

# Code Inspection

## Link to download the video:

Since the filesize limit on github is 100mb, we used mega:

<https://mega.nz/#!S9gg2DrD!1TAHmXPgn5p1oofh0z-EIKbZWlvDwpOXjgyCt1ml7uw>

Click "Download through your browser"

## Code Review Strategies

### **Our strategy:**

Code modules are inspected by team members individually, then the team members have a meeting to compile the overall list of flaws in the code and discuss the strategies to fix them. We used the checklist provided on the course website. Every team member had a few minutes to share his findings uninterrupted, as per the Speed Review Method, as we say it most fitting for our agile development process.

In our code review, the biggest problems we look for are bugs, coding style and vulnerability. In order to ensure a good deliverable, we want to reduce the number of ways in which bugs can occur in our code. Improving coding style and fixing existing bugs will assist greatly in regards to reducing possible errors in the program. If there are existing tests for the specific classes, they are ran to ensure no errors occur; and if it does not yet have a testing component, the author of the class is then asked to create a test class for it in the coming sprint(s).

Examples of things we looked for included flawed logic in code, amount of legacy code, code reuse and documentation. We often do a brief trace of a the code to ensure the logic of the author is correct in all cases and works as intended. While we do not look out for style such as PEP8 or google's programming style, we do consider things such as encapsulation, better ways to complete a task in a cleaner and more efficient manner as well as common programming paradigms and practices.

# Who reviewed what

Driver.py - Jerry - pg. 4

Comparator.py - Vasili - pg. 11

Proposed\_change.py - Tony - pg. 8

CSV\_data\_parser.py - Eric - pg. 6

XML\_data\_parser.py - Jerry - pg. 1

Planetary\_object.py - Albion - pg. 12

Planet.py - Albion - pg. 12

Star.py - Albion - pg. 11

System.py - Albion - pg. 13

apiGet.py - Eric - pg. 7

gitClone.py - Albion - pg. 14

CSVTest.py - Tony - pg. 8

ProposedChangeTest.py - Tony - pg. 8

TestComparator.py - Tony - pg. 8

XML\_data\_parser\_test.py - Vasili - pg. 11

storage\_manager, storage\_manager\_test - pg. 15

Unit tests are in the test folders

# XML\_data\_parser.py

**Reviewer Name:** Jerry Cheng

**Code under review:** XML\_data\_parser.py

**Date of Review:** November 9, 2016

**Author of code:** Eric

## Bugs:

- **FileNotFoundError:** [Errno 2] No such file or directory: '../storage/OEC\_XML.gz'  
To reproduce: change location of OEC\_XML, run downloadXML()  
Suggestions to fix: Have exception handling for FileNotFoundError when the path does not lead to a file.
- **TypeError:** 'tuple' object is not callable  
To reproduce: run buildSystemFromXML("path")  
Suggestions to fix: Issue seems to be caused when path is set to a non XML file.  
Would suggest exception handling to check if the file from path

## Poor Code Logic:

- hard coded value for Url variable. Would be better if it can be modified to be more modular: line 10.
- The function buildSystemFromXML returns both a list of all planets, stars and systems as well as a dictionary of the same planets, stars and systems. It's unnecessary to store the same objects in two separate data structures. Would suggest to remove one of them.
- Function buildSystemFromXML seems to parse all the systems, stars and planets at once. It's very complex and not easily modular. Would suggest breaking it down into three separate functions, with one parsing planet, star and systems as separate objects.
- Variables StarData and systemData are created but not used. Line 171, 190.

## Poor Coding Style:

- Loop statements in buildSystemFromXML have variable names i, ii, and iii. Variable names undescriptive to its purpose. Lines 66, 98, 136
- Each loop in buildSystemFromXML loops through a child variable (lines 67, 100, 138). It's confusing what each reference means.

## Missing Documentation:

**Unreadable Code:**

- Commented out code in line 11 – 13

**Vulnerabilities in Code:**

- No exception handling in readXML or downloadXML.
- No checking in downloadXML, readXML or buildSystemFromXML for invalid path inputs.

**Poor Testing:**

- Refer to XML\_data\_parser\_test.py code review

# driver.py

**Reviewer Name:** Jerry Cheng

**Code under review:** driver.py

**Date of Review:** November 9, 2016

**Author of code:** Tony, Vasili

## Bugs:

- Code crashes when a non-number value is given for shownumber with ValueError  
To reproduce: "python3 driver.py --shownumber a"  
Suggested fix: check if the input is a non negative int first
- Code crashes when a non-number value is given for accept with ValueError  
To reproduce: "python3 driver.py --accept a"  
Suggested: check if the input is a non negative int first

## Poor Code Logic:

- OEC\_systems and OEC\_planets unused in update() : line 122, 124
- variables output and planet in main() at initiated but unused: line 233, 237
- exception (TimeoutError, API.CannotRetrieveDataException) set as e but not used : line 136, 143
- Hard coded values for NASA\_link and exoplanetEU\_link. Could be problematic if the links to the online databases change. Lines 20-23
- Hard coded values for the paths to Nasa csv and Exoplanet csv and OEC xml. Could be made modular for more flexibility. Lines 24-28
- show\_number and accept calls update() each time per use. This feels redundant and very time intensive. Would be much better to call update(), store the changes identified in update, then use those changes in show\_number/accept.
- Use of `__class__.__name__` on line 158, when the syntax strongly suggests the value should not be referenced outside its class.
- data\_retrieval.remoteGet imported but not used: line 4

## Poor Coding Style:

- Inconsistent coding casing. Variable names shortOPT, longOPT, shortARG, longARG in main are camelcase, but all function names in driver as well as all other variable names in main are snake case. Lines 185-193

#### **Missing Documentation:**

- Docstrings insufficient for usage(), no description for functionality of usage(), no description for the specific requirements and type of input: line 37
- Docstrings insufficient for print\_help(), no description for functionality of print\_help(), no description for the specific requirements and type of input: line 37
- No Docstrings for accept(n) : line 90
- No Docstrings for accept\_all() : line 103
- Docstrings insufficient for update(), no description for functionality of update(), no description for the specific requirements and type of input: line 113
- Docstrings insufficient for main(), no description for the specific requirements and type of input for main(): line 113

#### **Unreadable Code:**

- Commented out code in main() : line 289 – 294

#### **Vulnerabilities in Code:**

- invalid parameters in show\_numbers and accept lead to crashes.
- Update() is called to change the global variable CHANGES. Not sure if this is a good idea. Would suggest creating a return statement in Update and having CHANGES = update() to modify it.
- Accept(n) and accept\_all() can be called to modify the user's XML data without the user being aware of what has changed. Should limit accept so it can only be called once the field being changed has been shown. Suggested approach: show\_number(n) -> accept(); show\_all() -> accept\_all()

#### **Poor Testing:**

- No integration testing of Driver with the various modules imported to driver

# CSV\_data\_parser.py

**Reviewer Name:** Eric Papagiannis

**Code under review:** CSV\_data\_parser.py

**Date of Review:** November 10, 2016

**Author of code:** Albion

## **Bugs:**

- Missing some fields for scraping data in eu and nasa dicts on line 5 and 7
- First name field class methods should be self. Offender: line 174: def convertToOpen(field, data, source):

## **Poor Code Logic:**

- Local variable “re” is not used in \_fixval()
- Parameter “source” is not used in buildDictStar(planets, source)
- Unnecessarily complex way to convert specific databases date fields, try doing it without a function within a function, and just locate the logic into the if statements just below

## **Poor Coding Style:**

- Quality of documentation extremely poor (i. e. line 205: call dat function doe, 79, 83, 92, 176, 199, 202, 205), and includes too much “personality”
- Commented out old-lines of code (i. e. line 71, 131, 133)
- Function buildListStarAllField is redundant
- Local variables shadow names of variables in outer global scope

## **Missing Documentation:**

- Any specific requirements related to parameters in methods not present

## **Unreadable Code:**

- Numerous PEP8 style violations
- UnitConverter took more effort to understand than the rest of the file

## **Vulnerabilities in Code:**

- Only vulnerabilities is anything related to opening files through Python in buildDictionaryPlanets, but it was closed so no memory leaks

## **Poor Testing:**

- Sufficient testing done, but could be a bit more exhaustive to ensure operational excellence

# apiGet.py

**Reviewer Name:** Eric Papagiannis

**Code under review:** apiGet.py

**Date of Review:** November 10, 2016

**Author of code:** Tony

## **Bugs:**

- Line 27, outFile.close has no effect, it should be outFile.close(), and as a result, could lead to potential memory leaks

## **Poor Code Logic:**

- Fairly straightforward implementation using requests library

## **Poor Coding Style:**

- Multiple try catches could lead the flow control being difficult to follow, but it is so basic that

## **Missing Documentation:**

- No internal comments, but code is verbose and self-explanatory
- Unsure of what the variable parameters is, I can guess, but would rather have it described
- No RAISE: statements for exceptions on docstrings

## **Unreadable Code:**

- Numerous PEP8 style violations

## **Vulnerabilities in Code:**

- As stated in bugs, potential memory leak due to lack of closing outFile in getFromApi(self, parameters)

## **Poor Testing:**

- Mainly non-automated testing, but since the whole file is essentially using third party library requests and retrieval of data from online, and nothing in the future will essentially change in this file, it is sufficient to say that these non-automated tests are good enough to ensure correct operation



# Proposed\_change.py, CSVTest.py, ProposeChangeTest.py, TestComparator.py

**Reviewer:** Tony

**Code under review:**

Proposed\_change.py, CSVTest.py, ProposeChangeTest.py, TestComparator.py

**Date:** Nov 10, 2016

**Author of code:**

Vasili, Eric, Jerry, Albion

## Bugs

### **test\_comparator.py:**

- import of data\_comparasion.Comparator fails, from line 8 (may be platform specific), reproduce by running test\_comparator.py (python 3.5, \*nix), fix by moving to root directory
- in class TestComparator, testSQLjoin test resulting in false positive FAIL, from line 41, reproduce by running test\_comparator.py (python 3.5, \*nix), fix by using sets instead of list

### **ProposeChangeTest.py:**

- import of data\_comparasion.proposed\_change fails, from line 8 (may be platform specific), reproduce by running test\_comparator.py (python 3.5, \*nix), fix by moving to root directory

### **ProposeChangeTest.py:**

- test\_sort2 test case, unexpected output before "ok", from line 170, reproduce by running ProposedChangeTest.py (python 3.5, \*nix), fix by removing print statement

## Poor code logic

### **test\_comparator.py:**

- missing methods for important test cases (1.0 == 1, etc.)

### **CSVTest.py:**

- missing any methods for negative test cases

### **proposed\_change.py:**

- merge sort implementation, line 164 to 166, can be simplified to a single line, no need to create separate variables

## Poor code style

### **proposed\_change.py:**

- in class Addition and Modification, toString methods, string building over unnecessarily many statements, line 71 and line 22
- in class Addition and Modification strings are hard coded into comparison statements, line 81 and line 116

### **CSVTest.py:**

- in class TestCSV\_parser, strings are hard coded into multiple methods

### **ProposeChangeTest.py:**

- in all classes, strings are hard coded into multiple methods

### **test\_comparator.py:**

- in class TestComparator, strings are hard coded into multiple methods
- in class TestComparator, line 70, unneeded semicolon

## Missing Documentation

### **test\_comparator.py:**

- in class TestComparator, missing docstrings of methods
- in class TestComparator, missing inline comments

### **CSVTest.py:**

- in class TestCSV\_parser, missing docstring of methods
- in class TestCSV\_parser, missing inline comments
- in class TestCSV\_parser, missing author

### **proposed\_change.py:**

- missing author
- in class Modification, incomplete docstrings of methods
- in class Addition, incomplete class documentation

### **ProposeChangeTest.py:**

- missing author
- in all classes, incomplete docstrings of methods
- in all classes, incomplete class documentation

## Unreadable code

### **ProposeChangeTest.py**

- assorted whitespace left throughout file, line 5 for example

### **Vulnerabilities in code**

#### **ProposeChangeTest.py:**

- no exception handling in code at all

**Poor testing:** N/A since test code

# XML\_data\_parser\_test.py, Comparator.py

**Reviewer:** Vasili Skurydzin

**Code under review:** XML\_data\_parser\_test.py, Comparator.py

**Date:** Nov 11, 2016

**Author of the code:** Eric, Tony

## **Bugs:**

- Comparator.py - crash due to import (builtins.ImportError: No module named 'data\_parsing')

## **Poor code logic:**

- Comparator.py - line 134 : "result\_dict = []" - what?

## **Poor coding style:**

- Comparator.py - lines 150 : commented out code block
- Comparator.py - lines 164 : commented out code block

## **Missing documentation:**

- XML\_data\_parser\_test.py - some tests could use a 1-liner description of what is being tested and what the expected result is, otherwise they are a bit obscure. (specifically: test\_system\_fields, test\_star\_fields)
- Comparator.py - nested comments could be more verbose, given that Comparator is one of the more complex components. (Places - innerJoinDiff, proposedChangeStarCompare)
- Comparator.py - docstrings assume the reader has knowledge of SQL

## **Unreadable code:**

- Comparator.py - method names are cryptic (sqlJoin, sqlJoinNewOnly, innerJoinDiff)

## **Vulnerabilities in code:**

- Comparator.py - \_\_init\_\_ method has a contract condition "origin must be one of {"NASA archive", "exoplanet.eu"}", however this is not checked.

**Poor testing:**

- XML\_data\_parser\_test.py - does not run due to broken import
- XML\_data\_parser\_test.py - crash due to broken `__init__`

# PlanetaryObject.py, Planet.py, Star.py, System.py

**Reviewer:** Albion Fung

**Code under review:** PlanetaryObject.py, Planet.py, Star.py, System.py

**Date:** Nov 11, 2016

**Author of code:** Eric Papagiannis

## **Bugs:**

- None found at the moment

## **Poor code logic:**

- in addVal and addValList, there is logic for catching and adding value properly for values that are not String; however, due to how the parser works it's exclusively a string, so it's unnecessary.

## **Poor coding style:**

- In \_\_str\_\_, there are code that can be easily combined into 1 or 2 lines (eg 18-24) but is instead spread over multiple lines which reduced readability
- Values and datas do not have proper setters and getters, resulting in forcing other classes to directly access the object's variables in order to add or access values in Planet, Star, and System
- Some repeated code found in addVal and addValList

## **Missing Documentation:**

- Basically no documentation
- Doc strings are missing
- Overly detailed comments in the subclasses (planet, star, and system)

## **Unreadable code:**

- Hard to understand some of \_\_str\_\_ due to a lot of things being split up into multiple lines while also having a lot of accessing in one variable (eg self.data[key].\_\_class\_\_.\_\_name\_\_). Even though it's understandable that it might be necessary, a comment on what it is would really help
- Rest of code is perfectly readable

## **Vulnerability in code:**

- Since Planet, Star, and System don't have proper getter and setters, it is very easy for other code to create type mismatch or modify information incorrectly, leading to unwanted differences during --showall.

## **Poor testing:**

- Testing does not exist for any of the above classes

# gitClone.py

**Reviewer:** Albion Fung

**Code under review:** GitClone.py

**Date:** Nov 11, 2016

**Author of code:** Eric Papagiannis

## **Bugs:**

- None found

## **Poor code logic:**

- In modifyXML, there is a lot of repeated code. Code can be reduced by approximately  $\frac{2}{3}$  to  $\frac{3}{4}$ . Create functions for repeated code or put them at the end of the if/else statements if they're common between the two.
- Instead of git push with an upstream specified each time, just set the upstream on the first push and it won't need to be specified again

## **Poor coding style:**

- While not exactly poor style, a dictionary of functions may be beneficial
- Tons of repeated code and unused functions (eg initGit vs initGit2)
- Weird naming of functions (finalizeGit2 but no finalizeGit)
- A lot of commented out code

## **Missing documentation:**

- Missing a lot of docstring
- Insufficient explanation on code logic at times
- A lot of commented out code causing some confusion on first sight
- Instructions and requirements for the document to work (eg, hub) should not be included in this document in the form of comments. Either in the docstring of the class or in the readme file of the repo

## **Unreadable code:**

- Code is readable

## **Vulnerability in code:**

- Some operations may require sudo, which may become an obstacle in the future if user does not have elevated privilege when running the program

## **Poor testing:**

- Testing does not exist for this class, however it may not be possible due to network dependencies

# storage\_manager.py, storage\_manager\_test.py

**Reviewer:** Jerry

**Code under review:** storage\_manager, storage\_manager\_test

**Author of the code under review:** Vasili

**Date:** Nov 13, 2016

## **Bugs:**

- N/A

## **Poor code logic:**

Global variables are declared in storage\_manager and are inaccessible in other components.

## **Poor coding style:**

- storage\_manager\_test - test cases are not commented.

## **Missing documentation:**

- storage\_manager - poor docstrings

## **Unreadable code:**

N/A

## **Vulnerabilities in code:**

N/A

## **Poor testing:**

N / A