Detecting Silent JSON Changes In Dynamic Programming Languages

RIT Rochester Institute of Technology

`collect` previous frame details for good data using inspect module, and `match` it against future data to detect errors that would otherwise go unnoticed, or make the program to fail elsewhere.

Student: Gautam Gadipudi [gg7148@rit.edu] • Advisor: Dr. Michael Mior [mmior@cs.rit.edu]

Introduction

Silent JSON errors are not the regular *syntax errors* or *exceptions*, they are unexpected changes to the JSON data that makes the program to silently process data incorrectly, or raise errors elsewhere in the program.

Tracky is a python class with static methods to collect and match previous frame details like *datatype*, *function*, *line #*, *module* and *code context*.

`Tracky.track()` is called in methods that may have these silent JSON errors, to collect previous frame details or match against target frames (frame metadata from good data).

Background

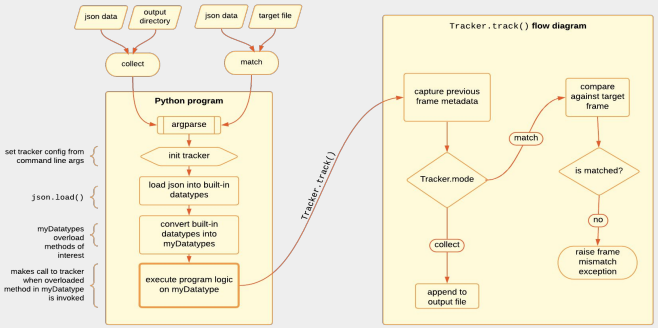
Valid JSON data that is fed into a program can be comprised of datatypes like *object*, *array*, *string*, *number*, *true*, *false* and *null*.

The native approach to work with JSON data in Python is to use the **`json`** module. **`json.load()`** uses the following mapping to decode json data into built-in python datatypes:

JSON datatype	Python3 datatype
Object	`dict`
Array	`list`
String	`str`
Number	`int` or `float`
true	*`True`* (`bool`)
false	*`False`* (`bool`)
null	`None`

Methodology

Control flow:

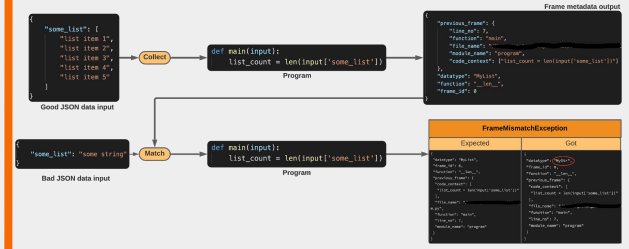


Steps:

- Load JSON data using `json` module.
- Convert JSON data in built-in Python3 datatypes into derived datatypes (prefixed with "my", like `str` -> `myStr`)
- Overload methods of interest in derived datatypes with methods that return the same result as built-in methods, but also capture previous frame details (frame metadata).
- If the mode of the tracker is collect, append the frame metadata to the output file.
- If the mode of the tracker is match, check the captured frame metadata with target frame metadata, and raise a FrameMismatchException if they are not the same.

Use the above steps to discover examples and scenarios that result in silent JSON errors.

Example



Results

Valid Python3 **unary** operations on JSON datatypes:

	Number	String	Array	Boolean	Object
+	✓	X	X	✓	X
-	✓	X	X	✓	X
not	✓	✓	✓	✓	✓
len()	X	✓	✓	X	✓
iter()	X	✓	✓	X	✓
clear()	X	X	✓	X	✓

In the above table, "✓" along a row means that the operation is susceptible to silent JSON error when there is a datatype mismatch. A "X" means that operation is not possible and will raise a runtime exception at the point of evaluation.

Using the above table, we were able to collect 16 scenarios that cause silent JSON errors due to datatype mismatch:

	myDict	myList	myStr
len()	2	2	2
iter()	2	2	2
clear()	2	2	N/A