SharkNinja Coding Task: Number Analyzer

Aditya Kulkarni

The Approach:

The approach taken in the provided code revolves around creating a flexible and extensible number analysis tool, the NumberAnalyzer class, which processes numbers within a specified range and categorizes them based on rules defined in a configuration file. The following steps summarize the approach:

1. **Initialization and Input Validation**: The constructor (__init__) ensures the start and end values are integers, with start being less than or equal to end. It also verifies the existence of the configuration file and its directory.
2. **Configuration Parsing**: The _parse_config method reads a JSON configuration file containing rules for categorizing numbers. Rules can include built-in checks (e.g., prime, even, odd) or custom rules defined as Python functions or lambda expressions. Assumptions made here include:
   - The configuration file is correctly formatted as JSON.
   - Any custom rules provided are syntactically valid Python expressions or functions.
3. **Rule Mapping**: Each rule in the configuration is mapped to a corresponding function. Built-in rules like "prime," "even," and "odd" are handled by predefined methods, while custom rules are dynamically evaluated or executed using eval or exec. This assumes that users provide safe and valid Python code for custom rules.
4. **Number Analysis**: The _get_results method iterates through the specified range of numbers, applying each rule to categorize numbers. Results are stored in two formats:
   - A list (self.results) containing categories for each number as strings.
   - A dictionary (self.results_debug) mapping numbers to their categories for debugging purposes.
5. **Error Handling**: Custom exceptions (NumberAnalyzerException) are raised when invalid inputs, configuration issues, or syntax errors in rules occur.
6. **Output**: The print_results method provides options to display results in a detailed (debug) or simplified format.

This approach emphasizes modularity, extensibility, and error handling while assuming that users provide valid inputs and configurations.

Test analysis:

All test cases for the NumberAnalyzer class passed successfully, confirming the robustness and correctness of its implementation. The tests validated various aspects of the class, including initialization, configuration parsing, rule application, and error handling. Key highlights include:

Configuration-Based Tests

- Valid Initialization: Ensures that valid configuration files are loaded correctly.
- Invalid Configurations:
- Missing configuration directory or file.
- Syntax errors in rules or JSON structure.
- Custom Configurations: Verifies that configurations at custom paths or with full method definitions work as expected.

Parameter Validation Tests

- Ensures proper handling of invalid input parameters such as:
- Start greater than end.
- Non-integer start or end values.

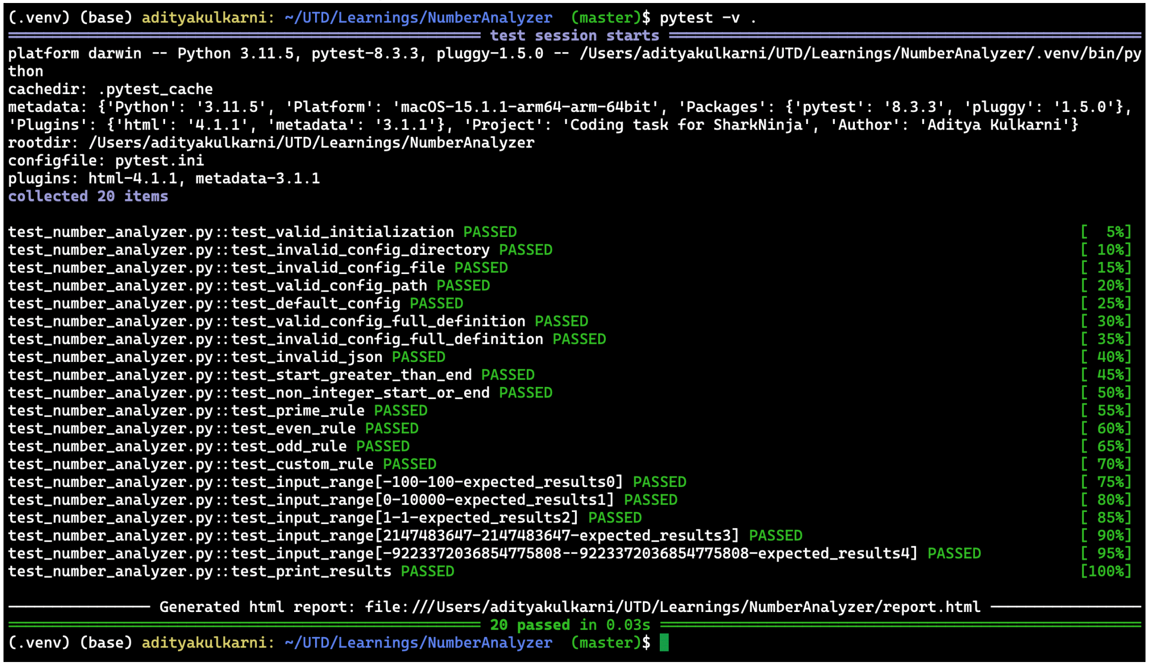
Functional Tests

- Verifies that individual rules (prime, even, odd) work as expected.
- Ensures correct behavior for large ranges, negative ranges, and edge cases like single-number ranges.

Output-Based Tests

- Validates that results are printed in both detailed and simplified formats.

These results indicate that the NumberAnalyzer is well-designed to handle diverse scenarios while maintaining accuracy and reliability.

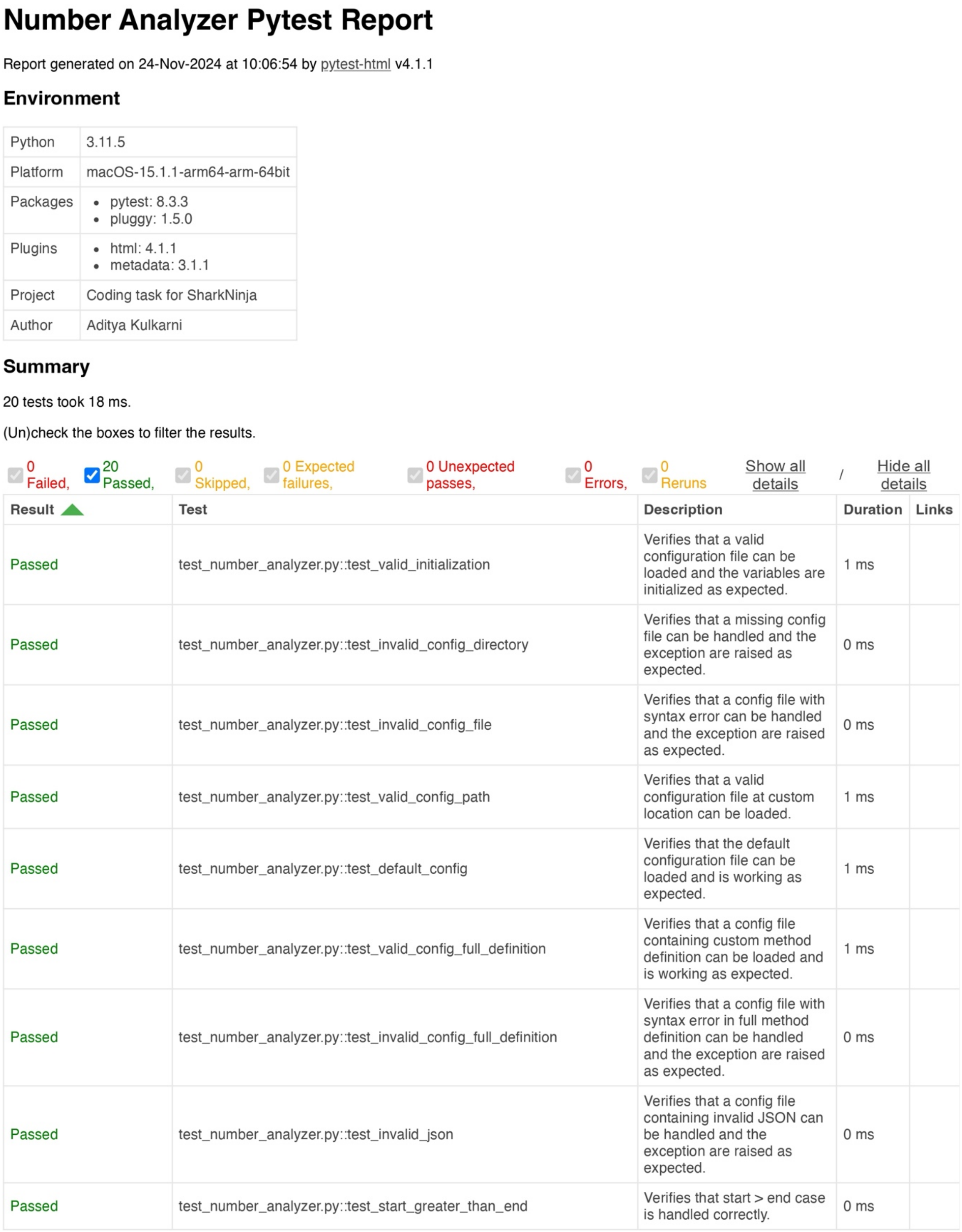


```
(.venv) (base) adityakulkarni: ~/UTD/Learnings/NumberAnalyzer  (master)$ pytest -v .
===================================================== test session starts =====================================================
platform darwin -- Python 3.11.5, pytest-8.3.3, pluggy-1.5.0 -- /Users/adityakulkarni/UTD/Learnings/NumberAnalyzer/.venv/bin/py
thon
cachedir: .pytest_cache
metadata: {'Python': '3.11.5', 'Platform': 'macOS-15.1.1-arm64-arm-64bit', 'Packages': {'pytest': '8.3.3', 'pluggy': '1.5.0'},
'Plugins': {'html': '4.1.1', 'metadata': '3.1.1'}, 'Project': 'Coding task for SharkNinja', 'Author': 'Aditya Kulkarni'}
rootdir: /Users/adityakulkarni/UTD/Learnings/NumberAnalyzer
configfile: pytest.ini
plugins: html-4.1.1, metadata-3.1.1
collected 20 items

test_number_analyzer.py::test_valid_initialization PASSED                                                                 [  5%]
test_number_analyzer.py::test_invalid_config_directory PASSED                                                             [ 10%]
test_number_analyzer.py::test_invalid_config_file PASSED                                                                  [ 15%]
test_number_analyzer.py::test_valid_config_path PASSED                                                                    [ 20%]
test_number_analyzer.py::test_default_config PASSED                                                                       [ 25%]
test_number_analyzer.py::test_valid_config_full_definition PASSED                                                         [ 30%]
test_number_analyzer.py::test_invalid_config_full_definition PASSED                                                       [ 35%]
test_number_analyzer.py::test_invalid_json PASSED                                                                         [ 40%]
test_number_analyzer.py::test_start_greater_than_end PASSED                                                               [ 45%]
test_number_analyzer.py::test_non_integer_start_or_end PASSED                                                             [ 50%]
test_number_analyzer.py::test_prime_rule PASSED                                                                           [ 55%]
test_number_analyzer.py::test_even_rule PASSED                                                                            [ 60%]
test_number_analyzer.py::test_odd_rule PASSED                                                                             [ 65%]
test_number_analyzer.py::test_custom_rule PASSED                                                                          [ 70%]
test_number_analyzer.py::test_input_range[-100-100-expected_results0] PASSED                                              [ 75%]
test_number_analyzer.py::test_input_range[0-10000-expected_results1] PASSED                                               [ 80%]
test_number_analyzer.py::test_input_range[1-1-expected_results2] PASSED                                                   [ 85%]
test_number_analyzer.py::test_input_range[2147483647-2147483647-expected_results3] PASSED                                 [ 90%]
test_number_analyzer.py::test_input_range[-9223372036854775808--9223372036854775808-expected_results4] PASSED             [ 95%]
test_number_analyzer.py::test_print_results PASSED                                                                        [100%]

---------------- Generated html report: file:///Users/adityakulkarni/UTD/Learnings/NumberAnalyzer/report.html -----------------
===================================================== 20 passed in 0.03s ======================================================
(.venv) (base) adityakulkarni: ~/UTD/Learnings/NumberAnalyzer  (master)$
```

The report for all testcases can be found in: NumberAnalyzer /report.html



Number Analyzer Pytest Report

Report generated on 24-Nov-2024 at 10:06:54 by pytest-html v4.1.1

Environment

Python	3.11.5
Platform	macOS-15.1.1-arm64-arm-64bit
Packages	• pytest: 8.3.3 • pluggy: 1.5.0
Plugins	• html: 4.1.1 • metadata: 3.1.1
Project	Coding task for SharkNinja
Author	Aditya Kulkarni

Summary

20 tests took 18 ms.

(Un)check the boxes to filter the results.

☑ 0 Failed, ☑ 20 Passed, ☑ 0 Skipped, ☑ 0 Expected failures, ☑ 0 Unexpected passes, ☑ 0 Errors, ☑ 0 Reruns Show all details / Hide all details

Result ▲	Test	Description	Duration	Links
Passed	test_number_analyzer.py::test_valid_initialization	Verifies that a valid configuration file can be loaded and the variables are initialized as expected.	1 ms	
Passed	test_number_analyzer.py::test_invalid_config_directory	Verifies that a missing config file can be handled and the exception are raised as expected.	0 ms	
Passed	test_number_analyzer.py::test_invalid_config_file	Verifies that a config file with syntax error can be handled and the exception are raised as expected.	0 ms	
Passed	test_number_analyzer.py::test_valid_config_path	Verifies that a valid configuration file at custom location can be loaded.	1 ms	
Passed	test_number_analyzer.py::test_default_config	Verifies that the default configuration file can be loaded and is working as expected.	1 ms	
Passed	test_number_analyzer.py::test_valid_config_full_definition	Verifies that a config file containing custom method definition can be loaded and is working as expected.	1 ms	
Passed	test_number_analyzer.py::test_invalid_config_full_definition	Verifies that a config file with syntax error in full method definition can be handled and the exception are raised as expected.	0 ms	
Passed	test_number_analyzer.py::test_invalid_json	Verifies that a config file containing invalid JSON can be handled and the exception are raised as expected.	0 ms	
Passed	test_number_analyzer.py::test_start_greater_than_end	Verifies that start > end case is handled correctly.	0 ms	

Result ▲	Test	Description	Duration	Links
Passed	test_number_analyzer.py::test_non_integer_start_or_end	Verifies that non-integer start or end case is handled correctly.	0 ms	
Passed	test_number_analyzer.py::test_prime_rule	Verifies that the prime rule works as expected.	0 ms	
Passed	test_number_analyzer.py::test_even_rule	Verifies that the even rule works as expected.	0 ms	
Passed	test_number_analyzer.py::test_odd_rule	Verifies that the odd rule works as expected.	0 ms	
Passed	test_number_analyzer.py::test_custom_rule	Verifies that the divisible by 5 custom rule works as expected.	0 ms	
Passed	test_number_analyzer.py::test_input_range[-100-100-expected_results0]	Verifies the function works as expected for various input ranges : start = -100, end = 100.	0 ms	
Passed	test_number_analyzer.py::test_input_range[0-10000-expected_results1]	Verifies the function works as expected for various input ranges : start = 0, end = 10000.	9 ms	
Passed	test_number_analyzer.py::test_input_range[1-1-expected_results2]	Verifies the function works as expected for various input ranges : start = 1, end = 1.	0 ms	
Passed	test_number_analyzer.py::test_input_range[2147483647-2147483647-expected_results3]	Verifies the function works as expected for various input ranges : start = 2147483647, end = 2147483647.	2 ms	
Passed	test_number_analyzer.py::test_input_range[-9223372036854775808--9223372036854775808-expected_results4]	Verifies the function works as expected for various input ranges : start = -9223372036854775808, end = -9223372036854775808.	0 ms	
Passed	test_number_analyzer.py::test_print_results	Verifies that results are printed in expected format.	1 ms	