhub.net/accounts/meetmangukiya
University Name	K.J.Somaiya College of Engineering
Major	I.T. Engineering
Current Year & expected graduation year	First Year Engineering. 2020.
Degree	B.Tech.
Alternate Contact	meetmangukiya@coala.io
Homepage	meetmangukiya.github.io
Twitter	@meetmangukiya98
Instant Messaging	Telegram - @meetmangukiya Messenger - @meetbmangukiya