Lab 5

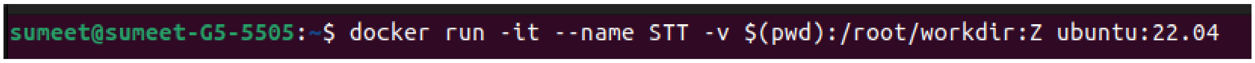
Introduction:

In this lab, we will learn about code coverage analysis and automated test generation. We use code coverage to understand how much of code is tested when we run a test suite. There are different types of coverage, such as **line coverage**, **branch coverage**, and **function coverage**. By measuring these, we can find out which parts of the code are not tested properly.

We will work with a Python repository called **keon/algorithms** and use **pytest**, **coverage**, and **pynguin** to analyze and improve test coverage. First, we will check the coverage of test cases provided by the developer, then generate new test cases using **pynguin** to improve the coverage. This will help us understand the importance of writing effective test cases and how automated testing can assist in this process.

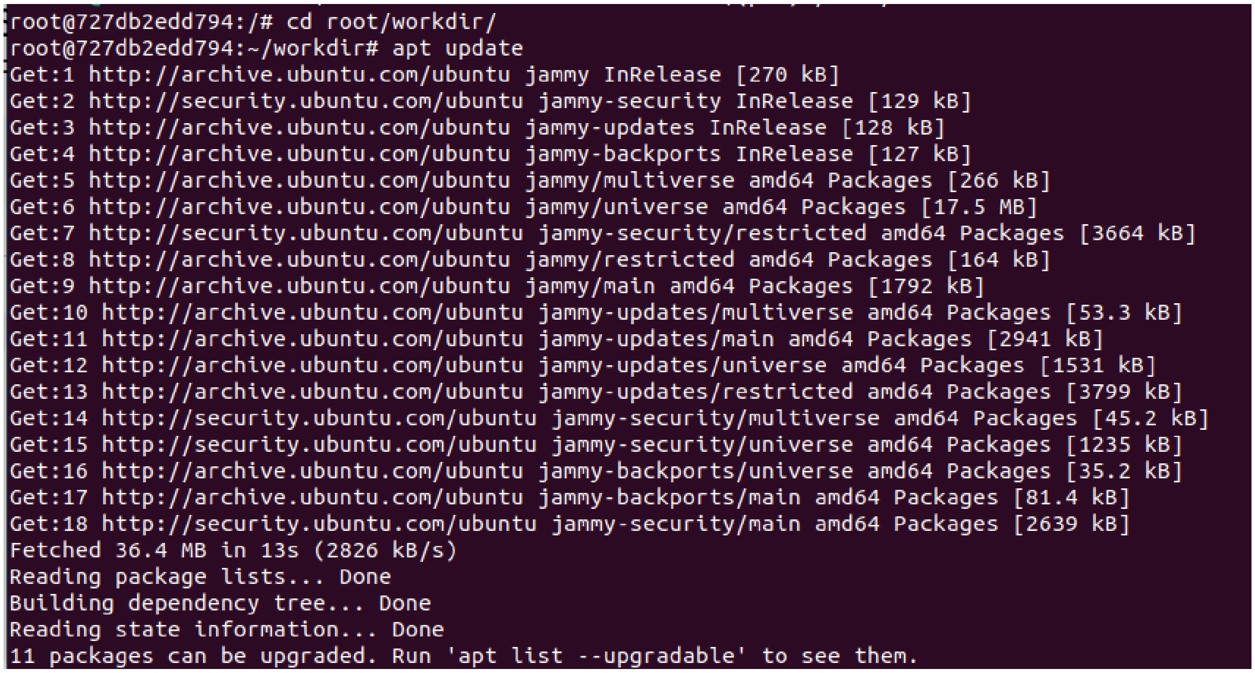
Setup and Tools:

1. **Setting up isolated environment:**
   Using a docker container



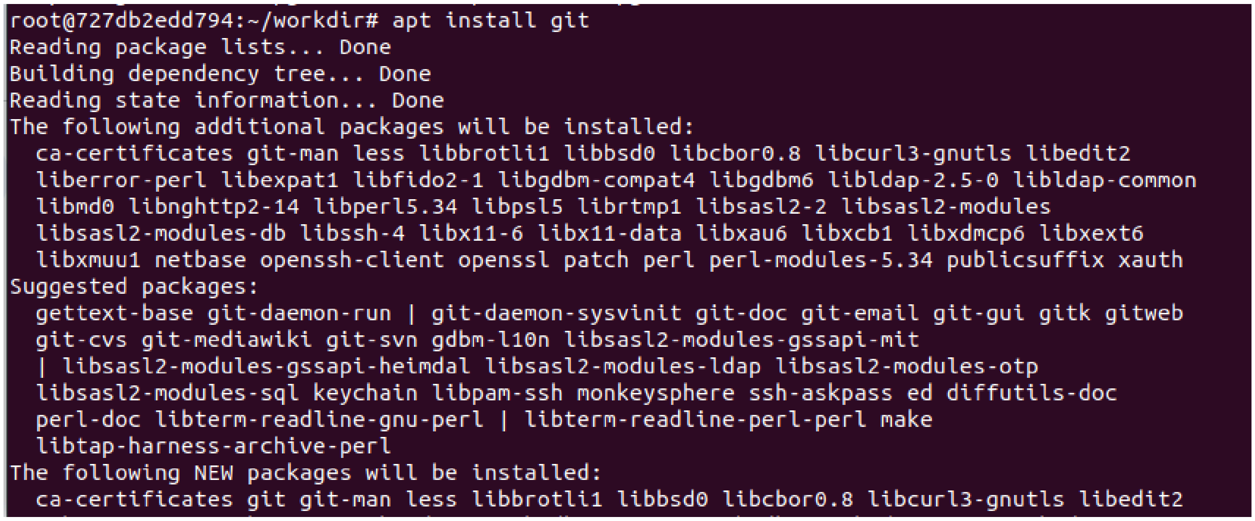
```
sumeet@sumeet-G5-5505:~$ docker run -it --name STT -v $(pwd):/root/workdir:Z ubuntu:22.04
```

2. **Going to working directory and doing apt update**



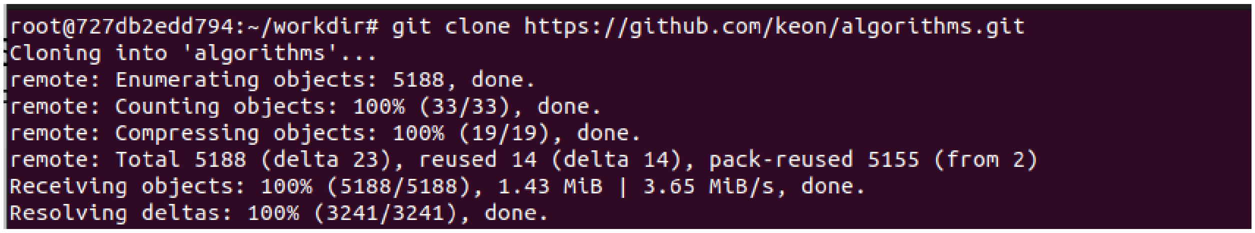
```
root@727db2edd794:/# cd root/workdir/
root@727db2edd794:~/workdir# apt update
Get:1 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [3664 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [53.3 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2941 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1531 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [3799 kB]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [45.2 kB]
Get:15 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [1235 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [35.2 kB]
Get:17 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [81.4 kB]
Get:18 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2639 kB]
Fetched 36.4 MB in 13s (2826 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
11 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

3. **Installing git**



```
root@727db2edd794:~/workdir# apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates git-man less libbrotli1 libbsd0 libcbor0.8 libcurl3-gnutls libedit2
  liberror-perl libexpat1 libfido2-1 libgdbm-compat4 libgdbm6 libldap-2.5-0 libldap-common
  libmd0 libnghttp2-14 libperl5.34 libpsl5 librtmp1 libsasl2-2 libsasl2-modules
  libsasl2-modules-db libssh-4 libx11-6 libx11-data libxau6 libxcb1 libxdmcp6 libxext6
  libxmuu1 netbase openssh-client openssl patch perl perl-modules-5.34 publicsuffix xauth
Suggested packages:
  gettext-base git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb
  git-cvs git-mediawiki git-svn gdbm-l10n libsasl2-modules-gssapi-mit
  | libsasl2-modules-gssapi-heimdal libsasl2-modules-ldap libsasl2-modules-otp
  libsasl2-modules-sql keychain libpam-ssh monkeysphere ssh-askpass ed diffutils-doc
  perl-doc libterm-readline-gnu-perl | libterm-readline-perl-perl make
  libtap-harness-archive-perl
The following NEW packages will be installed:
  ca-certificates git git-man less libbrotli1 libbsd0 libcbor0.8 libcurl3-gnutls libedit2
```

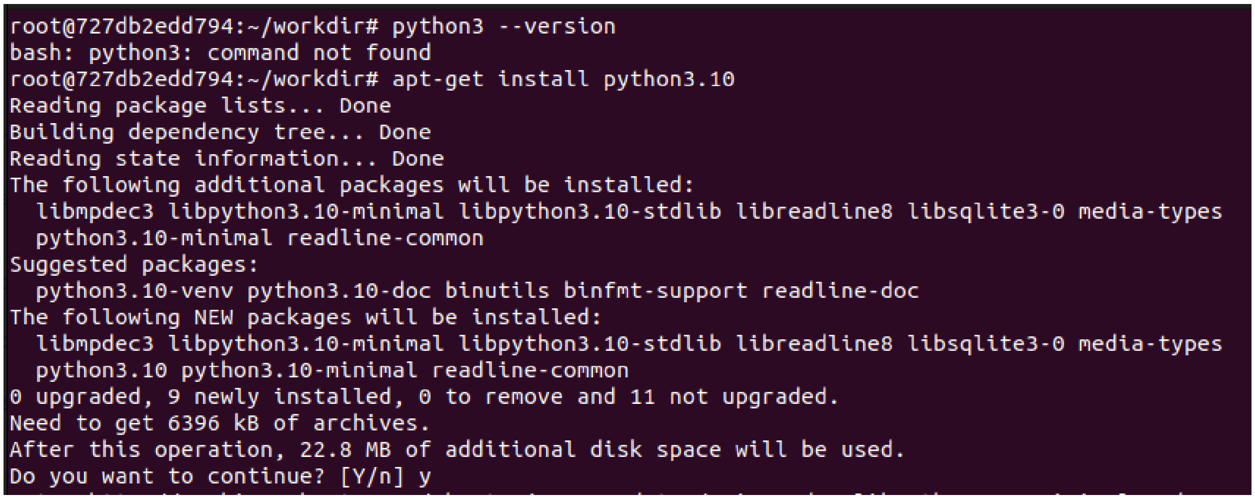
4. Cloning the repository



```
root@727db2edd794:~/workdir# git clone https://github.com/keon/algorithms.git
Cloning into 'algorithms'...
remote: Enumerating objects: 5188, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 5188 (delta 23), reused 14 (delta 14), pack-reused 5155 (from 2)
Receiving objects: 100% (5188/5188), 1.43 MiB | 3.65 MiB/s, done.
Resolving deltas: 100% (3241/3241), done.
```

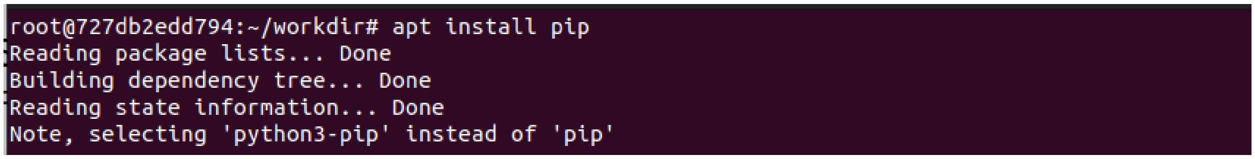
Commit Hash: cad4754bc71742c2d6fcbd3b92ae74834d359844

5. Installing Python



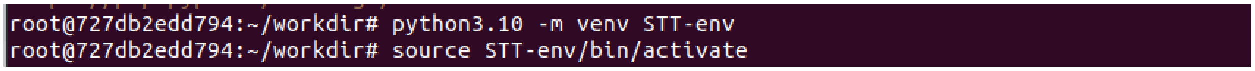
```
root@727db2edd794:~/workdir# python3 --version
bash: python3: command not found
root@727db2edd794:~/workdir# apt-get install python3.10
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libmpdec3 libpython3.10-minimal libpython3.10-stdlib libreadline8 libsqlite3-0 media-types
  python3.10-minimal readline-common
Suggested packages:
  python3.10-venv python3.10-doc binutils binfmt-support readline-doc
The following NEW packages will be installed:
  libmpdec3 libpython3.10-minimal libpython3.10-stdlib libreadline8 libsqlite3-0 media-types
  python3.10 python3.10-minimal readline-common
0 upgraded, 9 newly installed, 0 to remove and 11 not upgraded.
Need to get 6396 kB of archives.
After this operation, 22.8 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

6. Installing pip



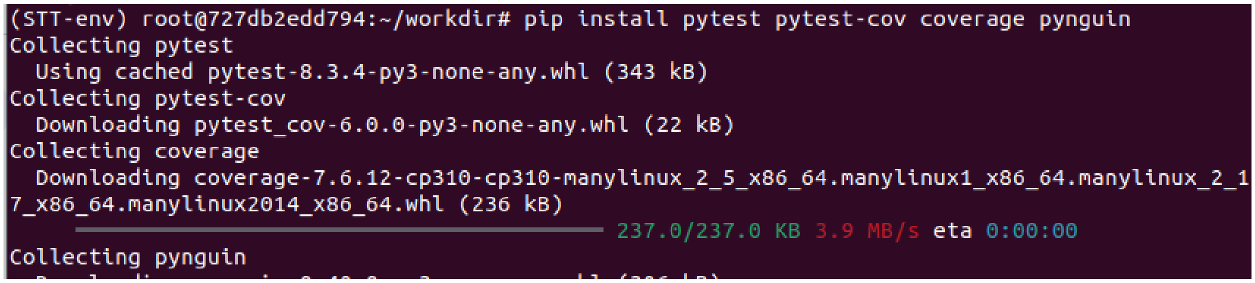
```
root@727db2edd794:~/workdir# apt install pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'python3-pip' instead of 'pip'
```

7. Setting up the virtual environment



```
root@727db2edd794:~/workdir# python3.10 -m venv STT-env
root@727db2edd794:~/workdir# source STT-env/bin/activate
```

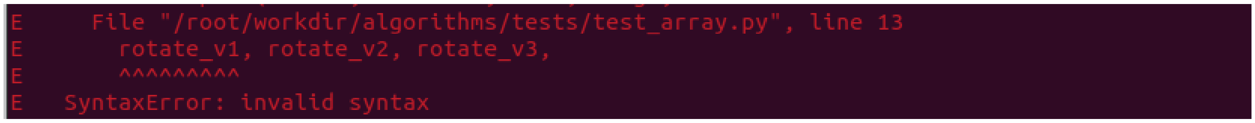
8. Installing pytest, pytest-cov, coverage, pynguin



```
(STT-env) root@727db2edd794:~/workdir# pip install pytest pytest-cov coverage pynguin
Collecting pytest
  Using cached pytest-8.3.4-py3-none-any.whl (343 kB)
Collecting pytest-cov
  Downloading pytest_cov-6.0.0-py3-none-any.whl (22 kB)
Collecting coverage
  Downloading coverage-7.6.12-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_1
7_x86_64.manylinux2014_x86_64.whl (236 kB)
                                    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 237.0/237.0 KB 3.9 MB/s eta 0:00:00
Collecting pynguin
```

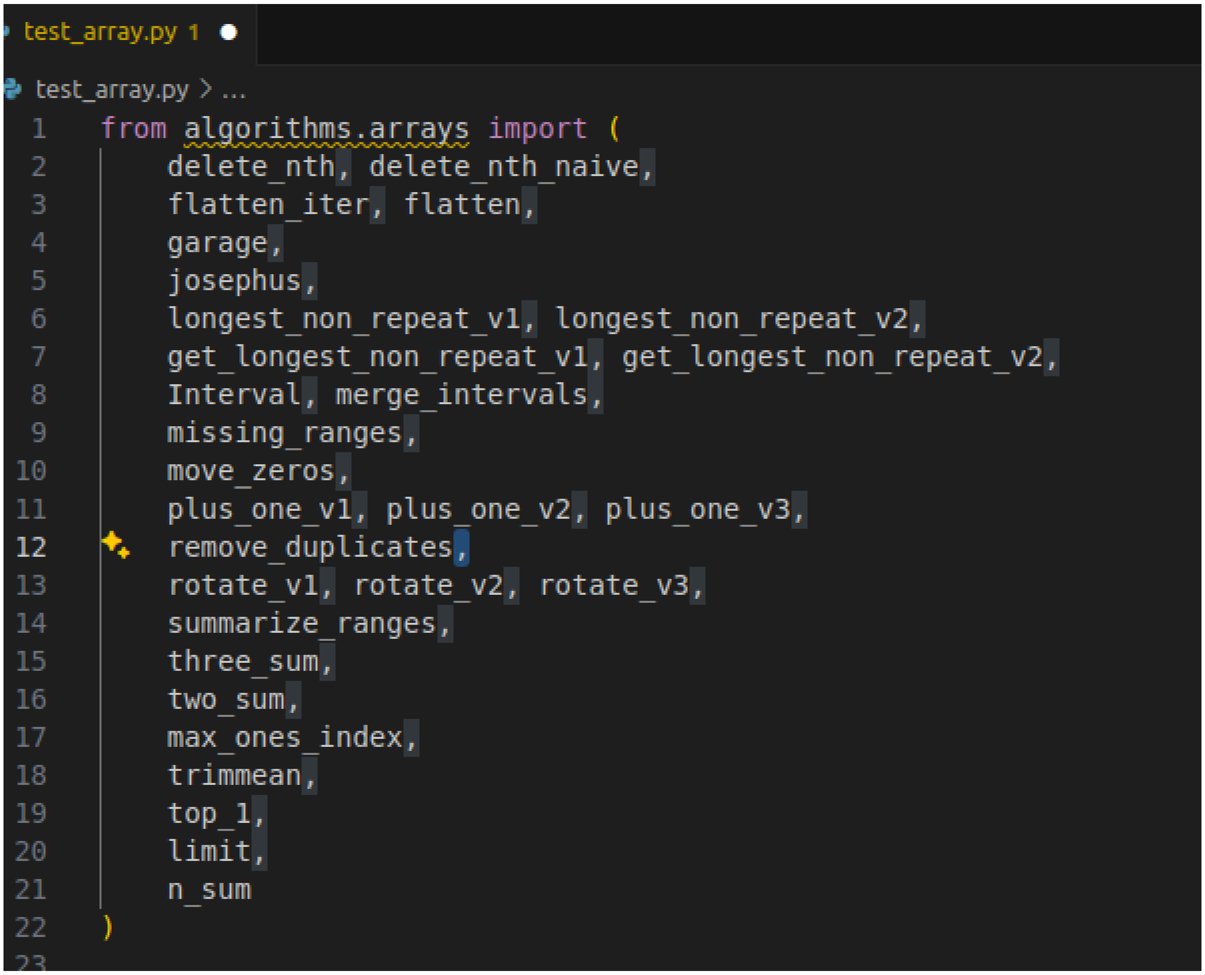
Methodology:

1. **One test file has errors which we need to fix.**



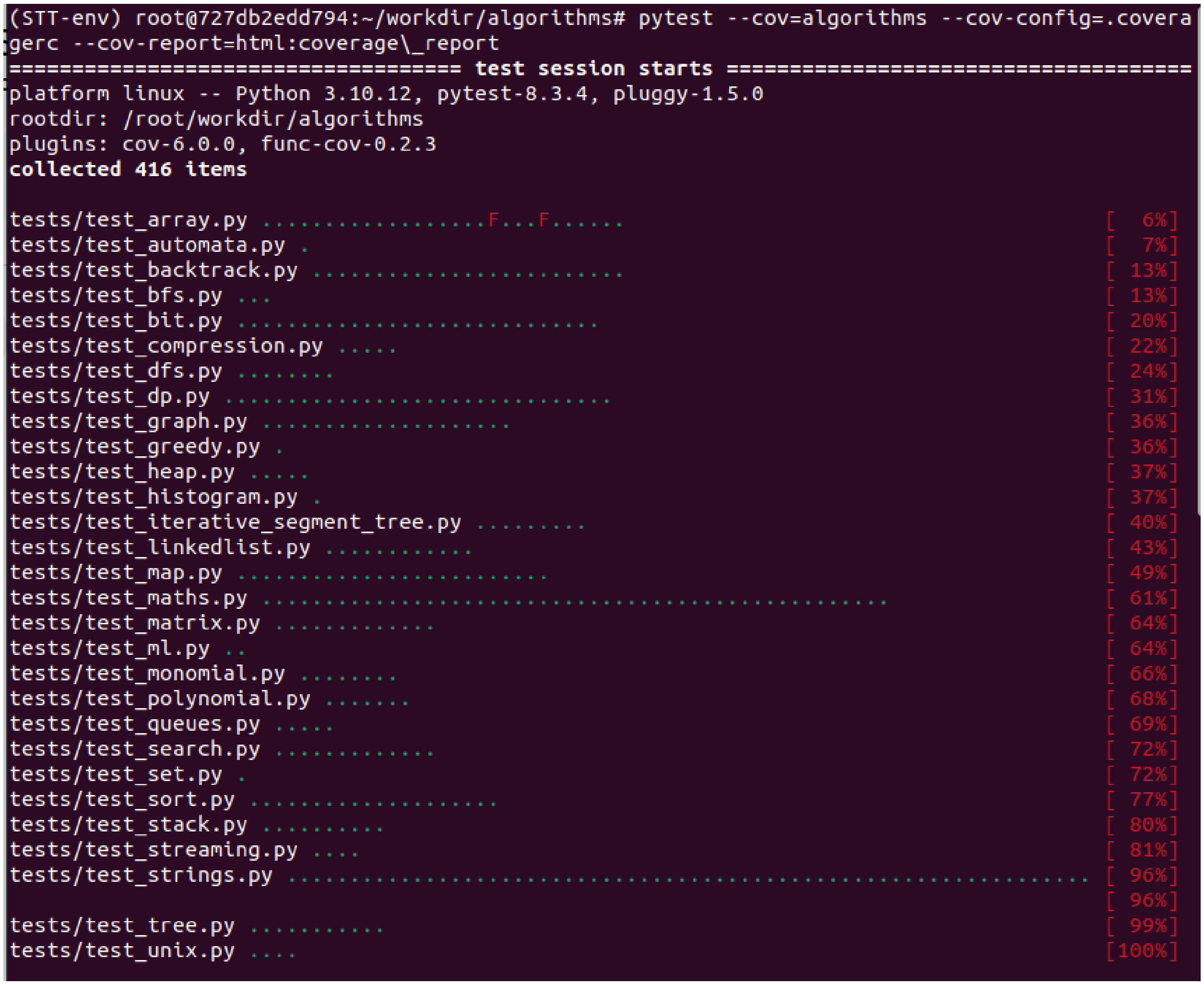
```
E     File "/root/workdir/algorithms/tests/test_array.py", line 13
E       rotate_v1, rotate_v2, rotate_v3,
E       ^^^^^^^^^
E   SyntaxError: invalid syntax
```

One comma was missing on line 12 of the “**test_array.py**” file



```
 1  from algorithms.arrays import (
 2      delete_nth, delete_nth_naive,
 3      flatten_iter, flatten,
 4      garage,
 5      josephus,
 6      longest_non_repeat_v1, longest_non_repeat_v2,
 7      get_longest_non_repeat_v1, get_longest_non_repeat_v2,
 8      Interval, merge_intervals,
 9      missing_ranges,
10      move_zeros,
11      plus_one_v1, plus_one_v2, plus_one_v3,
12      remove_duplicates,
13      rotate_v1, rotate_v2, rotate_v3,
14      summarize_ranges,
15      three_sum,
16      two_sum,
17      max_ones_index,
18      trimmean,
19      top_1,
20      limit,
21      n_sum
22  )
23
```

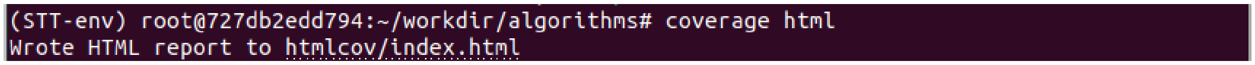
2. Executing existing test cases and recording the coverage metrics.



```
(STT-env) root@727db2edd794:~/workdir/algorithms# pytest --cov=algorithms --cov-config=.covera
gerc --cov-report=html:coverage\_report
=================================== test session starts ======================================
platform linux -- Python 3.10.12, pytest-8.3.4, pluggy-1.5.0
rootdir: /root/workdir/algorithms
plugins: cov-6.0.0, func-cov-0.2.3
collected 416 items

tests/test_array.py ..................F...F......                                      [  6%]
tests/test_automata.py .                                                               [  7%]
tests/test_backtrack.py .........................                                      [ 13%]
tests/test_bfs.py ...                                                                  [ 13%]
tests/test_bit.py .............................                                        [ 20%]
tests/test_compression.py .....                                                        [ 22%]
tests/test_dfs.py ........                                                             [ 24%]
tests/test_dp.py ...............................                                       [ 31%]
tests/test_graph.py ....................                                               [ 36%]
tests/test_greedy.py .                                                                 [ 36%]
tests/test_heap.py .....                                                               [ 37%]
tests/test_histogram.py .                                                              [ 37%]
tests/test_iterative_segment_tree.py .........                                         [ 40%]
tests/test_linkedlist.py ............                                                  [ 43%]
tests/test_map.py .........................                                            [ 49%]
tests/test_maths.py ..................................................                 [ 61%]
tests/test_matrix.py .............                                                     [ 64%]
tests/test_ml.py ..                                                                    [ 64%]
tests/test_monomial.py ........                                                        [ 66%]
tests/test_polynomial.py .......                                                       [ 68%]
tests/test_queues.py .....                                                             [ 69%]
tests/test_search.py .............                                                     [ 72%]
tests/test_set.py .                                                                    [ 72%]
tests/test_sort.py ....................                                                [ 77%]
tests/test_stack.py ..........                                                         [ 80%]
tests/test_streaming.py ....                                                           [ 81%]
tests/test_strings.py ................................................................ [ 96%]
                                                                                       [ 96%]
tests/test_tree.py ...........                                                         [ 99%]
tests/test_unix.py ....                                                                [100%]
```

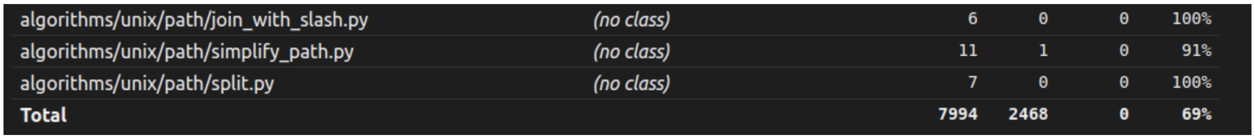
There are 2 failures in the test_array.py file

3. Generating visualizations for the coverage file.



```
(STT-env) root@727db2edd794:~/workdir/algorithms# coverage html
Wrote HTML report to htmlcov/index.html
```

4. Visualizing the index.html file

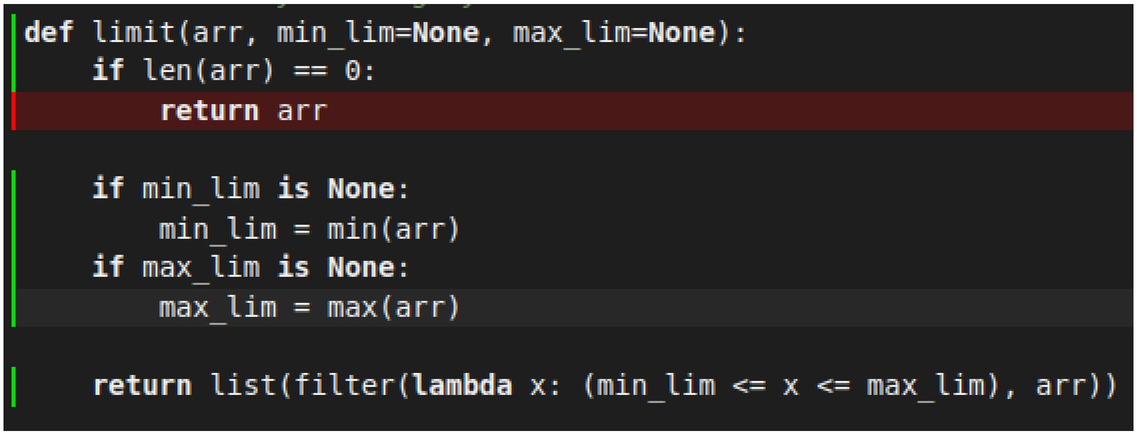


algorithms/unix/path/join_with_slash.py	*(no class)*	6	0	0	100%
algorithms/unix/path/simplify_path.py	*(no class)*	11	1	0	91%
algorithms/unix/path/split.py	*(no class)*	7	0	0	100%
Total		**7994**	**2468**	**0**	**69%**

Total coverage across all files is 69%

5. Analyzing 3 files with incomplete coverage

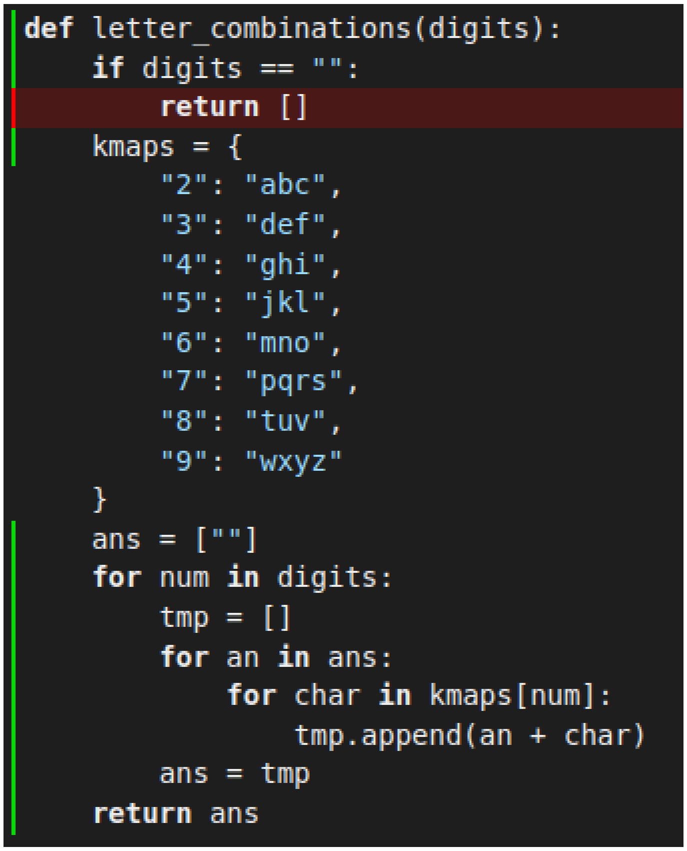
a. arrays/limit.py - **coverage = 88%**



```
def limit(arr, min_lim=None, max_lim=None):
    if len(arr) == 0:
        return arr

    if min_lim is None:
        min_lim = min(arr)
    if max_lim is None:
        max_lim = max(arr)

    return list(filter(lambda x: (min_lim <= x <= max_lim), arr))
```

- One return is not called.
- All other lines are covered.

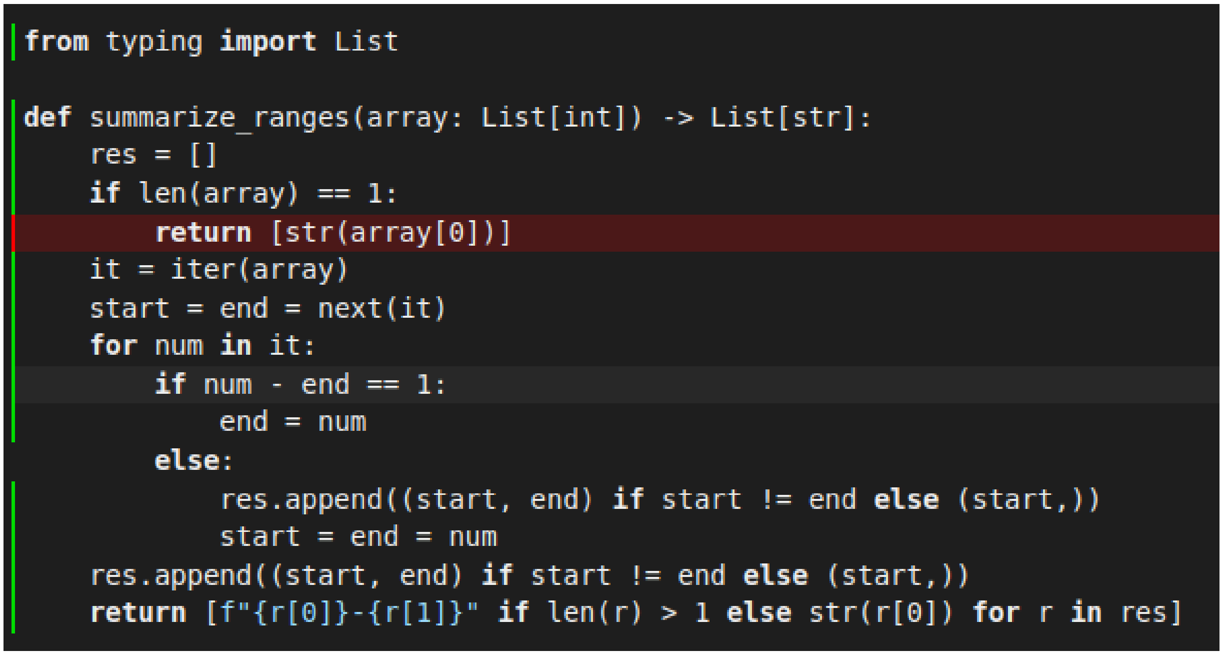
b. backtrack/letter_combination.py - **coverage = 92%**



```
def letter_combinations(digits):
    if digits == "":
        return []
    kmaps = {
        "2": "abc",
        "3": "def",
        "4": "ghi",
        "5": "jkl",
        "6": "mno",
        "7": "pqrs",
        "8": "tuv",
        "9": "wxyz"
    }
    ans = [""]
    for num in digits:
        tmp = []
        for an in ans:
            for char in kmaps[num]:
                tmp.append(an + char)
        ans = tmp
    return ans
```

- One return is not called.
- All other lines are covered.

c. arrays/summarize_ranges.py - **coverage = 93%**

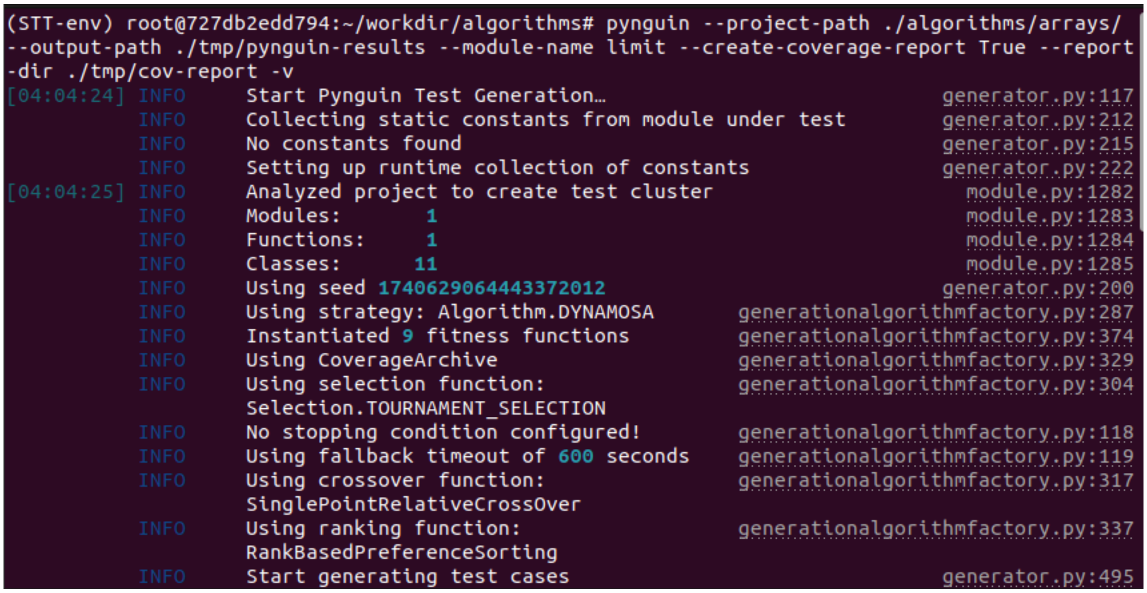


```
from typing import List

def summarize_ranges(array: List[int]) -> List[str]:
    res = []
    if len(array) == 1:
        return [str(array[0])]
    it = iter(array)
    start = end = next(it)
    for num in it:
        if num - end == 1:
            end = num
        else:
            res.append((start, end) if start != end else (start,))
            start = end = num
    res.append((start, end) if start != end else (start,))
    return [f"{r[0]}-{r[1]}" if len(r) > 1 else str(r[0]) for r in res]
```

- One return is not called.
- All other lines are covered.

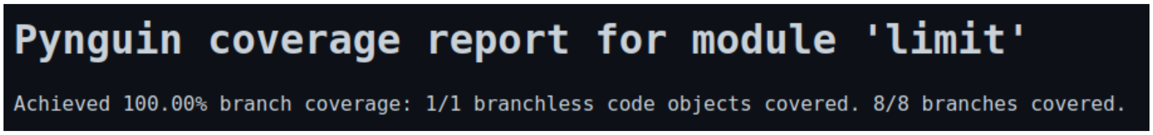
6. Generating test cases and coverage report using pynguin

a. arrays/limit.py



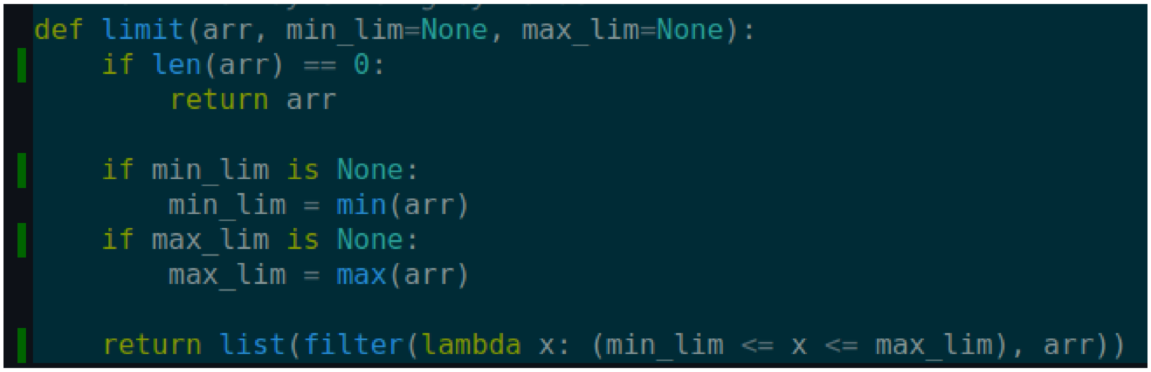
```
(STT-env) root@727db2edd794:~/workdir/algorithms# pynguin --project-path ./algorithms/arrays/
--output-path ./tmp/pynguin-results --module-name limit --create-coverage-report True --report
-dir ./tmp/cov-report -v
[04:04:24] INFO     Start Pynguin Test Generation…                          generator.py:117
           INFO     Collecting static constants from module under test      generator.py:212
           INFO     No constants found                                      generator.py:215
           INFO     Setting up runtime collection of constants              generator.py:222
[04:04:25] INFO     Analyzed project to create test cluster                  module.py:1282
           INFO     Modules:        1                                        module.py:1283
           INFO     Functions:      1                                        module.py:1284
           INFO     Classes:       11                                        module.py:1285
           INFO     Using seed 1740629064443372012                          generator.py:200
           INFO     Using strategy: Algorithm.DYNAMOSA      generationalgorithmfactory.py:287
           INFO     Instantiated 9 fitness functions        generationalgorithmfactory.py:374
           INFO     Using CoverageArchive                   generationalgorithmfactory.py:329
           INFO     Using selection function:               generationalgorithmfactory.py:304
                    Selection.TOURNAMENT_SELECTION
           INFO     No stopping condition configured!       generationalgorithmfactory.py:118
           INFO     Using fallback timeout of 600 seconds   generationalgorithmfactory.py:119
           INFO     Using crossover function:               generationalgorithmfactory.py:317
                    SinglePointRelativeCrossOver
           INFO     Using ranking function:                 generationalgorithmfactory.py:337
                    RankBasedPreferenceSorting
           INFO     Start generating test cases                             generator.py:495
```

Coverage for **pynguin = 100%**



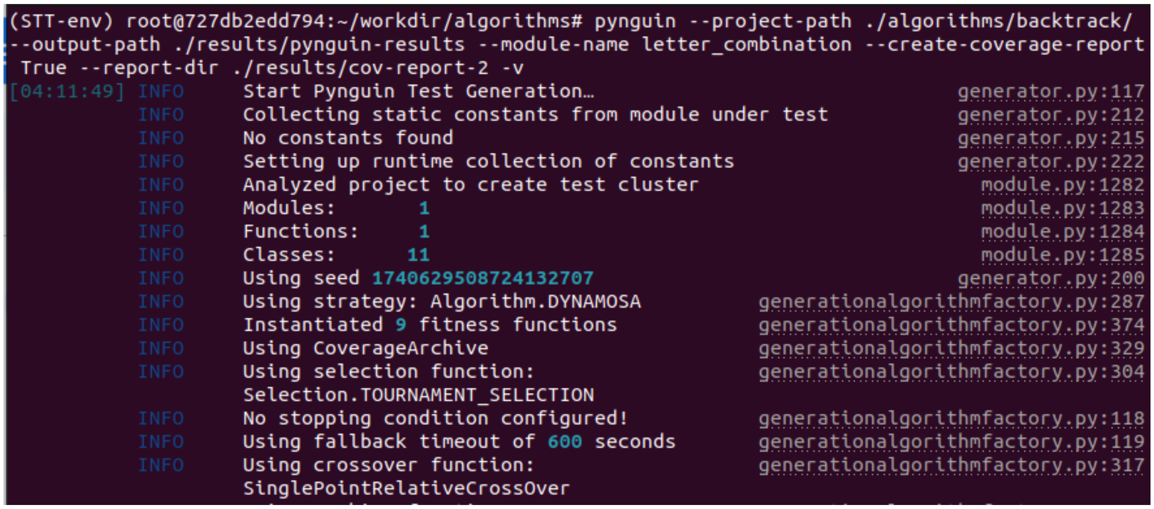
Pynguin coverage report for module 'limit'

Achieved 100.00% branch coverage: 1/1 branchless code objects covered. 8/8 branches covered.



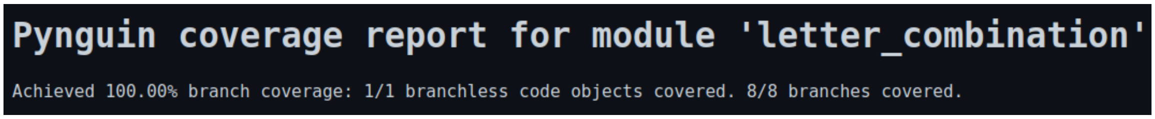
```
def limit(arr, min_lim=None, max_lim=None):
    if len(arr) == 0:
        return arr

    if min_lim is None:
        min_lim = min(arr)
    if max_lim is None:
        max_lim = max(arr)

    return list(filter(lambda x: (min_lim <= x <= max_lim), arr))
```

b. backtrack/letter_combination.py

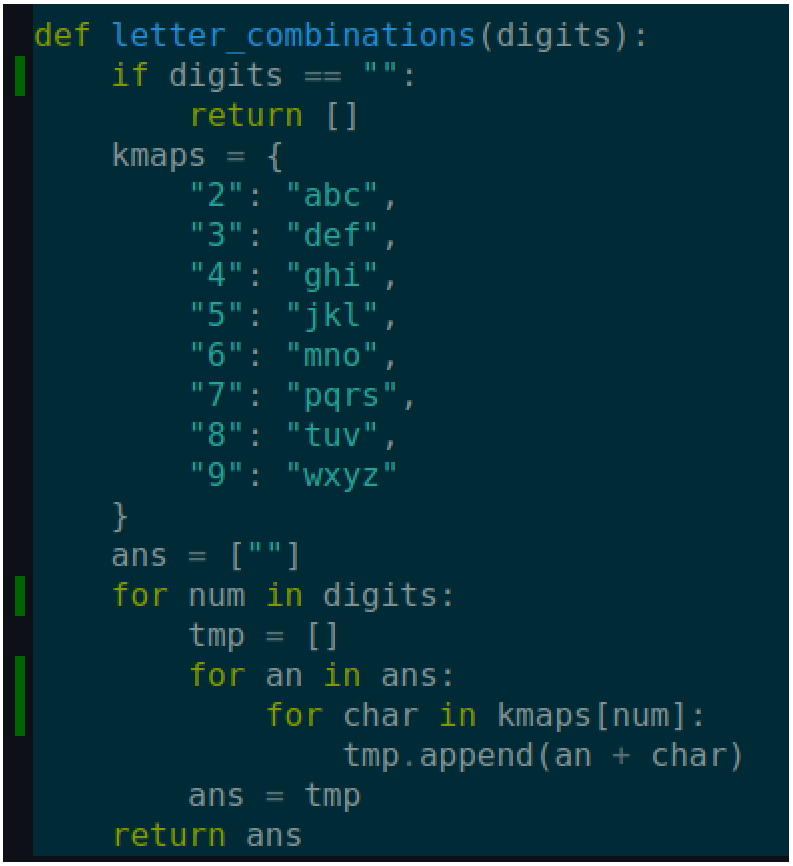


```
(STT-env) root@727db2edd794:~/workdir/algorithms# pynguin --project-path ./algorithms/backtrack/
--output-path ./results/pynguin-results --module-name letter_combination --create-coverage-report
 True --report-dir ./results/cov-report-2 -v
[04:11:49] INFO     Start Pynguin Test Generation…                          generator.py:117
           INFO     Collecting static constants from module under test      generator.py:212
           INFO     No constants found                                      generator.py:215
           INFO     Setting up runtime collection of constants              generator.py:222
           INFO     Analyzed project to create test cluster                  module.py:1282
           INFO     Modules:       1                                         module.py:1283
           INFO     Functions:     1                                         module.py:1284
           INFO     Classes:      11                                         module.py:1285
           INFO     Using seed 1740629508724132707                          generator.py:200
           INFO     Using strategy: Algorithm.DYNAMOSA      generationalgorithmfactory.py:287
           INFO     Instantiated 9 fitness functions        generationalgorithmfactory.py:374
           INFO     Using CoverageArchive                   generationalgorithmfactory.py:329
           INFO     Using selection function:               generationalgorithmfactory.py:304
                    Selection.TOURNAMENT_SELECTION
           INFO     No stopping condition configured!       generationalgorithmfactory.py:118
           INFO     Using fallback timeout of 600 seconds   generationalgorithmfactory.py:119
           INFO     Using crossover function:               generationalgorithmfactory.py:317
                    SinglePointRelativeCrossOver
```

Coverage for **pynguin = 100%**

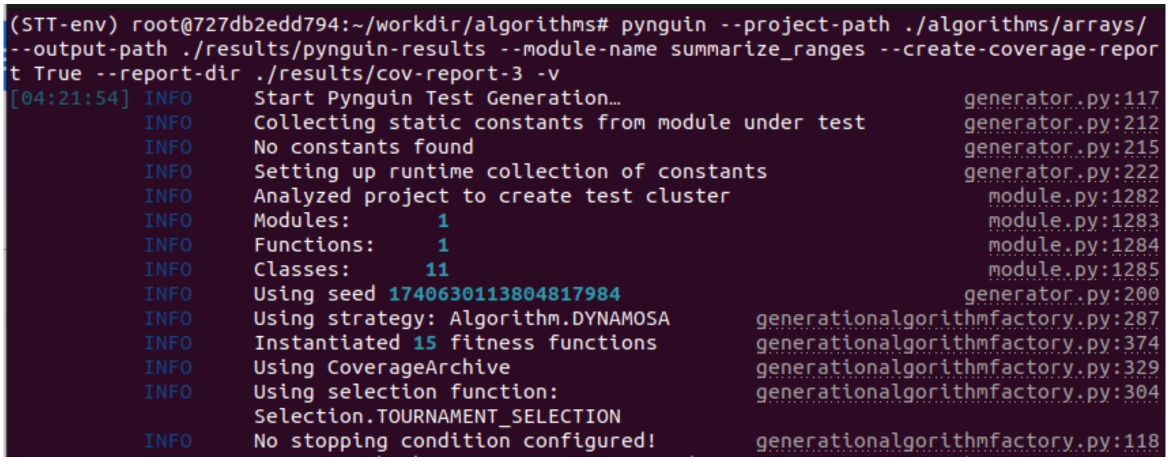


```
Pynguin coverage report for module 'letter_combination'

Achieved 100.00% branch coverage: 1/1 branchless code objects covered. 8/8 branches covered.
```



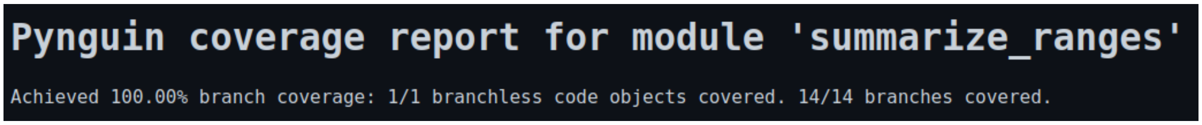
```
def letter_combinations(digits):
    if digits == "":
        return []
    kmaps = {
        "2": "abc",
        "3": "def",
        "4": "ghi",
        "5": "jkl",
        "6": "mno",
        "7": "pqrs",
        "8": "tuv",
        "9": "wxyz"
    }
    ans = [""]
    for num in digits:
        tmp = []
        for an in ans:
            for char in kmaps[num]:
                tmp.append(an + char)
        ans = tmp
    return ans
```

c. arrays/summarize_ranges.py

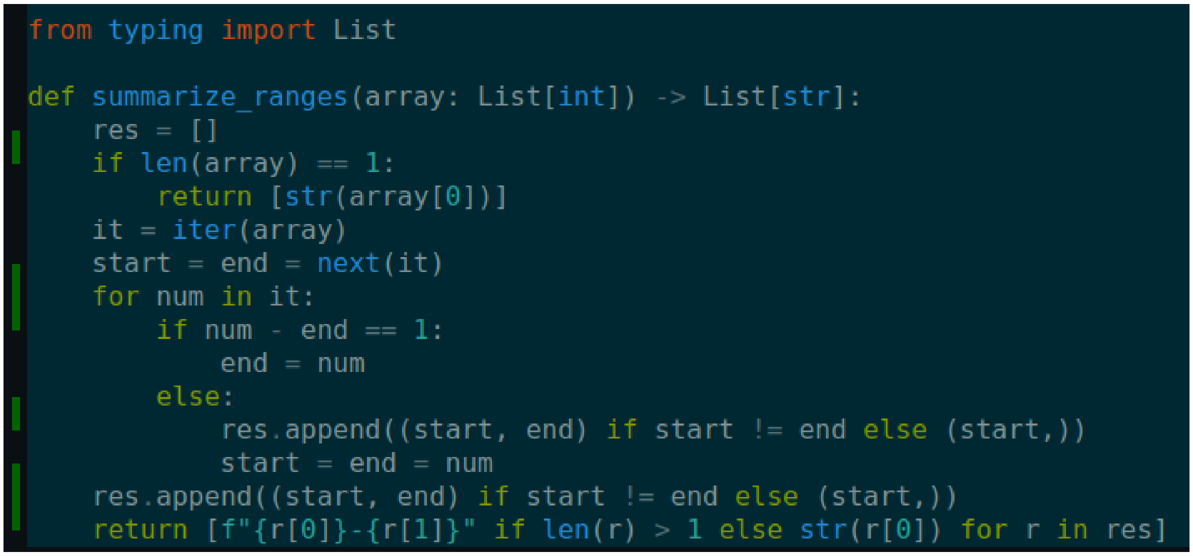


```
(STT-env) root@727db2edd794:~/workdir/algorithms# pynguin --project-path ./algorithms/arrays/
--output-path ./results/pynguin-results --module-name summarize_ranges --create-coverage-repor
t True --report-dir ./results/cov-report-3 -v
[04:21:54] INFO     Start Pynguin Test Generation…                           generator.py:117
           INFO     Collecting static constants from module under test       generator.py:212
           INFO     No constants found                                       generator.py:215
           INFO     Setting up runtime collection of constants               generator.py:222
           INFO     Analyzed project to create test cluster                  module.py:1282
           INFO     Modules:       1                                         module.py:1283
           INFO     Functions:     1                                         module.py:1284
           INFO     Classes:      11                                         module.py:1285
           INFO     Using seed 1740630113804817984                           generator.py:200
           INFO     Using strategy: Algorithm.DYNAMOSA       generationalgorithmfactory.py:287
           INFO     Instantiated 15 fitness functions        generationalgorithmfactory.py:374
           INFO     Using CoverageArchive                    generationalgorithmfactory.py:329
           INFO     Using selection function:                generationalgorithmfactory.py:304
                    Selection.TOURNAMENT_SELECTION
           INFO     No stopping condition configured!        generationalgorithmfactory.py:118
```

Coverage for **pynguin = 100%**



```
Pynguin coverage report for module 'summarize_ranges'

Achieved 100.00% branch coverage: 1/1 branchless code objects covered. 14/14 branches covered.
```



```python
from typing import List

def summarize_ranges(array: List[int]) -> List[str]:
    res = []
    if len(array) == 1:
        return [str(array[0])]
    it = iter(array)
    start = end = next(it)
    for num in it:
        if num - end == 1:
            end = num
        else:
            res.append((start, end) if start != end else (start,))
            start = end = num
    res.append((start, end) if start != end else (start,))
    return [f"{r[0]}-{r[1]}" if len(r) > 1 else str(r[0]) for r in res]
```

Result and Analysis:

1. **Comparing test suite A v/s. B:**

File Name	**Coverage A (%)**	**Coverage B (%)**
arrays/limit.py	88	100
backtrack/letter_combination.py	92	100
arrays/summarize_ranges.py	93	100
Average	**91**	**100**

Note: The above analysis is performed only for 3 files, which had incomplete coverage in test cases provided by the developer.

2. **Uncovered Scenario**
    - The test cases provided by developer (test suite A) did not account for base cases like empty arrays or empty strings.
    - These test cases were required for 100% code coverage.

Conclusion:

In this lab, I learned how to measure code coverage and generate test cases using **pynguin**. Sometimes existing test cases do not always cover the entire code, which can leave some parts untested. By generating additional test cases using **pynguin**, the coverage was improved and some cases were uncovered where the code might break.

This lab helped in understanding the importance of writing good test cases to ensure the correctness and reliability of a program. Automated tools can assist in this process, but they have their own limitations, such as generating unnecessary or redundant test cases. Overall, this lab gave hands-on experience in testing and helped us appreciate why code coverage analysis is important in software development.

Lab 6

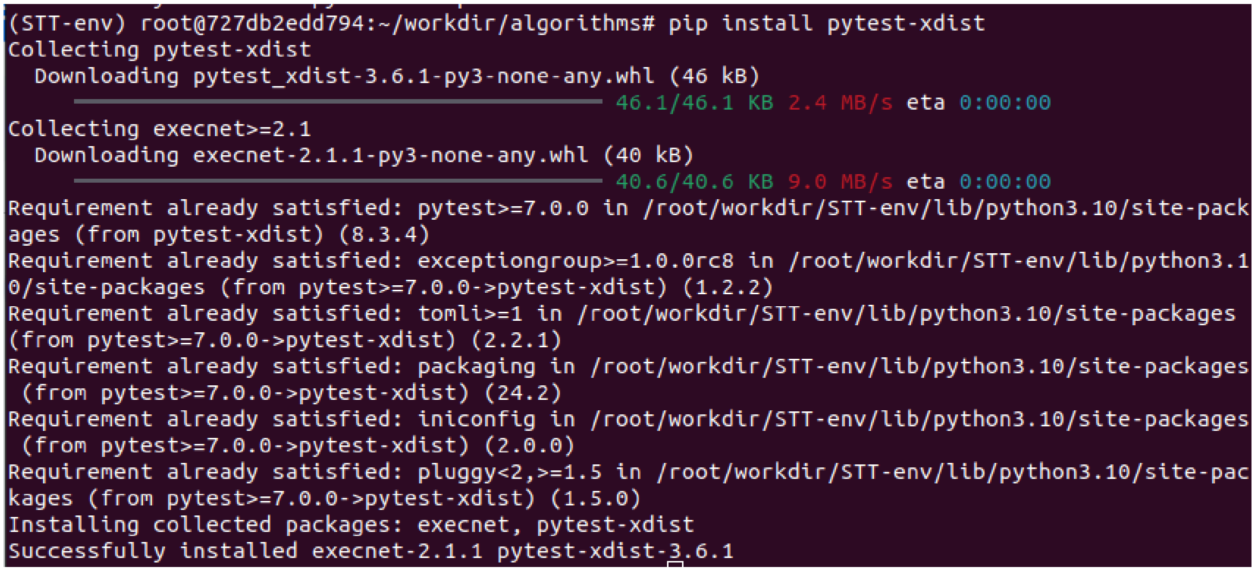
Introduction:

In this lab, we will explore the concept of **test parallelization** in Python, which helps us to speed up test execution by running multiple tests simultaneously. While parallel execution can improve efficiency, it also comes with challenges like **flaky tests**, **race conditions**, and **resource conflicts**.

We will be working with the **keon/algorithms** repository and use tools like **pytest-xdist** and **pytest-run-parallel** to perform and analyze test parallelization. By executing tests sequentially and in parallel, we will identify **unstable tests**, **measure speed improvements**, and evaluate how ready the repository is for parallel execution. This will help us understand the trade-offs and best practices for writing reliable and parallel-friendly test cases.

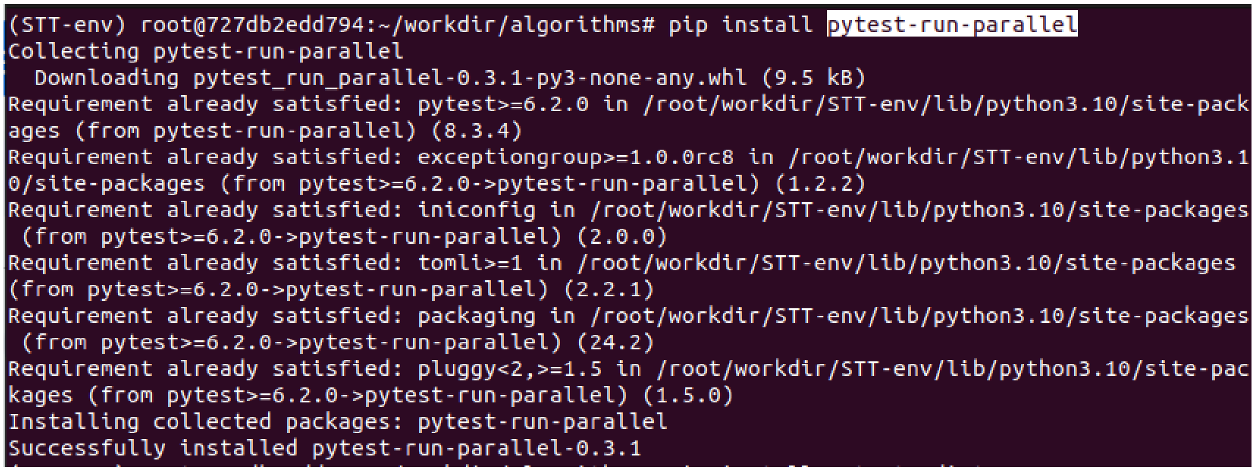
Tools and Setup:

1. **Installing pytest-xdist**



```
(STT-env) root@727db2edd794:~/workdir/algorithms# pip install pytest-xdist
Collecting pytest-xdist
  Downloading pytest_xdist-3.6.1-py3-none-any.whl (46 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.1/46.1 KB 2.4 MB/s eta 0:00:00
Collecting execnet>=2.1
  Downloading execnet-2.1.1-py3-none-any.whl (40 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 40.6/40.6 KB 9.0 MB/s eta 0:00:00
Requirement already satisfied: pytest>=7.0.0 in /root/workdir/STT-env/lib/python3.10/site-pack
ages (from pytest-xdist) (8.3.4)
Requirement already satisfied: exceptiongroup>=1.0.0rc8 in /root/workdir/STT-env/lib/python3.1
0/site-packages (from pytest>=7.0.0->pytest-xdist) (1.2.2)
Requirement already satisfied: tomli>=1 in /root/workdir/STT-env/lib/python3.10/site-packages
(from pytest>=7.0.0->pytest-xdist) (2.2.1)
Requirement already satisfied: packaging in /root/workdir/STT-env/lib/python3.10/site-packages
 (from pytest>=7.0.0->pytest-xdist) (24.2)
Requirement already satisfied: iniconfig in /root/workdir/STT-env/lib/python3.10/site-packages
 (from pytest>=7.0.0->pytest-xdist) (2.0.0)
Requirement already satisfied: pluggy<2,>=1.5 in /root/workdir/STT-env/lib/python3.10/site-pac
kages (from pytest>=7.0.0->pytest-xdist) (1.5.0)
Installing collected packages: execnet, pytest-xdist
Successfully installed execnet-2.1.1 pytest-xdist-3.6.1
```

2. Installing pytest-run-parallel

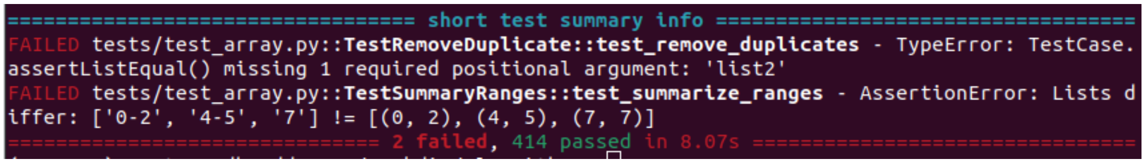


```
(STT-env) root@727db2edd794:~/workdir/algorithms# pip install pytest-run-parallel
Collecting pytest-run-parallel
  Downloading pytest_run_parallel-0.3.1-py3-none-any.whl (9.5 kB)
Requirement already satisfied: pytest>=6.2.0 in /root/workdir/STT-env/lib/python3.10/site-pack
ages (from pytest-run-parallel) (8.3.4)
Requirement already satisfied: exceptiongroup>=1.0.0rc8 in /root/workdir/STT-env/lib/python3.1
0/site-packages (from pytest>=6.2.0->pytest-run-parallel) (1.2.2)
Requirement already satisfied: iniconfig in /root/workdir/STT-env/lib/python3.10/site-packages
 (from pytest>=6.2.0->pytest-run-parallel) (2.0.0)
Requirement already satisfied: tomli>=1 in /root/workdir/STT-env/lib/python3.10/site-packages
(from pytest>=6.2.0->pytest-run-parallel) (2.2.1)
Requirement already satisfied: packaging in /root/workdir/STT-env/lib/python3.10/site-packages
 (from pytest>=6.2.0->pytest-run-parallel) (24.2)
Requirement already satisfied: pluggy<2,>=1.5 in /root/workdir/STT-env/lib/python3.10/site-pac
kages (from pytest>=6.2.0->pytest-run-parallel) (1.5.0)
Installing collected packages: pytest-run-parallel
Successfully installed pytest-run-parallel-0.3.1
```

Methodology:

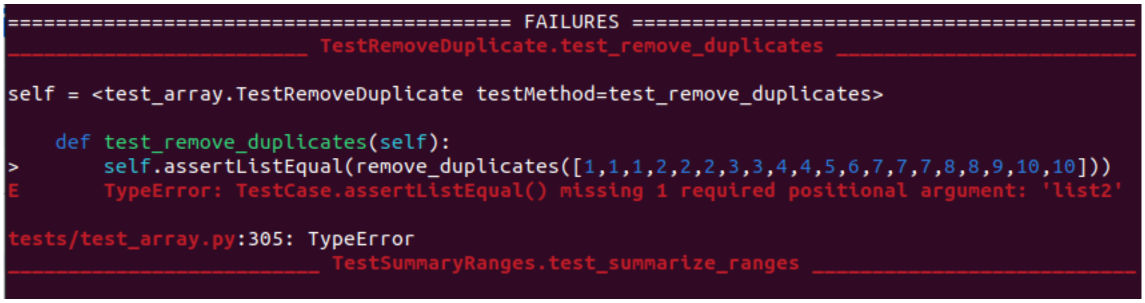
1. Identifying faulty and flaky test cases

a. Faulty test cases - 2



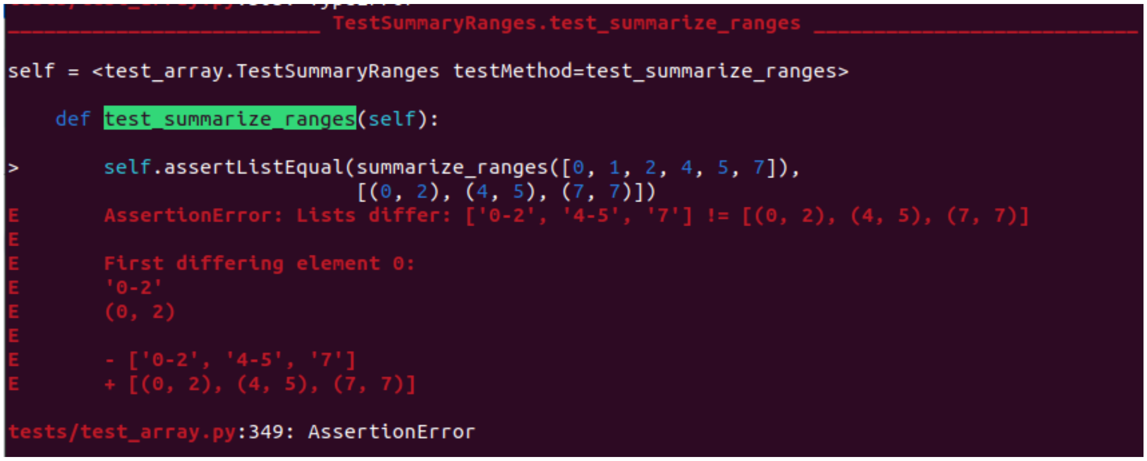
```
=================================== short test summary info ====================================
FAILED tests/test_array.py::TestRemoveDuplicate::test_remove_duplicates - TypeError: TestCase.
assertListEqual() missing 1 required positional argument: 'list2'
FAILED tests/test_array.py::TestSummaryRanges::test_summarize_ranges - AssertionError: Lists d
iffer: ['0-2', '4-5', '7'] != [(0, 2), (4, 5), (7, 7)]
============================== 2 failed, 414 passed in 8.07s ===============================
```

- test_array/test_remove_duplicates



```
========================================== FAILURES ===========================================
_________________________ TestRemoveDuplicate.test_remove_duplicates __________________________

self = <test_array.TestRemoveDuplicate testMethod=test_remove_duplicates>

    def test_remove_duplicates(self):
>       self.assertListEqual(remove_duplicates([1,1,1,2,2,2,3,3,4,4,5,6,7,7,7,8,8,9,10,10]))
E       TypeError: TestCase.assertListEqual() missing 1 required positional argument: 'list2'

tests/test_array.py:305: TypeError
__________________________ TestSummaryRanges.test_summarize_ranges ___________________________
```

- test_array/test_summarize_ranges

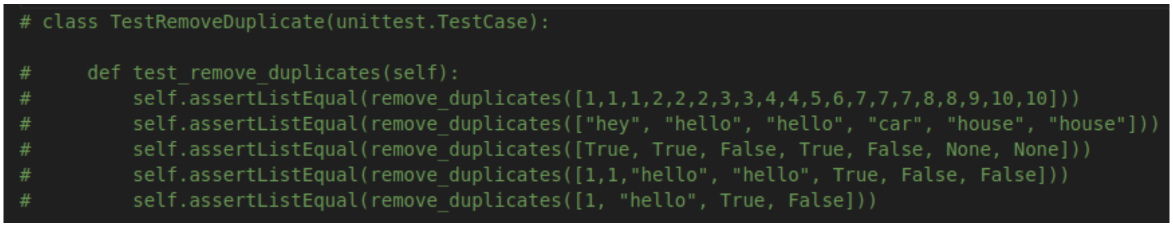


```
__________________________ TestSummaryRanges.test_summarize_ranges __________________________

self = <test_array.TestSummaryRanges testMethod=test_summarize_ranges>

    def test_summarize_ranges(self):

>       self.assertListEqual(summarize_ranges([0, 1, 2, 4, 5, 7]),
                             [(0, 2), (4, 5), (7, 7)])
E       AssertionError: Lists differ: ['0-2', '4-5', '7'] != [(0, 2), (4, 5), (7, 7)]
E
E       First differing element 0:
E       '0-2'
E       (0, 2)
E
E       - ['0-2', '4-5', '7']
E       + [(0, 2), (4, 5), (7, 7)]

tests/test_array.py:349: AssertionError
```

b. **Flaky test cases - 0**

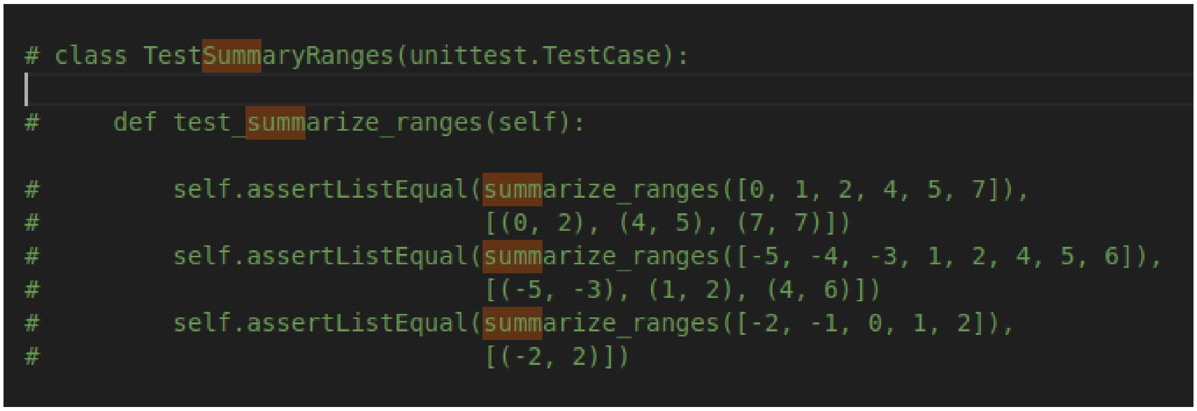
2. Removing the faulty test cases

a. test_array/test_remove_duplicates



```
# class TestRemoveDuplicate(unittest.TestCase):

#     def test_remove_duplicates(self):
#         self.assertListEqual(remove_duplicates([1,1,1,2,2,2,3,3,4,4,5,6,7,7,7,8,8,9,10,10]))
#         self.assertListEqual(remove_duplicates(["hey", "hello", "hello", "car", "house", "house"]))
#         self.assertListEqual(remove_duplicates([True, True, False, True, False, None, None]))
#         self.assertListEqual(remove_duplicates([1,1,"hello", "hello", True, False, False]))
#         self.assertListEqual(remove_duplicates([1, "hello", True, False]))
```

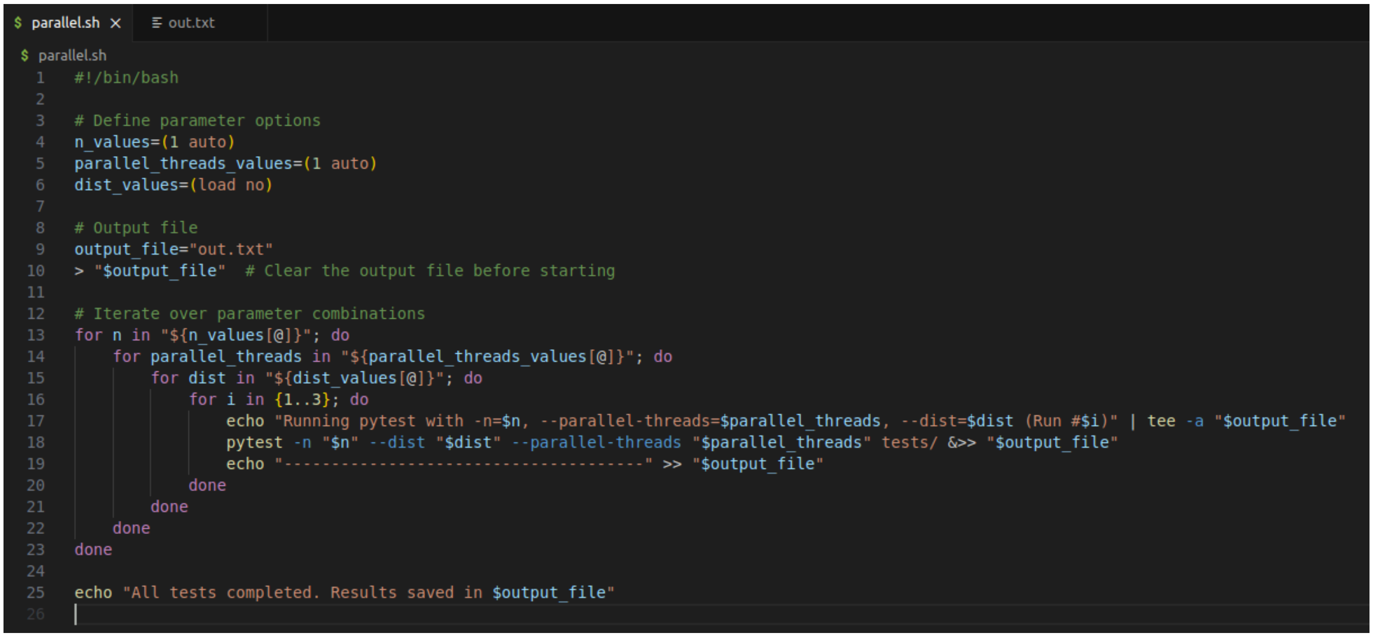
b. test_array/test_summarize_ranges



```
# class TestSummaryRanges(unittest.TestCase):

#     def test_summarize_ranges(self):

#         self.assertListEqual(summarize_ranges([0, 1, 2, 4, 5, 7]),
#                              [(0, 2), (4, 5), (7, 7)])
#         self.assertListEqual(summarize_ranges([-5, -4, -3, 1, 2, 4, 5, 6]),
#                              [(-5, -3), (1, 2), (4, 6)])
#         self.assertListEqual(summarize_ranges([-2, -1, 0, 1, 2]),
#                              [(-2, 2)])
```

3. Parallel execution

- Bash script to run all the combinations and store the output in text file for later analysis



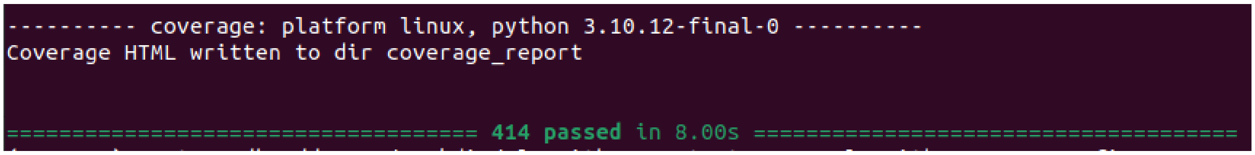
```bash
#!/bin/bash

# Define parameter options
n_values=(1 auto)
parallel_threads_values=(1 auto)
dist_values=(load no)

# Output file
output_file="out.txt"
> "$output_file"  # Clear the output file before starting

# Iterate over parameter combinations
for n in "${n_values[@]}"; do
    for parallel_threads in "${parallel_threads_values[@]}"; do
        for dist in "${dist_values[@]}"; do
            for i in {1..3}; do
                echo "Running pytest with -n=$n, --parallel-threads=$parallel_threads, --dist=$dist (Run #$i)" | tee -a "$output_file"
                pytest -n "$n" --dist "$dist" --parallel-threads "$parallel_threads" tests/ &>> "$output_file"
                echo "---------------------------------------" >> "$output_file"
            done
        done
    done
done

echo "All tests completed. Results saved in $output_file"

```

Results:

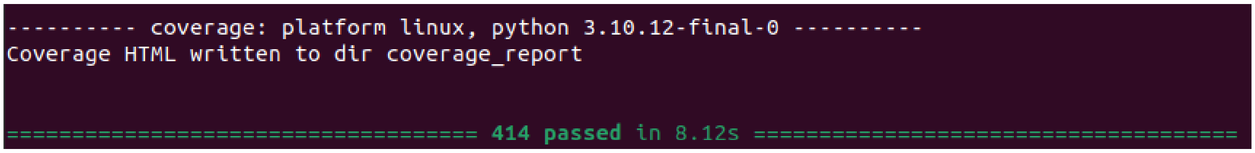
Sequential execution

Sr. No	Time (s)
1	8
2	8.12
3	8.17
Average	**8.10**

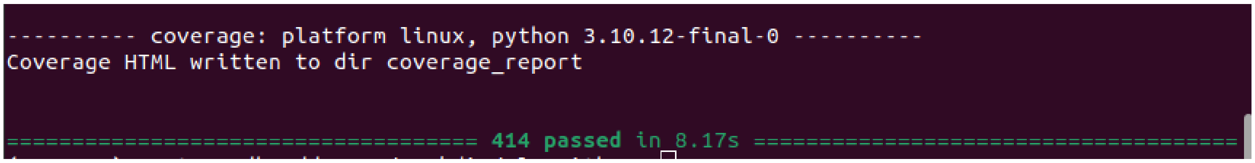
∴ T_{seq} = 8.10 sec



```
---------- coverage: platform linux, python 3.10.12-final-0 ----------
Coverage HTML written to dir coverage_report


==================================== 414 passed in 8.00s =====================================
```



```
---------- coverage: platform linux, python 3.10.12-final-0 ----------
Coverage HTML written to dir coverage_report


==================================== 414 passed in 8.12s =====================================
```



```
---------- coverage: platform linux, python 3.10.12-final-0 ----------
Coverage HTML written to dir coverage_report


==================================== 414 passed in 8.17s =====================================
```

Parallel execution

A. n = 1, parallel-threads = 1, dist = load

Run	Time (s)	Failed tests
1	4.20	0
2	4.17	0
3	4.17	0

- Avg time = 4.18 s
- Avg Failures = 0

B. n = 1, parallel-threads = 1, dist = no

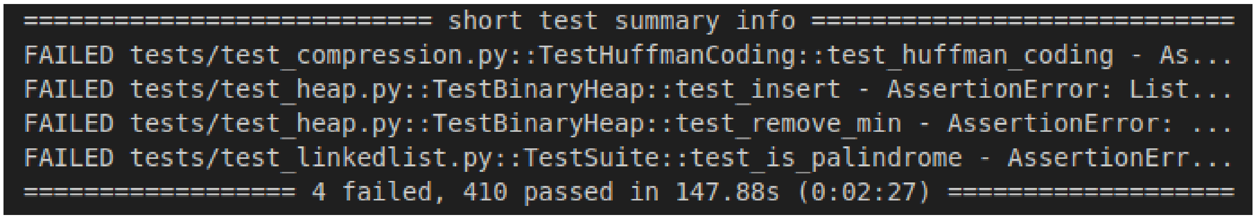
Run	Time (s)	Failed tests
1	4.21	0
2	4.19	0
3	4.18	0

- Avg time = 4.19 s
- Avg Failures = 0

C. n = 1, parallel-threads = auto, dist = load

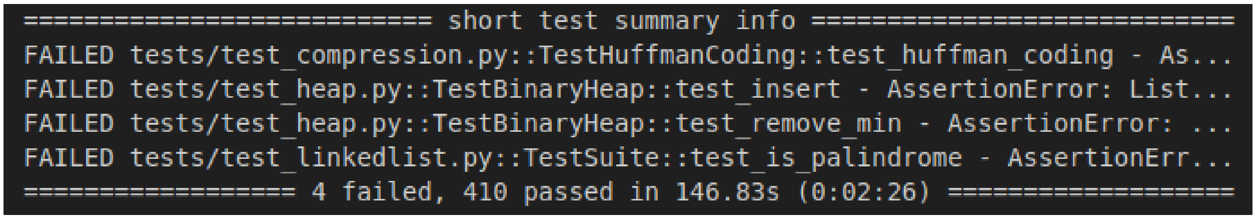
Run	Time (s)	Failed tests
1	147.88	4
2	146.83	4
3	148.07	4

- Failures in run 1:



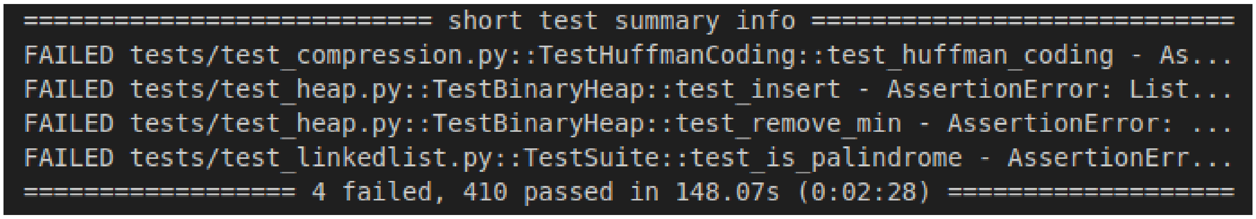
```
=========================== short test summary info ============================
FAILED tests/test_compression.py::TestHuffmanCoding::test_huffman_coding - As...
FAILED tests/test_heap.py::TestBinaryHeap::test_insert - AssertionError: List...
FAILED tests/test_heap.py::TestBinaryHeap::test_remove_min - AssertionError: ...
FAILED tests/test_linkedlist.py::TestSuite::test_is_palindrome - AssertionErr...
================== 4 failed, 410 passed in 147.88s (0:02:27) ===================
```

- Failures in run 2



```
=========================== short test summary info ============================
FAILED tests/test_compression.py::TestHuffmanCoding::test_huffman_coding - As...
FAILED tests/test_heap.py::TestBinaryHeap::test_insert - AssertionError: List...
FAILED tests/test_heap.py::TestBinaryHeap::test_remove_min - AssertionError: ...
FAILED tests/test_linkedlist.py::TestSuite::test_is_palindrome - AssertionErr...
================== 4 failed, 410 passed in 146.83s (0:02:26) ===================
```

- Failures in run 3



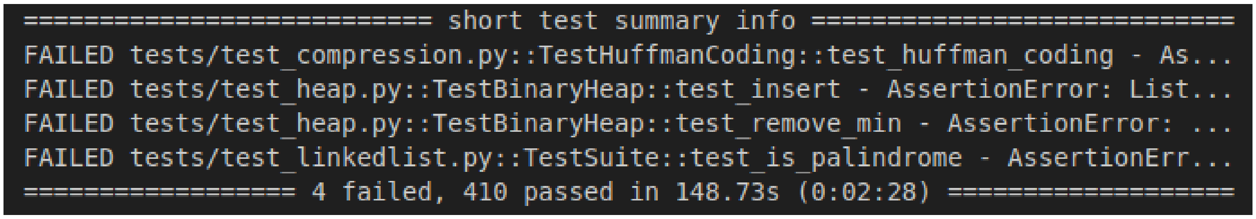
```
=========================== short test summary info ============================
FAILED tests/test_compression.py::TestHuffmanCoding::test_huffman_coding - As...
FAILED tests/test_heap.py::TestBinaryHeap::test_insert - AssertionError: List...
FAILED tests/test_heap.py::TestBinaryHeap::test_remove_min - AssertionError: ...
FAILED tests/test_linkedlist.py::TestSuite::test_is_palindrome - AssertionErr...
================== 4 failed, 410 passed in 148.07s (0:02:28) ===================
```

- Avg time = 147.59 s
- Avg Failures = 4

D. n = 1, parallel-threads = auto, dist = no

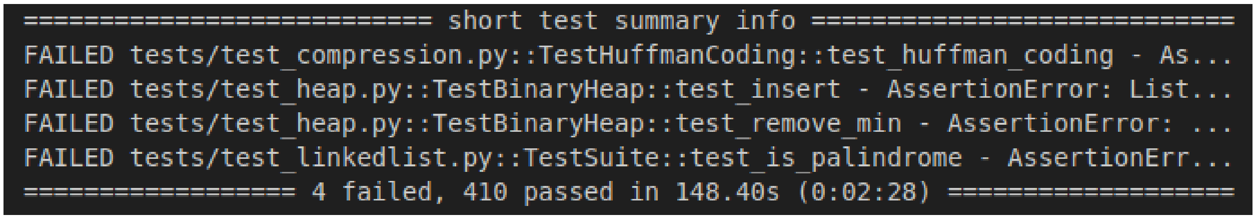
Run	Time (s)	Failed tests
1	148.73	4
2	148.40	4
3	147.79	4

- Failures in run 1:



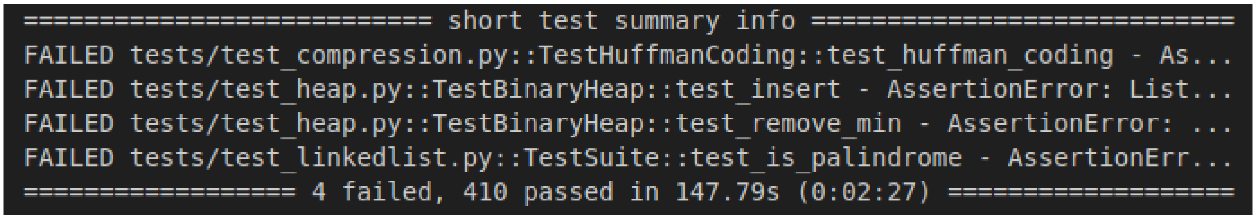
```
=========================== short test summary info ============================
FAILED tests/test_compression.py::TestHuffmanCoding::test_huffman_coding - As...
FAILED tests/test_heap.py::TestBinaryHeap::test_insert - AssertionError: List...
FAILED tests/test_heap.py::TestBinaryHeap::test_remove_min - AssertionError: ...
FAILED tests/test_linkedlist.py::TestSuite::test_is_palindrome - AssertionErr...
================== 4 failed, 410 passed in 148.73s (0:02:28) ===================
```

- Failures in run 2:



```
=========================== short test summary info ============================
FAILED tests/test_compression.py::TestHuffmanCoding::test_huffman_coding - As...
FAILED tests/test_heap.py::TestBinaryHeap::test_insert - AssertionError: List...
FAILED tests/test_heap.py::TestBinaryHeap::test_remove_min - AssertionError: ...
FAILED tests/test_linkedlist.py::TestSuite::test_is_palindrome - AssertionErr...
================== 4 failed, 410 passed in 148.40s (0:02:28) ===================
```

- Failures in run 3:



```
=========================== short test summary info ============================
FAILED tests/test_compression.py::TestHuffmanCoding::test_huffman_coding - As...
FAILED tests/test_heap.py::TestBinaryHeap::test_insert - AssertionError: List...
FAILED tests/test_heap.py::TestBinaryHeap::test_remove_min - AssertionError: ...
FAILED tests/test_linkedlist.py::TestSuite::test_is_palindrome - AssertionErr...
================== 4 failed, 410 passed in 147.79s (0:02:27) ===================
```

- Avg time = 148.31 s
- Avg Failures = 4

E. n = auto, parallel-threads = 1, dist = load

Run	Time (s)	Failed tests
1	3.44	0
2	3.57	0
3	3.57	0

- Avg time = 3.53 s
- Avg Failures = 0

F. n = auto, parallel-threads = 1, dist = no

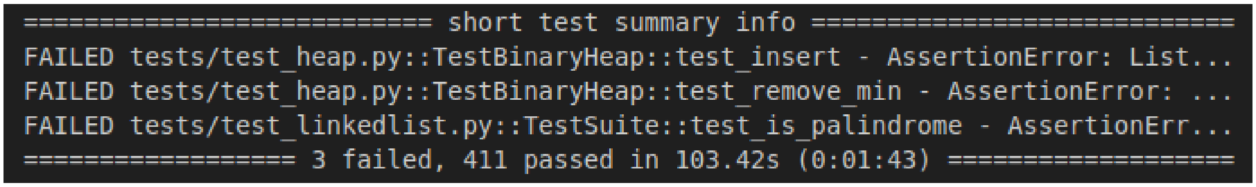
Run	Time (s)	Failed tests
1	3.53	0
2	3.60	0
3	3.57	0

- Avg time = 3.57 s
- Avg Failures = 0

G. n = auto, parallel-threads = auto, dist = load

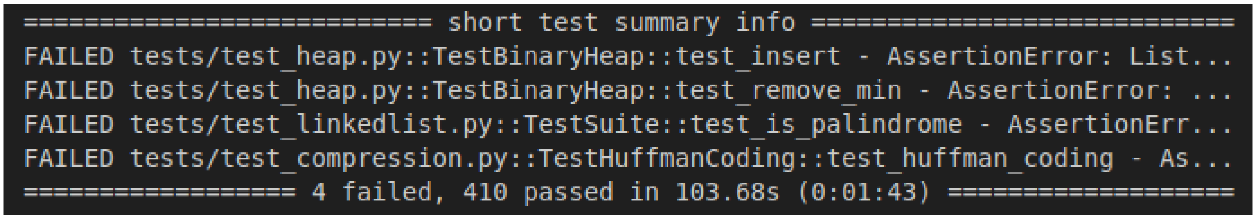
Run	Time (s)	Failed tests
1	103.42	3
2	103.68	4
3	101.35	4

- Failures in run 1:



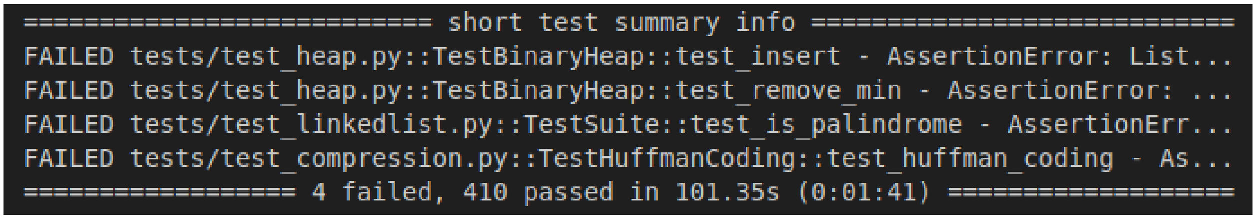
```
========================== short test summary info ============================
FAILED tests/test_heap.py::TestBinaryHeap::test_insert - AssertionError: List...
FAILED tests/test_heap.py::TestBinaryHeap::test_remove_min - AssertionError: ...
FAILED tests/test_linkedlist.py::TestSuite::test_is_palindrome - AssertionErr...
================== 3 failed, 411 passed in 103.42s (0:01:43) ===================
```

- Failures in run 2:



```
========================== short test summary info ============================
FAILED tests/test_heap.py::TestBinaryHeap::test_insert - AssertionError: List...
FAILED tests/test_heap.py::TestBinaryHeap::test_remove_min - AssertionError: ...
FAILED tests/test_linkedlist.py::TestSuite::test_is_palindrome - AssertionErr...
FAILED tests/test_compression.py::TestHuffmanCoding::test_huffman_coding - As...
================== 4 failed, 410 passed in 103.68s (0:01:43) ===================
```

- Failures in run 3:



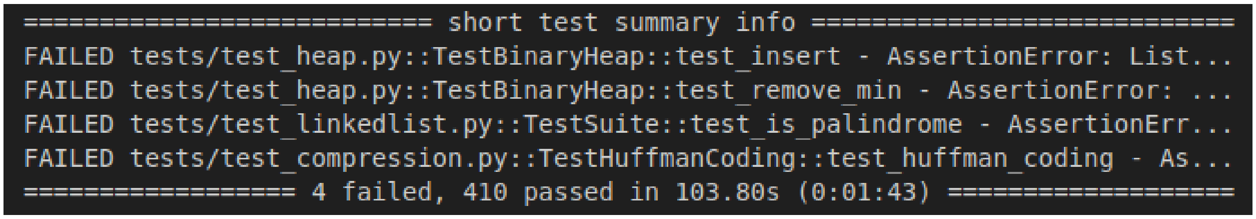
```
========================== short test summary info ============================
FAILED tests/test_heap.py::TestBinaryHeap::test_insert - AssertionError: List...
FAILED tests/test_heap.py::TestBinaryHeap::test_remove_min - AssertionError: ...
FAILED tests/test_linkedlist.py::TestSuite::test_is_palindrome - AssertionErr...
FAILED tests/test_compression.py::TestHuffmanCoding::test_huffman_coding - As...
================== 4 failed, 410 passed in 101.35s (0:01:41) ===================
```

- Avg time = 102.82
- Failures = 3 in run 1, 4 in run 2 and 3

H. n = auto, parallel-threads = auto, dist = no

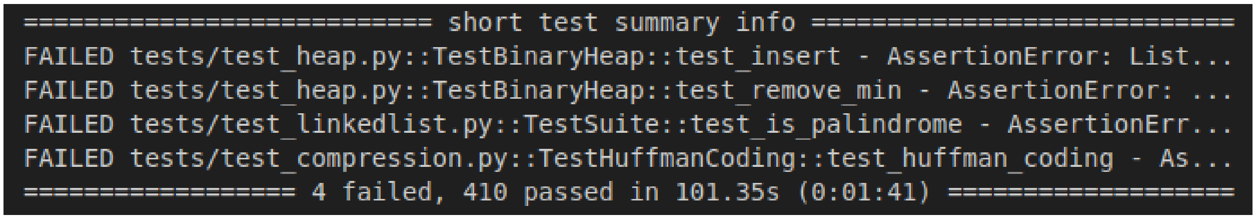
Run	Time (s)	Failed tests
1	103.80	4
2	101.35	4
3	101.23	4

- Failures in run 1:



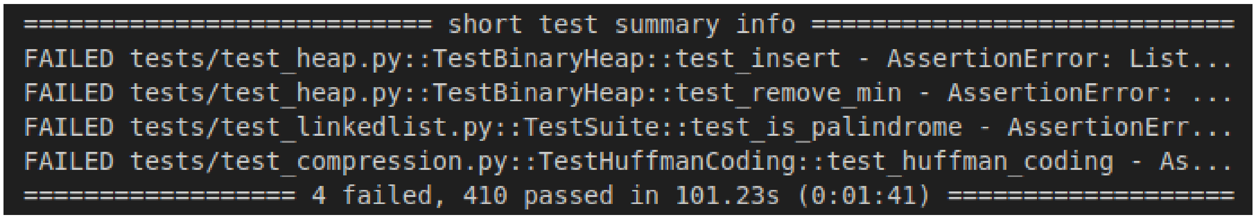
```
=========================== short test summary info ============================
FAILED tests/test_heap.py::TestBinaryHeap::test_insert - AssertionError: List...
FAILED tests/test_heap.py::TestBinaryHeap::test_remove_min - AssertionError: ...
FAILED tests/test_linkedlist.py::TestSuite::test_is_palindrome - AssertionErr...
FAILED tests/test_compression.py::TestHuffmanCoding::test_huffman_coding - As...
================== 4 failed, 410 passed in 103.80s (0:01:43) ===================
```

- Failures in run 2:



```
=========================== short test summary info ============================
FAILED tests/test_heap.py::TestBinaryHeap::test_insert - AssertionError: List...
FAILED tests/test_heap.py::TestBinaryHeap::test_remove_min - AssertionError: ...
FAILED tests/test_linkedlist.py::TestSuite::test_is_palindrome - AssertionErr...
FAILED tests/test_compression.py::TestHuffmanCoding::test_huffman_coding - As...
================== 4 failed, 410 passed in 101.35s (0:01:41) ===================
```

- Failures in run 3:



```
=========================== short test summary info ============================
FAILED tests/test_heap.py::TestBinaryHeap::test_insert - AssertionError: List...
FAILED tests/test_heap.py::TestBinaryHeap::test_remove_min - AssertionError: ...
FAILED tests/test_linkedlist.py::TestSuite::test_is_palindrome - AssertionErr...
FAILED tests/test_compression.py::TestHuffmanCoding::test_huffman_coding - As...
================== 4 failed, 410 passed in 101.23s (0:01:41) ===================
```

- Avg time = 102.13 s
- Avg failures = 4

Analysis:

1. Flaky tests

   There are 4 flaky test in parallel execution
   - **test_insert** from test_heap
   - **test_remove_min** from test_heap
   - **test_is_palindrome** from test_linkedlist
   - **test_huffman_coding** from test_compression

2. Speedup w.r.t. serial execution.

n	parallel-threads	dist	Avg time (s)	Speedup	Failures
1	1	load	4.18	1.938	0
1	1	no	4.19	1.933	0
1	auto	load	147.59	0.0548	4
1	auto	no	148.31	0.0546	4
auto	1	load	3.53	2.295	0
auto	1	no	3.57	2.269	0
auto	auto	load	102.82	0.0788	4
auto	auto	no	102.13	0.0793	4

3. **test_huffman_coding** is the only test which **opens a file**.

Conclusion:

In this lab, we gained practical experience in test parallelization and observed both its **benefits and challenges**. We found that parallel execution can significantly reduce test time when there are no flaky tests, but in some cases **flaky tests and failures** cause the time to increase instead of decreasing, due to issues like shared resources or timing conflicts.

By analyzing different parallelization modes and worker counts, we learned how to **optimize test execution while minimizing failures**. We also documented problematic test cases and suggested improvements for better parallel testing. This lab helped in understanding why ensuring **thread safety and stable test design** is crucial for effective test parallelization in large projects.

Lab 7

Introduction:

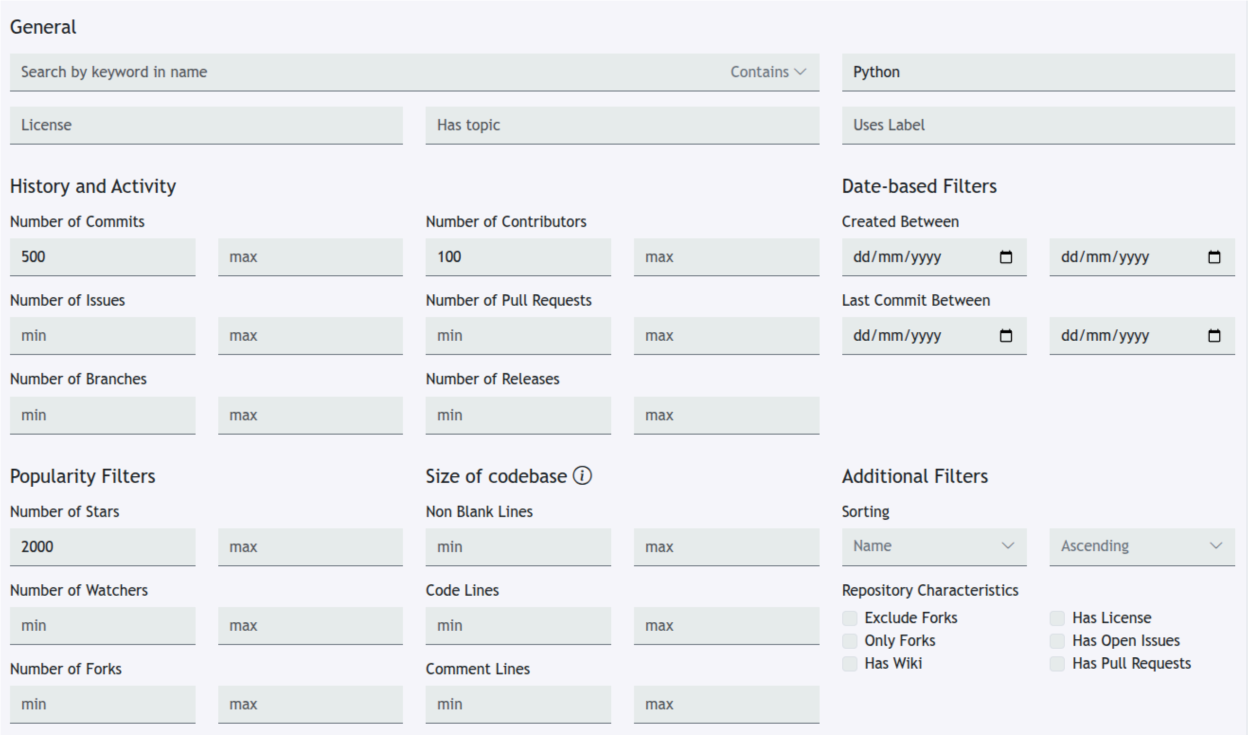
In this lab, we will look at vulnerability analysis in open-source Python projects using **Bandit**. Bandit is a **static** code analysis tool, which helps us in identifying vulnerabilities in software that can lead to serious risks. Automated tools like Bandit help in detecting them early.

We will select three large-scale open-source repositories, analyze them using Bandit, and study the different types of security issues found by bandit. This includes categorizing them based on **confidence levels**, **severity**, and **CWE** (**Common Weakness Enumeration**). We will also investigate when vulnerabilities are introduced and fixed.

By the end of this lab, we will have a deeper understanding of security vulnerabilities in Python projects, how to analyze them effectively, and the importance of writing secure code.

Setup:

1. Repository Selection Criterion:



General

Python

History and Activity

Number of Commits: 500

Number of Contributors: 100

Popularity Filters

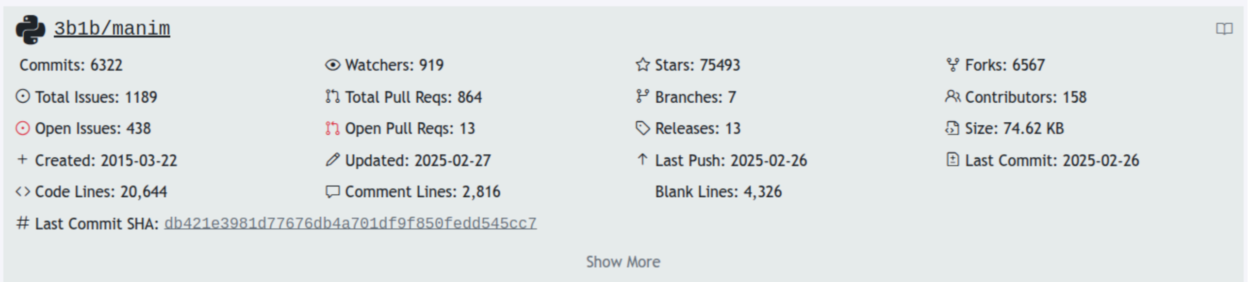
Number of Stars: 2000

Additional Filters

Sorting: Name, Ascending

- Language Python
- Minimum 500 commits
- Minimum 100 contributors
- Minimum 2000 stars

2. Selected repositories:

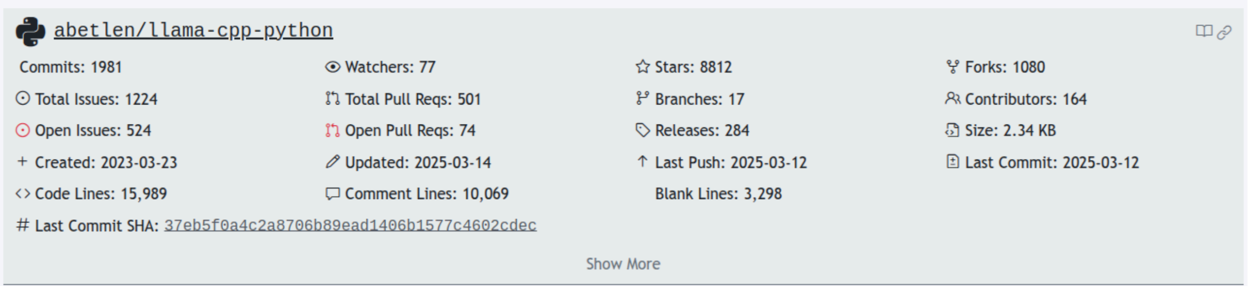


3b1b/manim

Commits: 6322	Watchers: 919	Stars: 75493	Forks: 6567
Total Issues: 1189	Total Pull Reqs: 864	Branches: 7	Contributors: 158
Open Issues: 438	Open Pull Reqs: 13	Releases: 13	Size: 74.62 KB
+ Created: 2015-03-22	Updated: 2025-02-27	Last Push: 2025-02-26	Last Commit: 2025-02-26
<> Code Lines: 20,644	Comment Lines: 2,816	Blank Lines: 4,326	

Last Commit SHA: db421e3981d77676db4a701df9f850fedd545cc7

- 6k commits
- 75k stars
- 158 contributors

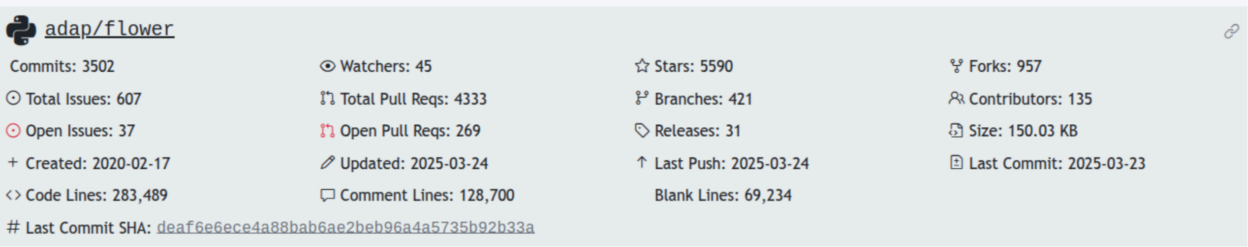


abetlen/llama-cpp-python

Commits: 1981	Watchers: 77	Stars: 8812	Forks: 1080
Total Issues: 1224	Total Pull Reqs: 501	Branches: 17	Contributors: 164
Open Issues: 524	Open Pull Reqs: 74	Releases: 284	Size: 2.34 KB
+ Created: 2023-03-23	Updated: 2025-03-14	Last Push: 2025-03-12	Last Commit: 2025-03-12
<> Code Lines: 15,989	Comment Lines: 10,069	Blank Lines: 3,298	

Last Commit SHA: 37eb5f0a4c2a8706b89ead1406b1577c4602cdec

- 2k commits
- 9k stars
- 164 contributors



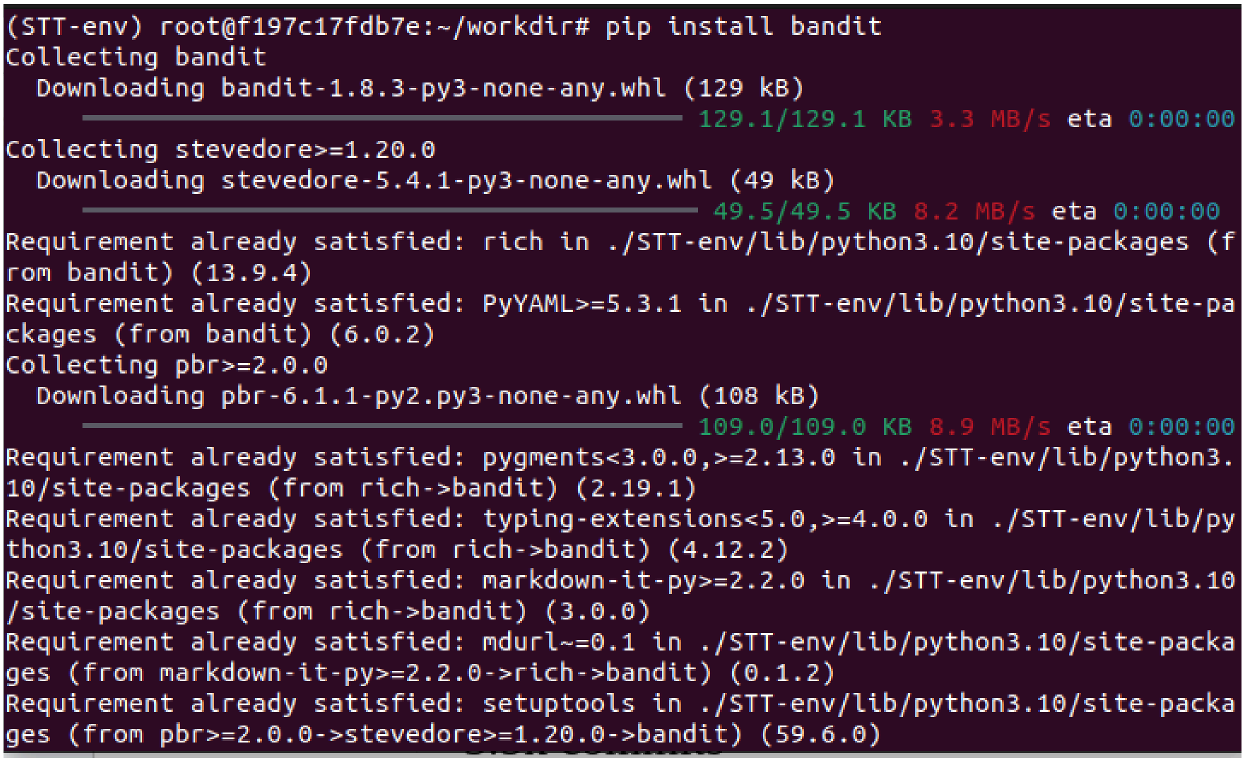
adap/flower

Commits: 3502	Watchers: 45	Stars: 5590	Forks: 957
Total Issues: 607	Total Pull Reqs: 4333	Branches: 421	Contributors: 135
Open Issues: 37	Open Pull Reqs: 269	Releases: 31	Size: 150.03 KB
+ Created: 2020-02-17	Updated: 2025-03-24	Last Push: 2025-03-24	Last Commit: 2025-03-23
<> Code Lines: 283,489	Comment Lines: 128,700	Blank Lines: 69,234	

Last Commit SHA: deaf6e6ece4a88bab6ae2beb96a4a5735b92b33a

- 3.5k commits
- 5.5k stars
- 135 contributors

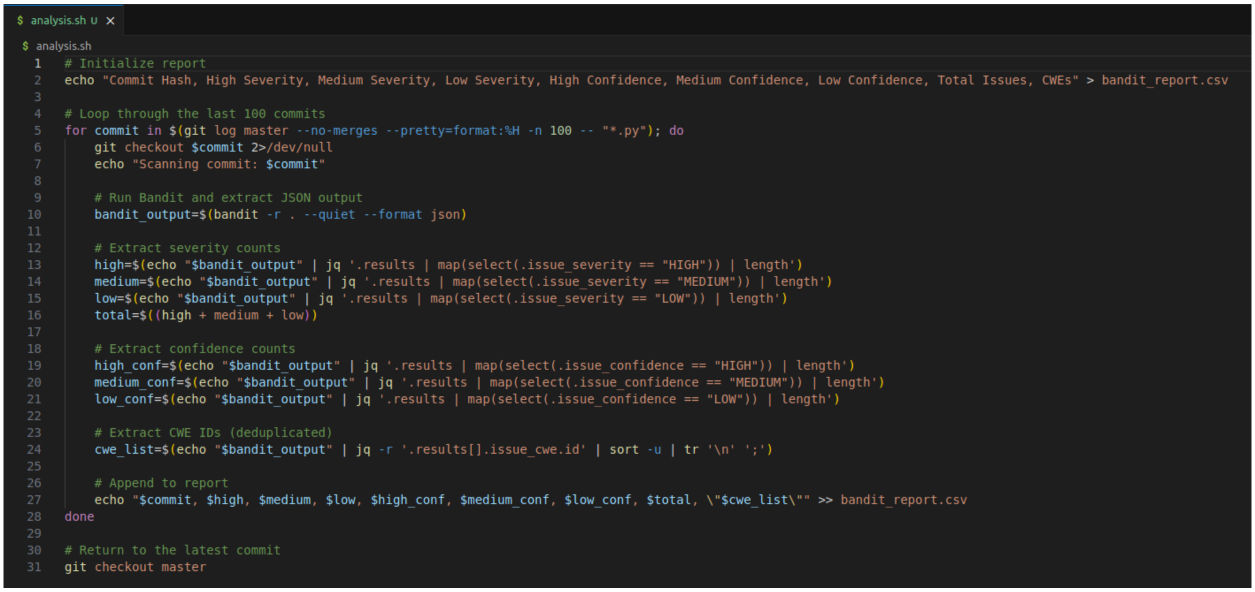
3. Installing bandit:



```
(STT-env) root@f197c17fdb7e:~/workdir# pip install bandit
Collecting bandit
  Downloading bandit-1.8.3-py3-none-any.whl (129 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.1/129.1 KB 3.3 MB/s eta 0:00:00
Collecting stevedore>=1.20.0
  Downloading stevedore-5.4.1-py3-none-any.whl (49 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.5/49.5 KB 8.2 MB/s eta 0:00:00
Requirement already satisfied: rich in ./STT-env/lib/python3.10/site-packages (f
rom bandit) (13.9.4)
Requirement already satisfied: PyYAML>=5.3.1 in ./STT-env/lib/python3.10/site-pa
ckages (from bandit) (6.0.2)
Collecting pbr>=2.0.0
  Downloading pbr-6.1.1-py2.py3-none-any.whl (108 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 109.0/109.0 KB 8.9 MB/s eta 0:00:00
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in ./STT-env/lib/python3.
10/site-packages (from rich->bandit) (2.19.1)
Requirement already satisfied: typing-extensions<5.0,>=4.0.0 in ./STT-env/lib/py
thon3.10/site-packages (from rich->bandit) (4.12.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in ./STT-env/lib/python3.10
/site-packages (from rich->bandit) (3.0.0)
Requirement already satisfied: mdurl~=0.1 in ./STT-env/lib/python3.10/site-packa
ges (from markdown-it-py>=2.2.0->rich->bandit) (0.1.2)
Requirement already satisfied: setuptools in ./STT-env/lib/python3.10/site-packa
ges (from pbr>=2.0.0->stevedore>=1.20.0->bandit) (59.6.0)
```

Methodology:

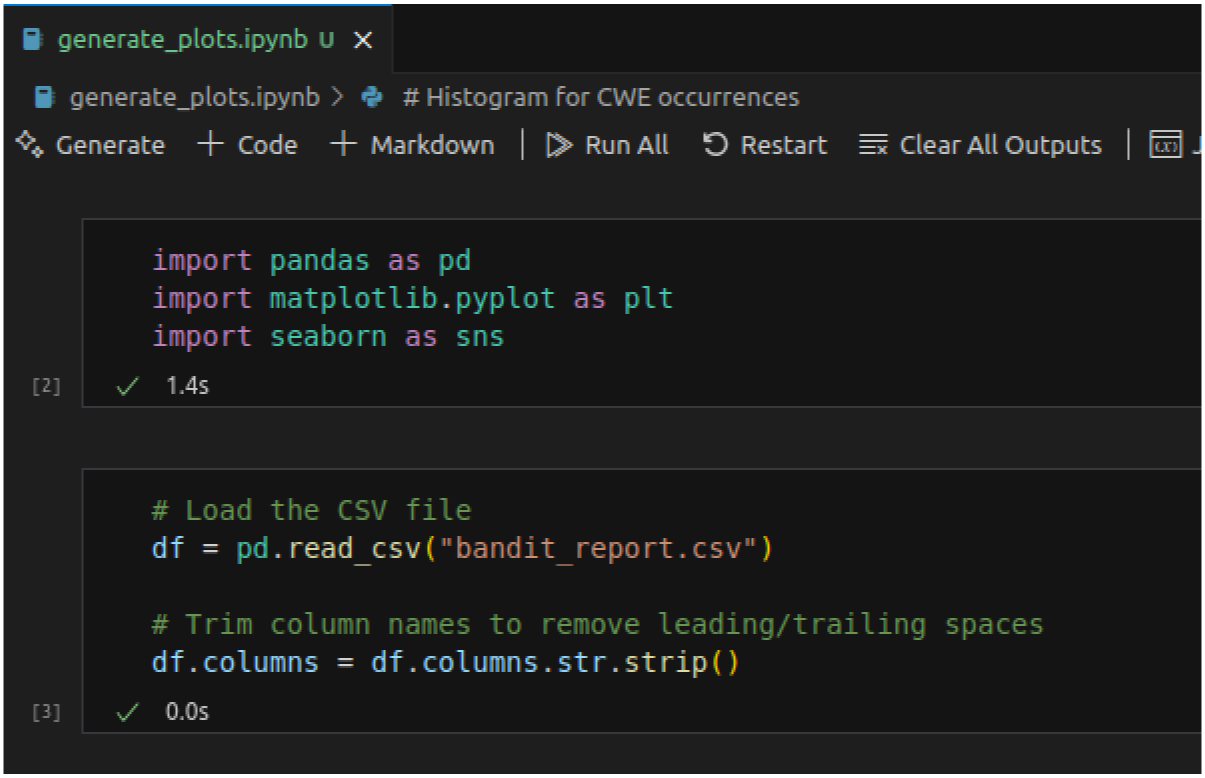
1. Writing bash script to generate the necessary data:



```bash
# Initialize report
echo "Commit Hash, High Severity, Medium Severity, Low Severity, High Confidence, Medium Confidence, Low Confidence, Total Issues, CWEs" > bandit_report.csv

# Loop through the last 100 commits
for commit in $(git log master --no-merges --pretty=format:%H -n 100 -- "*.py"); do
    git checkout $commit 2>/dev/null
    echo "Scanning commit: $commit"

    # Run Bandit and extract JSON output
    bandit_output=$(bandit -r . --quiet --format json)

    # Extract severity counts
    high=$(echo "$bandit_output" | jq '.results | map(select(.issue_severity == "HIGH")) | length')
    medium=$(echo "$bandit_output" | jq '.results | map(select(.issue_severity == "MEDIUM")) | length')
    low=$(echo "$bandit_output" | jq '.results | map(select(.issue_severity == "LOW")) | length')
    total=$((high + medium + low))

    # Extract confidence counts
    high_conf=$(echo "$bandit_output" | jq '.results | map(select(.issue_confidence == "HIGH")) | length')
    medium_conf=$(echo "$bandit_output" | jq '.results | map(select(.issue_confidence == "MEDIUM")) | length')
    low_conf=$(echo "$bandit_output" | jq '.results | map(select(.issue_confidence == "LOW")) | length')

    # Extract CWE IDs (deduplicated)
    cwe_list=$(echo "$bandit_output" | jq -r '.results[].issue_cwe.id' | sort -u | tr '\n' ';')

    # Append to report
    echo "$commit, $high, $medium, $low, $high_conf, $medium_conf, $low_conf, $total, \"$cwe_list\"" >> bandit_report.csv
done

# Return to the latest commit
git checkout master
```

- We get the last 100 non-merge commits in the default branch.
- We checkout the commit.
- We generate bandit output in **json** format.
- We count the number of issues using the **jq** command.
- We use the data in the ‘**results**' key.
- We use **map** functionality to count the number of issues.
- We store the statistics in a **csv file**.

2. Writing a script to generate plots from the csv file called “**generate_plots.ipynb**”.
    - Imports and loading the csv file



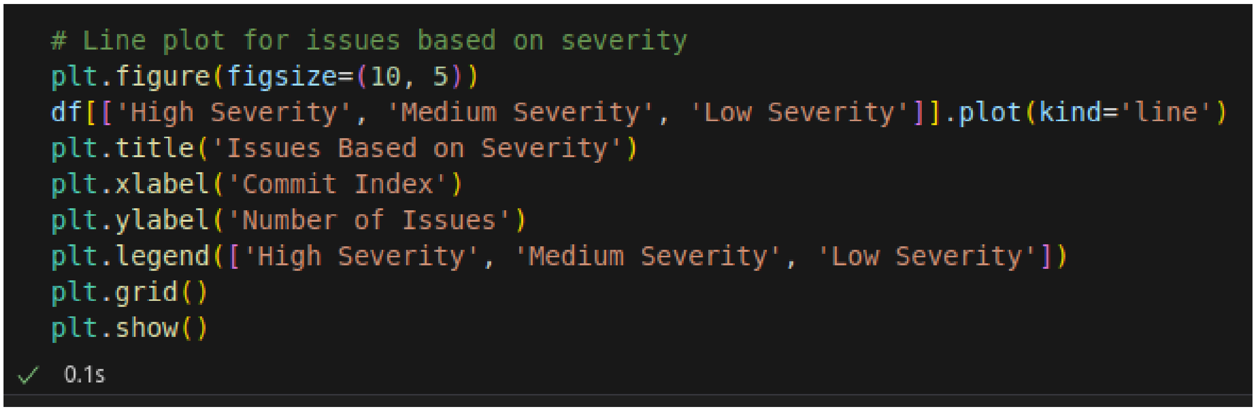
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[2] ✓ 1.4s

```
# Load the CSV file
df = pd.read_csv("bandit_report.csv")

# Trim column names to remove leading/trailing spaces
df.columns = df.columns.str.strip()
```

[3] ✓ 0.0s

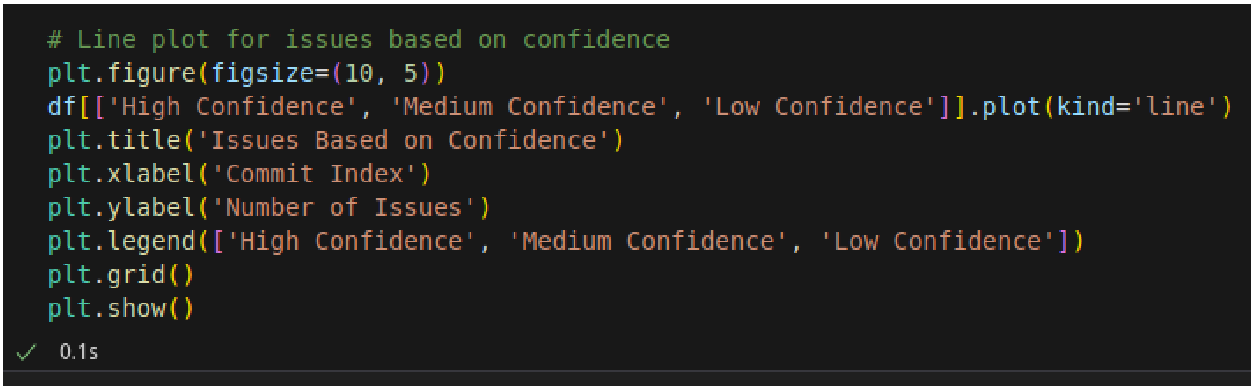
- Plots based on severity.



```
# Line plot for issues based on severity
plt.figure(figsize=(10, 5))
df[['High Severity', 'Medium Severity', 'Low Severity']].plot(kind='line')
plt.title('Issues Based on Severity')
plt.xlabel('Commit Index')
plt.ylabel('Number of Issues')
plt.legend(['High Severity', 'Medium Severity', 'Low Severity'])
plt.grid()
plt.show()
```

✓ 0.1s

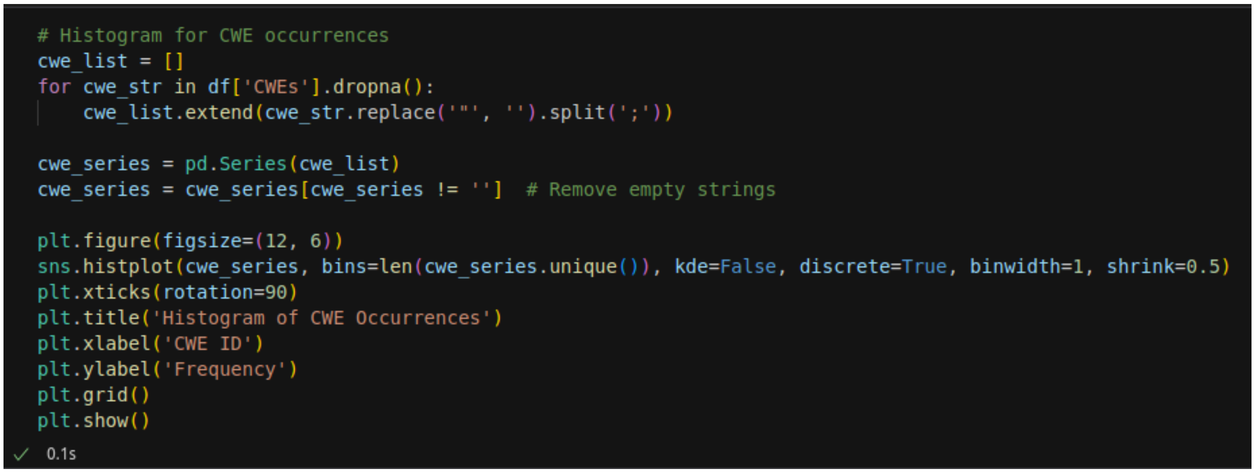
- Plots based on confidence.



```python
# Line plot for issues based on confidence
plt.figure(figsize=(10, 5))
df[['High Confidence', 'Medium Confidence', 'Low Confidence']].plot(kind='line')
plt.title('Issues Based on Confidence')
plt.xlabel('Commit Index')
plt.ylabel('Number of Issues')
plt.legend(['High Confidence', 'Medium Confidence', 'Low Confidence'])
plt.grid()
plt.show()
```

✓ 0.1s

- CWE histogram



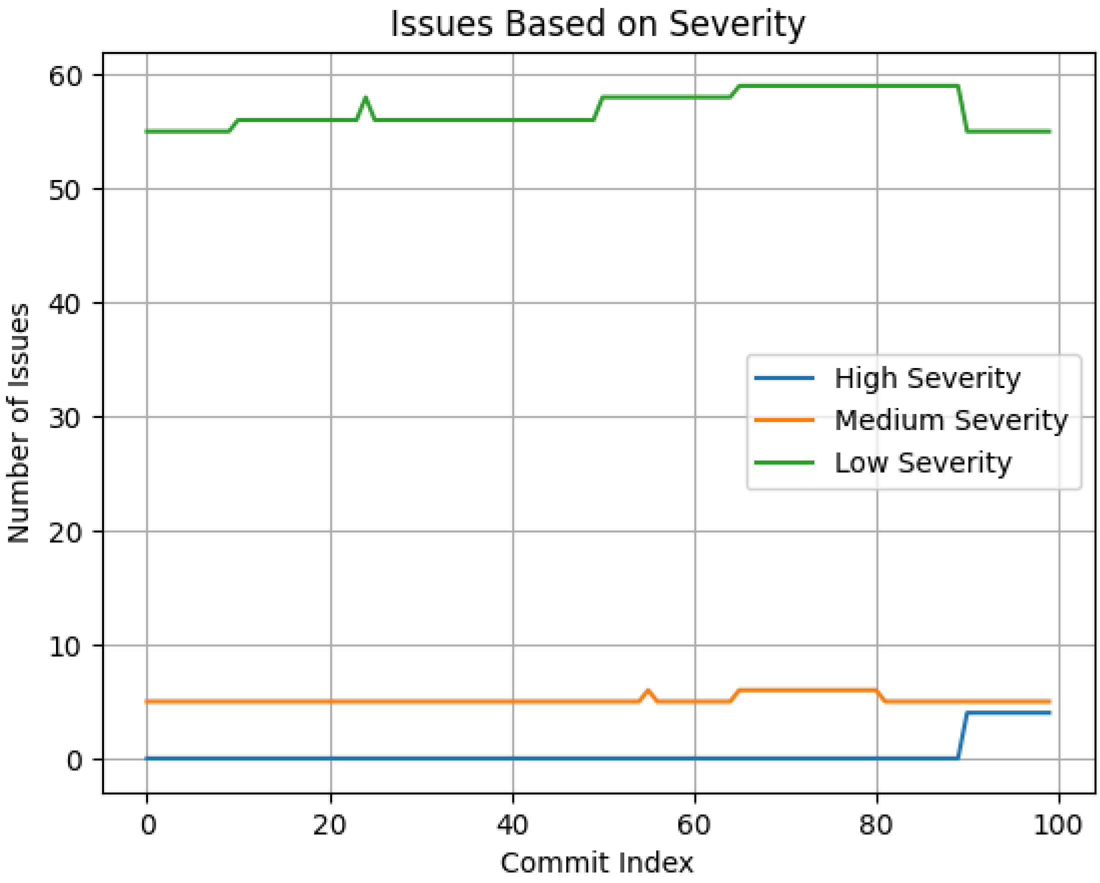
```python
# Histogram for CWE occurrences
cwe_list = []
for cwe_str in df['CWEs'].dropna():
    cwe_list.extend(cwe_str.replace('"', '').split(';'))

cwe_series = pd.Series(cwe_list)
cwe_series = cwe_series[cwe_series != '']  # Remove empty strings

plt.figure(figsize=(12, 6))
sns.histplot(cwe_series, bins=len(cwe_series.unique()), kde=False, discrete=True, binwidth=1, shrink=0.5)
plt.xticks(rotation=90)
plt.title('Histogram of CWE Occurrences')
plt.xlabel('CWE ID')
plt.ylabel('Frequency')
plt.grid()
plt.show()
```

✓ 0.1s

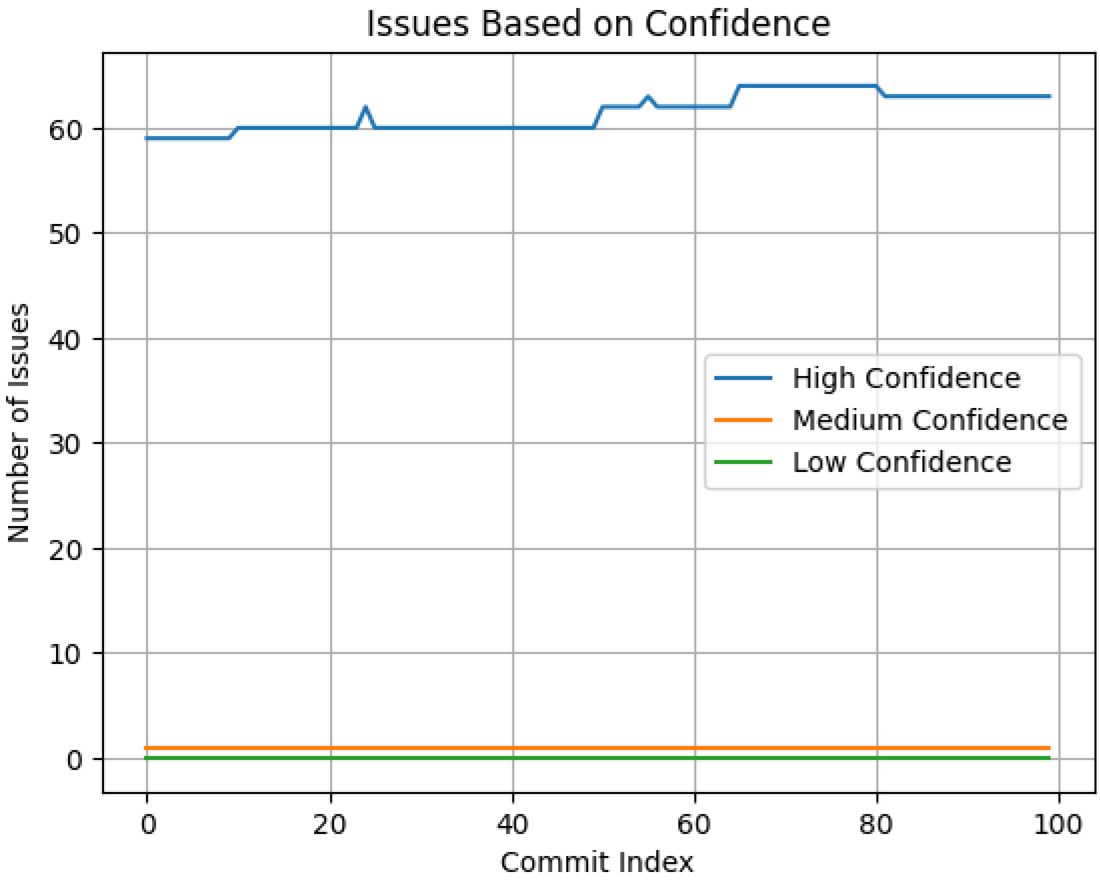
Results:

1. **mainm :**

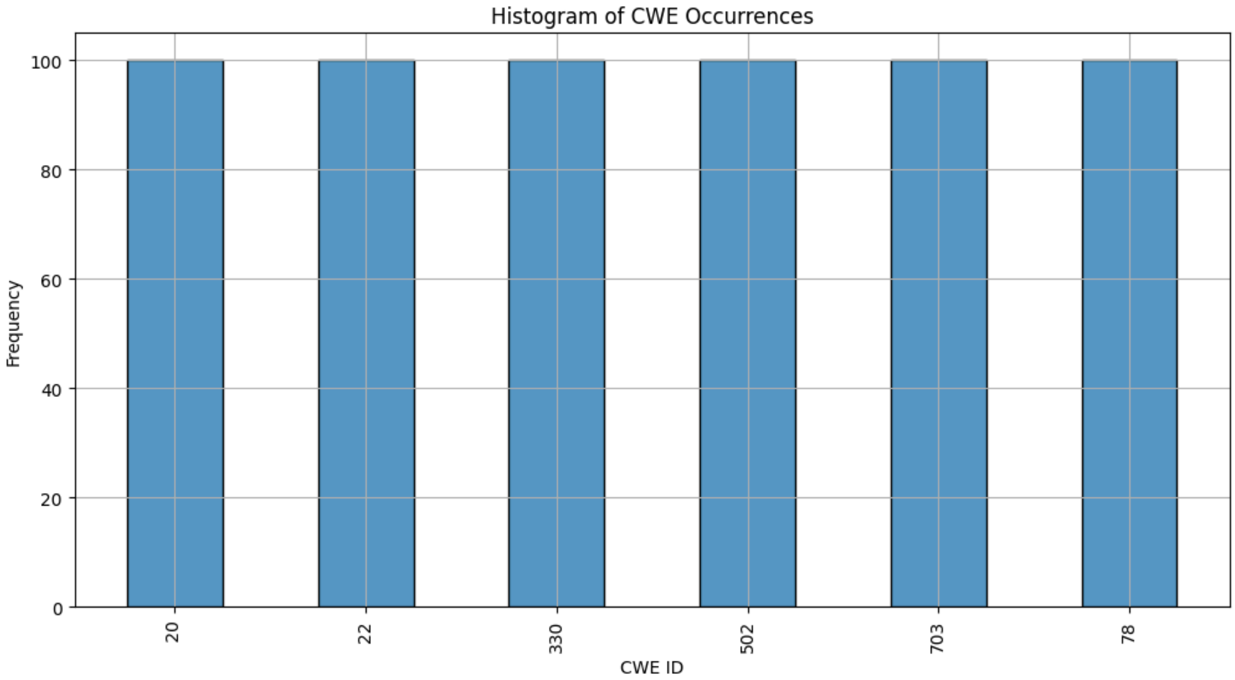
a. Issues based on severity:



b. Issues based on confidence:

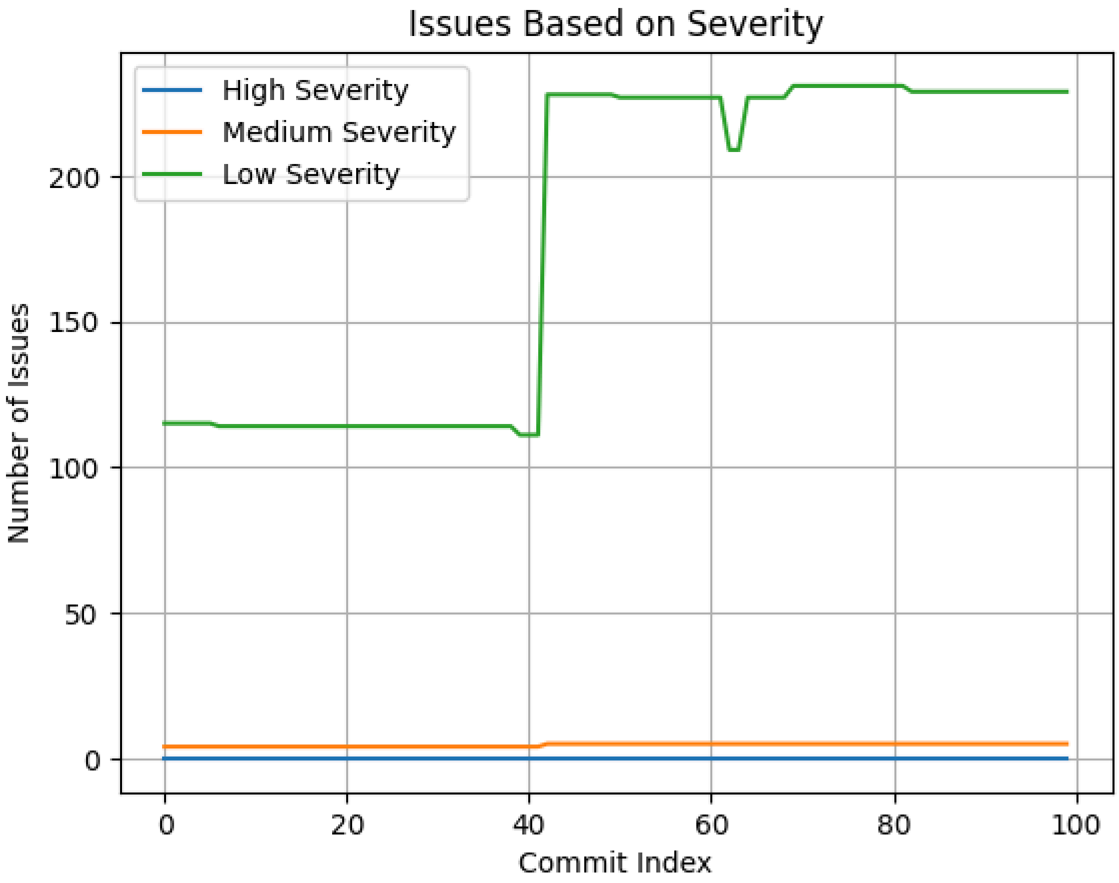


c. Histogram of CWEs:

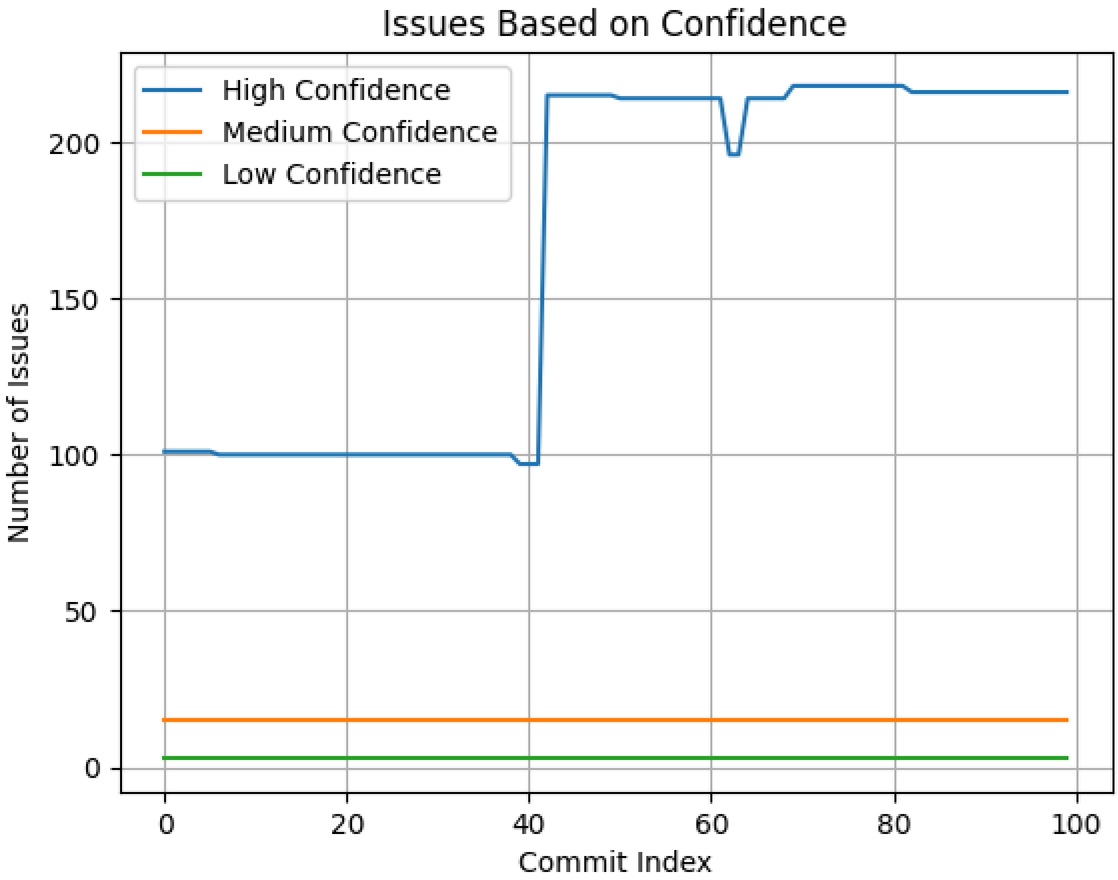


2. llama-cpp-python :

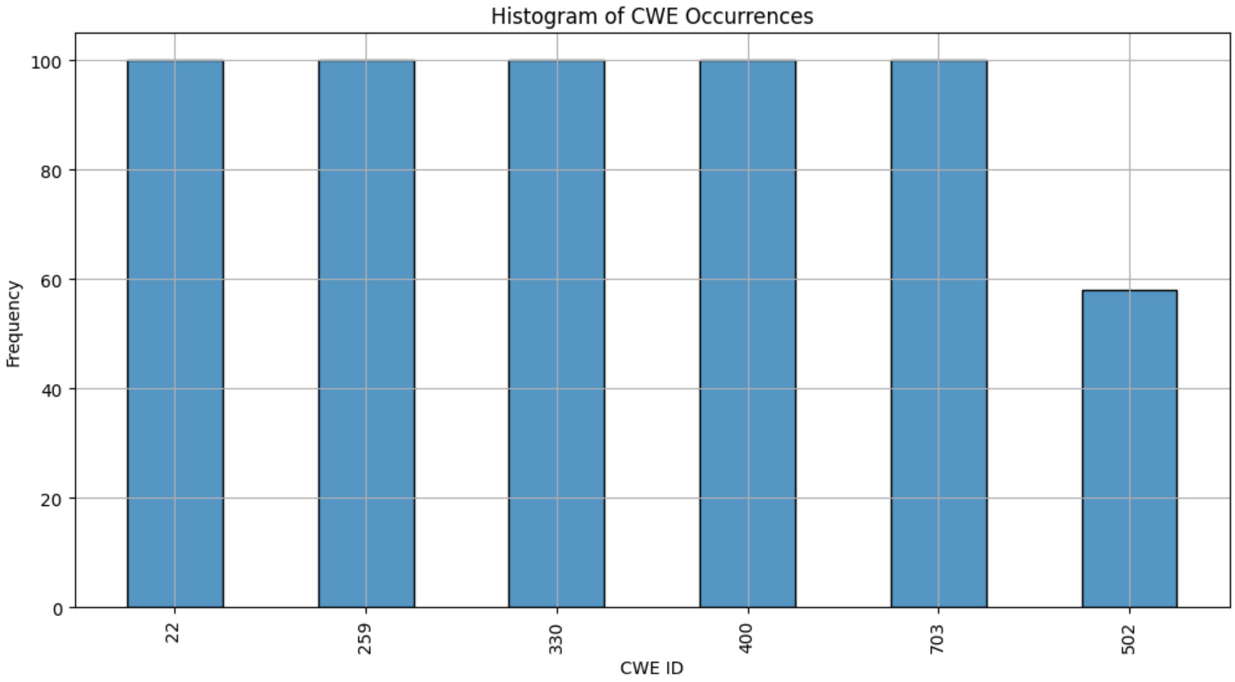
a. Issues based on severity:



b. Issues based on confidence:

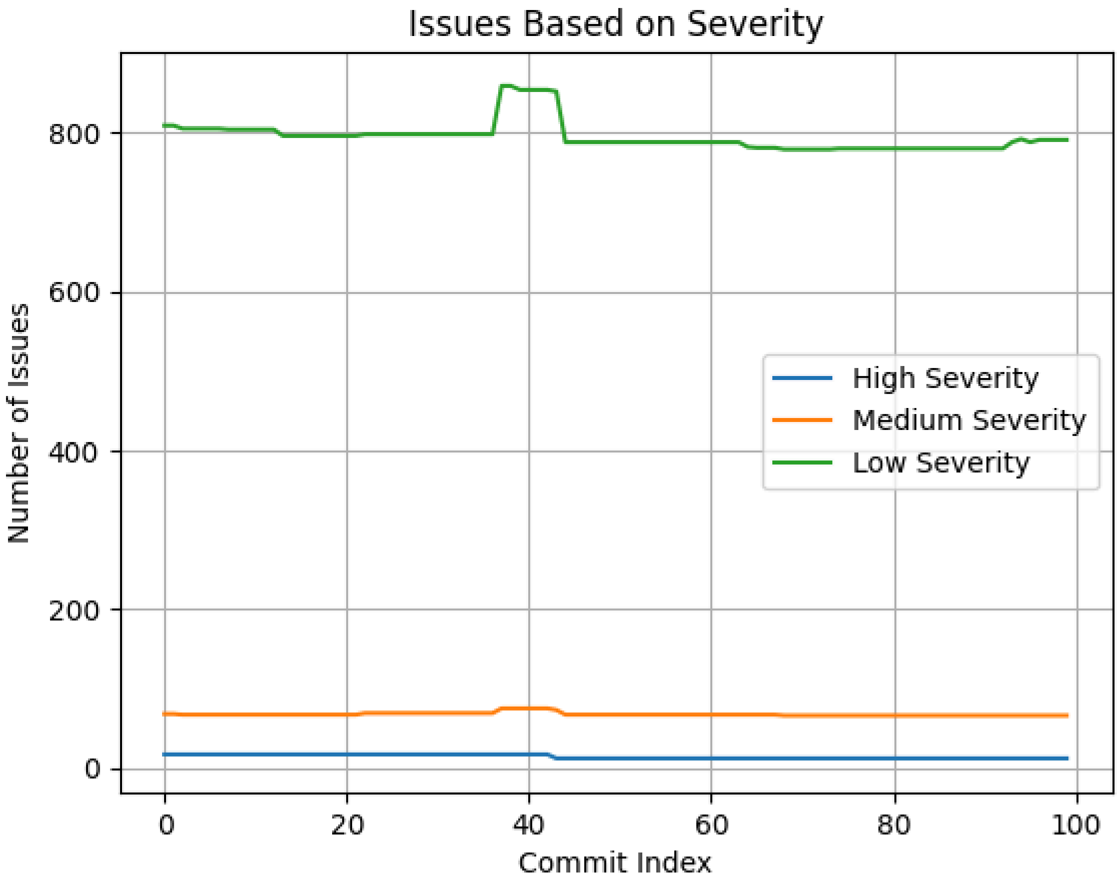


c. Histogram of CWEs:

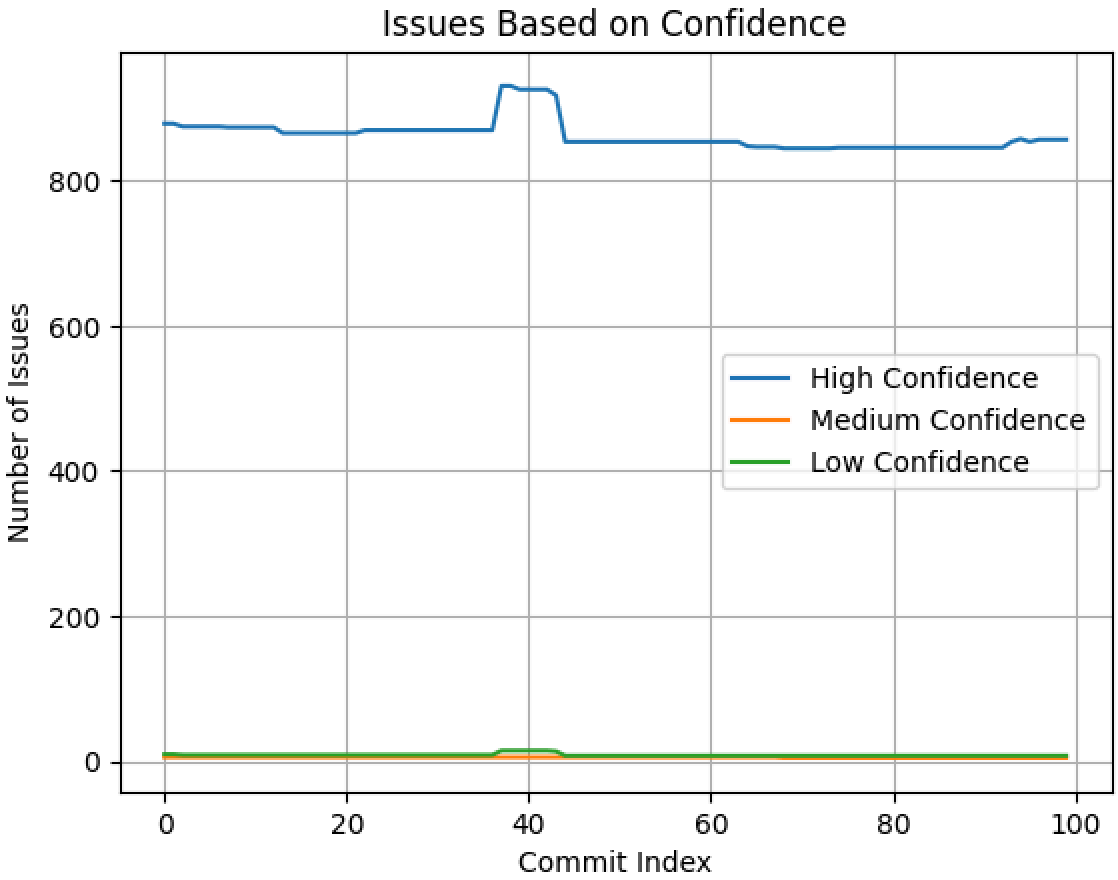


3. flower :

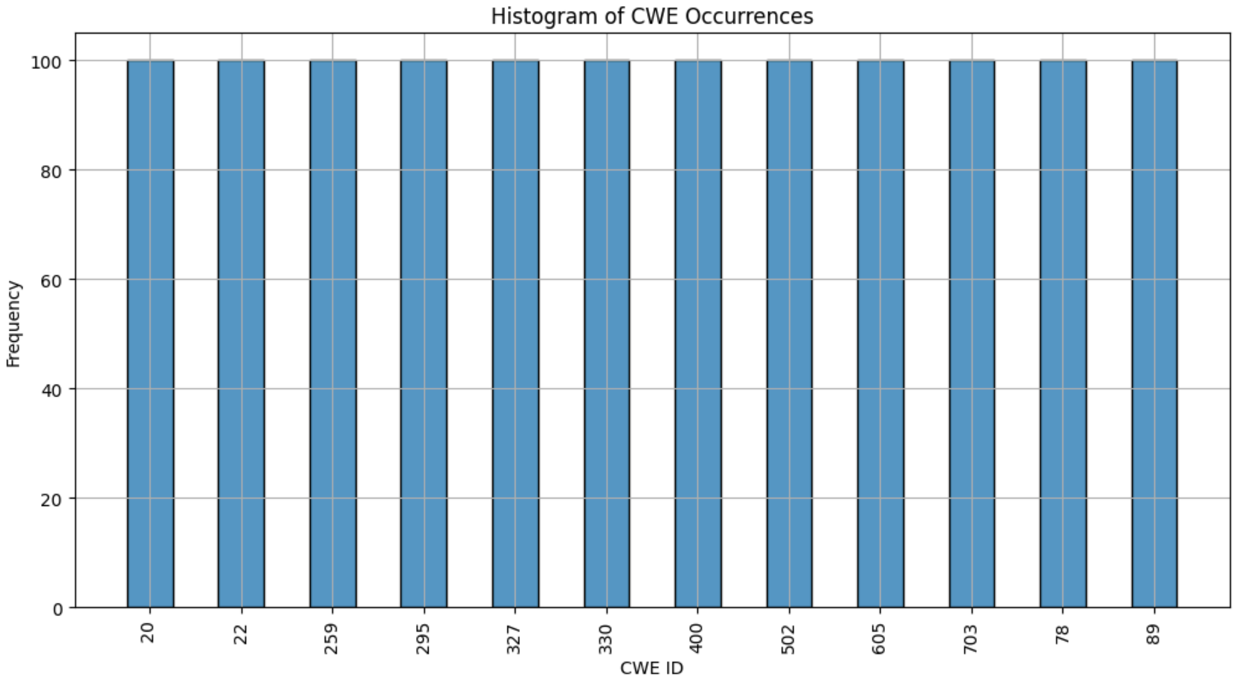
a. Issues based on severity:



b. Issues based on confidence:



c. Histogram of CWEs:



Research Questions:

1. When are vulnerabilities with high severity, introduced and fixed along the development timeline in OSS repositories?

Purpose:

In this research question, we analyze the timeline of high-severity vulnerabilities in open-source software (**OSS**) repositories. We look at when these vulnerabilities are introduced and when they are subsequently fixed. The goal is to understand how long they persist before remediation.

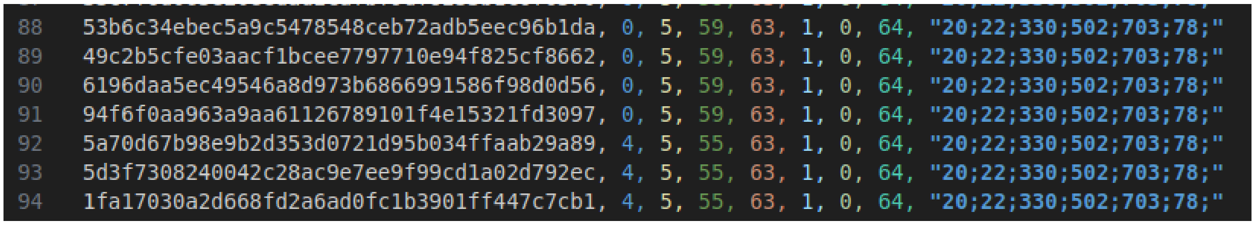
Approach:

Step1: We look at the graph of issues based on severity to figure out when the number of high severity issues increased/decreased.

Step 2: We look at the csv file used to generate the plots to figure out the exact commit in which high severity issues were introduced/fixed.

Results:

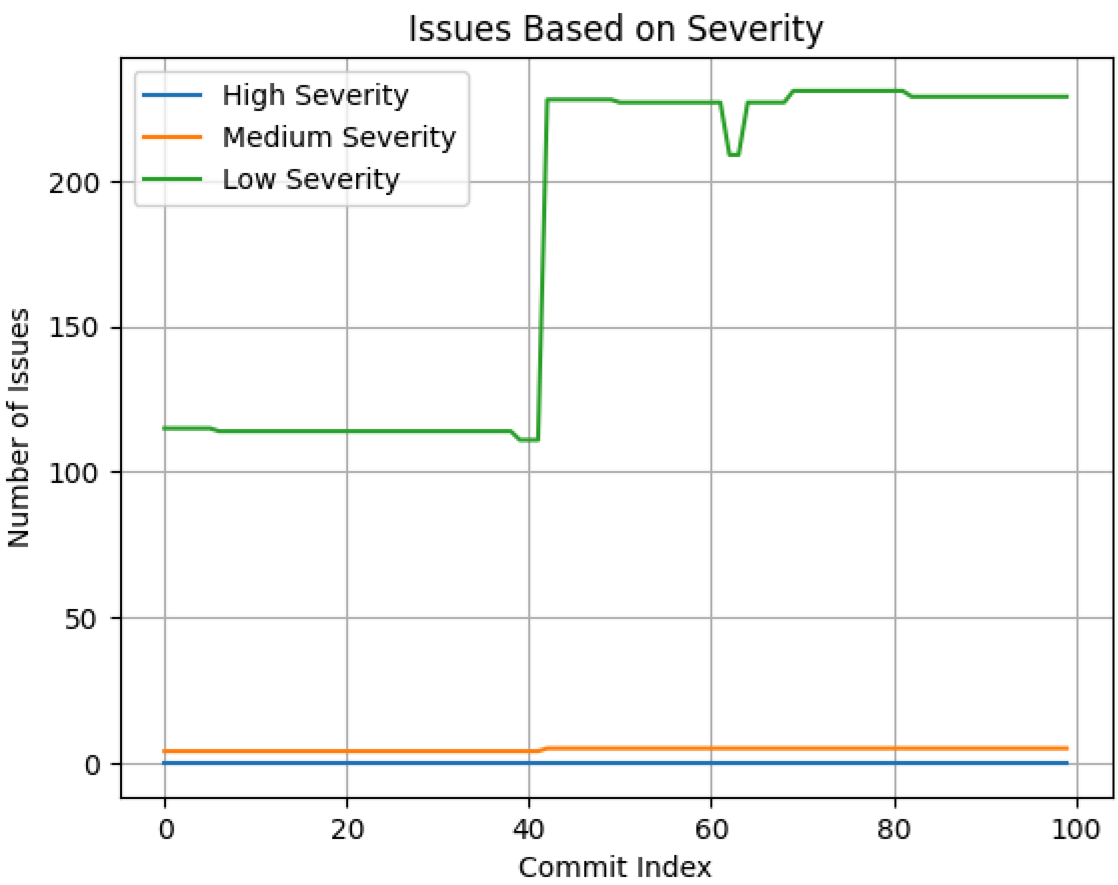
1. manim
    - High severity issues were fixed about 90 commits ago.



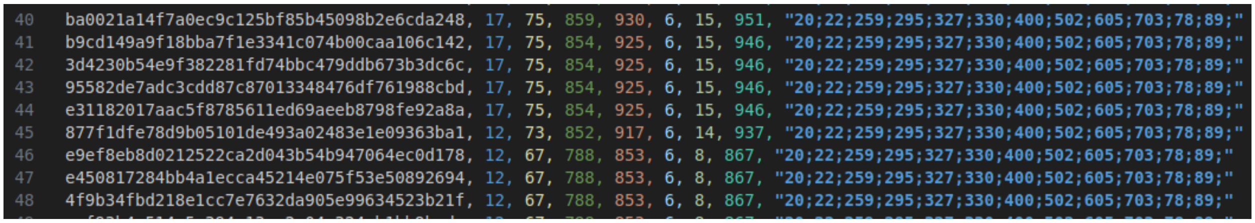
```
88   53b6c34ebec5a9c5478548ceb72adb5eec96b1da, 0, 5, 59, 63, 1, 0, 64, "20;22;330;502;703;78;"
89   49c2b5cfe03aacf1bcee7797710e94f825cf8662, 0, 5, 59, 63, 1, 0, 64, "20;22;330;502;703;78;"
90   6196daa5ec49546a8d973b6866991586f98d0d56, 0, 5, 59, 63, 1, 0, 64, "20;22;330;502;703;78;"
91   94f6f0aa963a9aa61126789101f4e15321fd3097, 0, 5, 59, 63, 1, 0, 64, "20;22;330;502;703;78;"
92   5a70d67b98e9b2d353d0721d95b034ffaab29a89, 4, 5, 55, 63, 1, 0, 64, "20;22;330;502;703;78;"
93   5d3f7308240042c28ac9e7ee9f99cd1a02d792ec, 4, 5, 55, 63, 1, 0, 64, "20;22;330;502;703;78;"
94   1fa17030a2d668fd2a6ad0fc1b3901ff447c7cb1, 4, 5, 55, 63, 1, 0, 64, "20;22;330;502;703;78;"
```

    - High severity issues were introduced before the last 100 commits and there is no information about when they were introduced as we limit our analysis to 100 commits.
    - There were 4 high severity issues which were fixed 90 commits ago, and no high severity issues have been introduced since.

2. llama-cpp-python
    - In the past 100 commits there have been no high severity issues.
    - The graph of high severity issues is a flat line at 0.



3. <u>flower</u>
    - There were high severity issues close to about 40 commits ago.
    - Then there was a slight increase in the number of high severity issues.



```
40   ba0021a14f7a0ec9c125bf85b45098b2e6cda248, 17, 75, 859, 930, 6, 15, 951, "20;22;259;295;327;330;400;502;605;703;78;89;"
41   b9cd149a9f18bba7f1e3341c074b00caa106c142, 17, 75, 854, 925, 6, 15, 946, "20;22;259;295;327;330;400;502;605;703;78;89;"
42   3d4230b54e9f382281fd74bbc479ddb673b3dc6c, 17, 75, 854, 925, 6, 15, 946, "20;22;259;295;327;330;400;502;605;703;78;89;"
43   95582de7adc3cdd87c87013348476df761988cbd, 17, 75, 854, 925, 6, 15, 946, "20;22;259;295;327;330;400;502;605;703;78;89;"
44   e31182017aac5f8785611ed69aeeb8798fe92a8a, 17, 75, 854, 925, 6, 15, 946, "20;22;259;295;327;330;400;502;605;703;78;89;"
45   877f1dfe78d9b05101de493a02483e1e09363ba1, 12, 73, 852, 917, 6, 14, 937, "20;22;259;295;327;330;400;502;605;703;78;89;"
46   e9ef8eb8d0212522ca2d043b54b947064ec0d178, 12, 67, 788, 853, 6, 8, 867, "20;22;259;295;327;330;400;502;605;703;78;89;"
47   e450817284bb4a1ecca45214e075f53e50892694, 12, 67, 788, 853, 6, 8, 867, "20;22;259;295;327;330;400;502;605;703;78;89;"
48   4f9b34fbd218e1cc7e7632da905e99634523b21f, 12, 67, 788, 853, 6, 8, 867, "20;22;259;295;327;330;400;502;605;703;78;89;"
```

- There were 12 high severity issues till 44 commits ago.
- In the last 43rd commit, 5 new high severity issues were introduced.
- These newly introduced issues along with the previous issues have not been solved yet.

2. Do vulnerabilities of different severity have the same pattern of introduction and elimination?

Purpose:

In this research question, we compare the lifecycle of vulnerabilities across different severity levels. We examine whether low, medium, and high-severity vulnerabilities follow similar patterns regarding their introduction and resolution. The analysis helps determine if high-severity vulnerabilities persist longer, require more effort to fix, or are concentrated in particular development phases compared to lower-severity vulnerabilities.

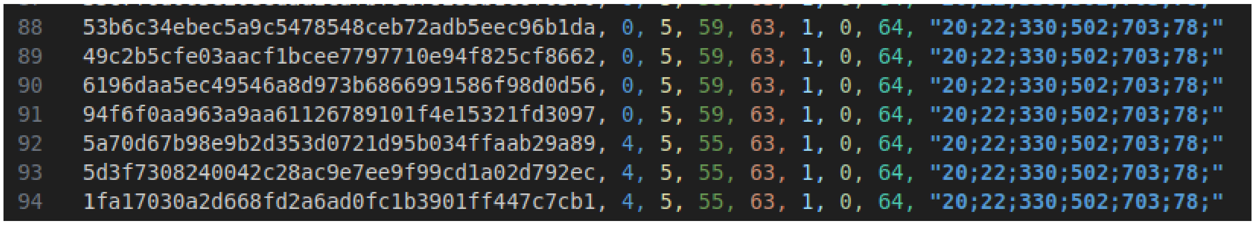
Approach:

Step1: We look at the graph of issues based on severity to figure out if there are any patterns across different severity levels.

Step 2: We look at the csv file used to generate the plots to figure out the exact commit.

Results:

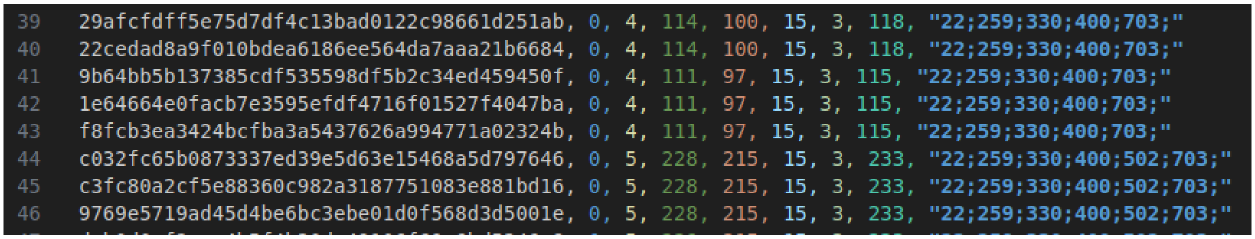
1. manim
   - There was a fall in high severity issues and a rise in low severity issues close to 90 commits ago.



```
88   53b6c34ebec5a9c5478548ceb72adb5eec96b1da, 0, 5, 59, 63, 1, 0, 64, "20;22;330;502;703;78;"
89   49c2b5cfe03aacf1bcee7797710e94f825cf8662, 0, 5, 59, 63, 1, 0, 64, "20;22;330;502;703;78;"
90   6196daa5ec49546a8d973b6866991586f98d0d56, 0, 5, 59, 63, 1, 0, 64, "20;22;330;502;703;78;"
91   94f6f0aa963a9aa61126789101f4e15321fd3097, 0, 5, 59, 63, 1, 0, 64, "20;22;330;502;703;78;"
92   5a70d67b98e9b2d353d0721d95b034ffaab29a89, 4, 5, 55, 63, 1, 0, 64, "20;22;330;502;703;78;"
93   5d3f7308240042c28ac9e7ee9f99cd1a02d792ec, 4, 5, 55, 63, 1, 0, 64, "20;22;330;502;703;78;"
94   1fa17030a2d668fd2a6ad0fc1b3901ff447c7cb1, 4, 5, 55, 63, 1, 0, 64, "20;22;330;502;703;78;"
```

   - Between the last 90th commit and 91st commit, the number of **high severity** issues went down from **4 to 0**.
   - The number of **low severity** issues went up from **55 to 59**.
   - This could indicate that 4 high severity issues were slightly modified and converted into low severity issues, rather than completely solving them.

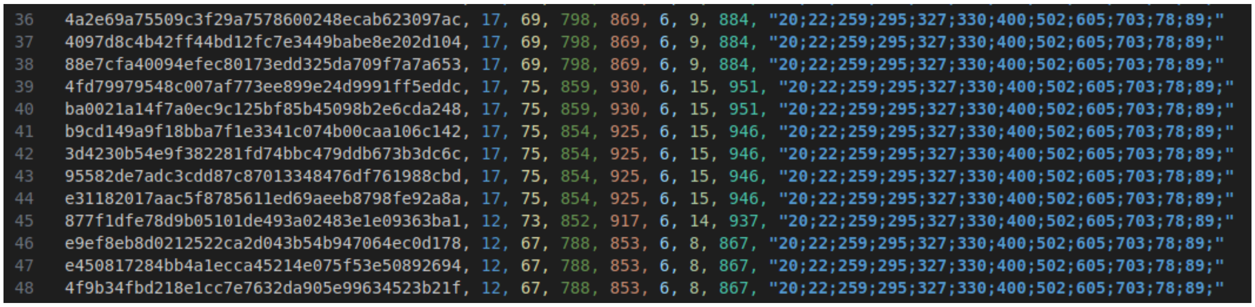
2. llama-cpp-python
    - The medium severity issues plot was almost entirely a straight line except for a little bump.
    - Around the same time there is a huge decrease in the number of low severity issues.



```
39   29afcfdff5e75d7df4c13bad0122c98661d251ab, 0, 4, 114, 100, 15, 3, 118, "22;259;330;400;703;"
40   22cedad8a9f010bdea6186ee564da7aaa21b6684, 0, 4, 114, 100, 15, 3, 118, "22;259;330;400;703;"
41   9b64bb5b137385cdf535598df5b2c34ed459450f, 0, 4, 111, 97, 15, 3, 115, "22;259;330;400;703;"
42   1e64664e0facb7e3595efdf4716f01527f4047ba, 0, 4, 111, 97, 15, 3, 115, "22;259;330;400;703;"
43   f8fcb3ea3424bcfba3a5437626a994771a02324b, 0, 4, 111, 97, 15, 3, 115, "22;259;330;400;703;"
44   c032fc65b0873337ed39e5d63e15468a5d797646, 0, 5, 228, 215, 15, 3, 233, "22;259;330;400;502;703;"
45   c3fc80a2cf5e88360c982a3187751083e881bd16, 0, 5, 228, 215, 15, 3, 233, "22;259;330;400;502;703;"
46   9769e5719ad45d4be6bc3ebe01d0f568d3d5001e, 0, 5, 228, 215, 15, 3, 233, "22;259;330;400;502;703;"
```

    - Around 43 commits ago the number of medium severity issues decreased from 5 to 4.
    - In the same commit the number of low severity issues drastically decreased from 228 to 111.
    - 502 was removed from the list of unique CWEs in the same commit.
    - This suggests that all the issues with CWE=502 were fixed in this commit.

3. flower
    - There is a bump spanning a few commits in low severity issues around 40 commits ago.
    - The same bump is resonated in medium severity issues with smaller amplitude.



```
36   4a2e69a75509c3f29a7578600248ecab623097ac, 17, 69, 798, 869, 6, 9, 884, "20;22;259;295;327;330;400;502;605;703;78;89;"
37   4097d8c4b42ff44bd12fc7e3449babe8e202d104, 17, 69, 798, 869, 6, 9, 884, "20;22;259;295;327;330;400;502;605;703;78;89;"
38   88e7cfa40094efec80173edd325da709f7a7a653, 17, 69, 798, 869, 6, 9, 884, "20;22;259;295;327;330;400;502;605;703;78;89;"
39   4fd79979548c007af773ee899e24d9991ff5eddc, 17, 75, 859, 930, 6, 15, 951, "20;22;259;295;327;330;400;502;605;703;78;89;"
40   ba0021a14f7a0ec9c125bf85b45098b2e6cda248, 17, 75, 859, 930, 6, 15, 951, "20;22;259;295;327;330;400;502;605;703;78;89;"
41   b9cd149a9f18bba7f1e3341c074b00caa106c142, 17, 75, 854, 925, 6, 15, 946, "20;22;259;295;327;330;400;502;605;703;78;89;"
42   3d4230b54e9f382281fd74bbc479ddb673b3dc6c, 17, 75, 854, 925, 6, 15, 946, "20;22;259;295;327;330;400;502;605;703;78;89;"
43   95582de7adc3cdd87c87013348476df761988cbd, 17, 75, 854, 925, 6, 15, 946, "20;22;259;295;327;330;400;502;605;703;78;89;"
44   e31182017aac5f8785611ed69aeeb8798fe92a8a, 17, 75, 854, 925, 6, 15, 946, "20;22;259;295;327;330;400;502;605;703;78;89;"
45   877f1dfe78d9b05101de493a02483e1e09363ba1, 12, 73, 852, 917, 6, 14, 937, "20;22;259;295;327;330;400;502;605;703;78;89;"
46   e9ef8eb8d0212522ca2d043b54b947064ec0d178, 12, 67, 788, 853, 6, 8, 867, "20;22;259;295;327;330;400;502;605;703;78;89;"
47   e450817284bb4a1ecca45214e075f53e50892694, 12, 67, 788, 853, 6, 8, 867, "20;22;259;295;327;330;400;502;605;703;78;89;"
48   4f9b34fbd218e1cc7e7632da905e99634523b21f, 12, 67, 788, 853, 6, 8, 867, "20;22;259;295;327;330;400;502;605;703;78;89;"
```

    - Medium severity issues went from 67 to 75 to 69.
    - Low severity issues went from 788 to 859 to 798.

3. Which CWEs are the most frequent across different OSS repositories?

Purpose:

In this research question, we identify the most common "**Common Weakness Enumeration (CWE)**" categories in OSS projects. By categorizing vulnerabilities based on CWEs, we gain insights into the most prevalent security weaknesses in OSS development. This information can help prioritize security measures, inform best practices, and guide automated vulnerability detection efforts.

Approach:

- Since all the CWEs in a given repository generally do not change over time let's look at the CWEs across different repositories
- Let's look at the list of CWEs that have appeared in each repository
    1. mainm: 20, 22, 330, 502, 703, 78
    2. llama-cpp-python: 22, 259, 330, 400, 703
    3. flower: 20, 22, 259, 295, 327, 330, 400, 502, 605, 703, 78, 89

Results:

CWE = **20** and **78** appeared in **manim** and **flower**
CWE = **22**, **330** and **703** appeared in **all** the repositories
CWE = **259** and **400** appeared in **manim** and **flower**
CWE = **502** appeared in **manim** and **flower**

Overall **3 CWEs** appeared in **all** repositories and **5 CWEs** appeared in **two** of the repositories.

Conclusion:

In this lab, we learnt about **static code analysis** and how to detect security flaws in Python projects, using **Bandit**. We identified various vulnerabilities and classified them based on severity and confidence levels. We also examined how vulnerabilities appear and get resolved over time in open-source repositories.
One key takeaway from this lab is that security issues are common in real-world projects, and automated tools like Bandit play a crucial role in identifying them. However, we also observed that not all flagged issues are equally critical, and **manual verification** is essential.
Overall, this lab enhanced our understanding of software security, vulnerability detection, and research reporting, which are valuable skills for secure software development.

Github Repository Link:

https://github.com/SumeetSawale/STT-A2