



---

*CS6474: Software Testing Laboratory  
(Spring 2023)*

---

**Bishwajit Prasad Gond**  
**222CS3113**

Master of Technology  
222cs3113@nitrkl.ac.in

**Department of Computer Science & Engineering  
NIT, Rourkela**

April 10, 2023

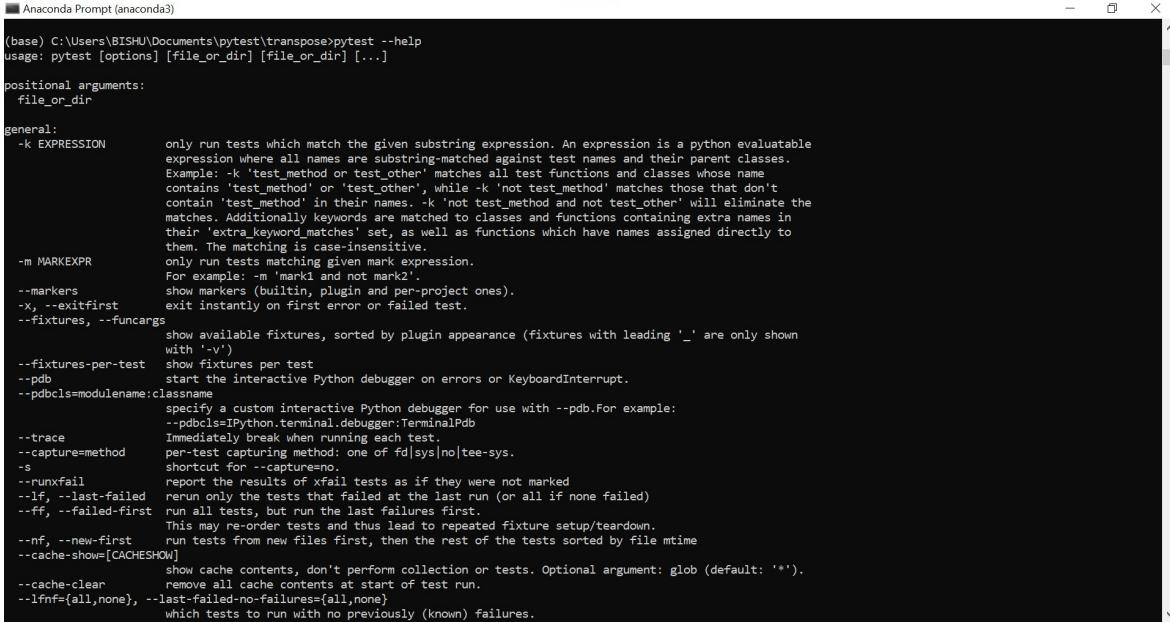
# Contents

<b>1 Pytest-Cov</b>	<b>1</b>
1.1 Write a program to generate a Factorial of numbers (where stack length should be at 3 (max) ). The numbers should be 5, 3, 8, and 15. . . . .	2
1.2 Write a program to generate Fibonacci numbers. . . . .	3
1.3 Write a program that performs sorting of a group of integer values using the quick sort technique. . . . .	4
1.4 Write a program that accepts elements of a matrix and displays its transpose. .	5
1.5 Write a program to add two matrices and display the sum matrix. . . . .	6
1.6 Write a program to Print Prime Numbers from 1 to 100 using Scanner Class and For Loop. . . . .	7
1.7 Write a program to generate a palindrome of numbers. . . . .	8
1.8 Write a program to find out the sum of two arrays. . . . .	9
1.9 Write a program to check whether the number is even or odd. . . . .	10
1.10 Write a program for binary to hexadecimal conversion. . . . .	11

# 1 Pytest-Cov

Pytest is a Python testing framework that originated from the PyPy project. It can be used to write various types of software tests, including unit tests, integration tests, end-to-end tests, and functional tests. Its features include parametrized testing, fixtures, and assert re-writing.

Pytest fixtures provide the contexts for tests by passing in parameter names in test cases; its parametrization eliminates duplicate code for testing multiple sets of input and output; and its rewritten assert statements provide detailed output for causes of failures.

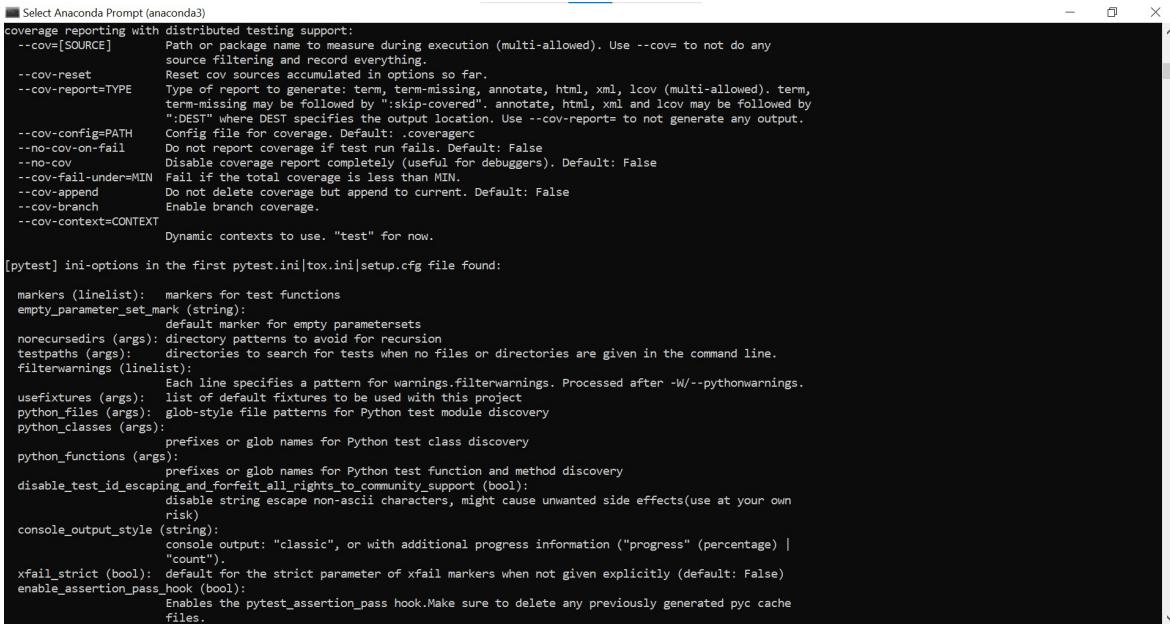


```
(base) C:\Users\BISHU\Documents\pytest\transpose>pytest --help
usage: pytest [options] [file_or_dir] [file_or_dir] [...]

positional arguments:
  file_or_dir

general:
  -k EXPRESSION          only run tests which match the given substring expression. An expression is a python evaluable expression where all names are substring-matched against test names and their parent classes.
    Example: -k 'test_method or test_other' matches all test functions and classes whose name contains 'test_method' or 'test_other', while -k 'not test_method' matches those that don't contain 'test_method' in their names. -k 'not test_method and not test_other' will eliminate the matches. Additionally keywords are matched to classes and functions containing extra names in their 'extra_keyword_matches' set, as well as functions which have names assigned directly to them. The matching is case-insensitive.
  -m MARKEXPR            only run tests matching given mark expression.
    For example: -m 'mark1 and not mark2'.
  --markers              show markers (builtin, plugin and per-project ones).
  -x, --exitfirst        exit instantly on first error or failed test.
  --fixtures, --funcargs show available fixtures, sorted by plugin appearance (fixtures with leading '_' are only shown with '-v')
  --fixtures-per-test   show fixtures per test
  --pdb                 start the interactive Python debugger on errors or KeyboardInterrupt.
  --pdbcls=modulename:classname
    specify a custom interactive Python debugger for use with --pdb. For example:
    --pdbcls=IPython.terminal.debugger.TerminalPdb
  --trace                Immediately break when running each test.
  --capture=method       per-test capturing method: one of fd|sys|no|tee-sys.
  -s                     shortcut for --capture=no.
  --runxfail             report the results of xfail tests as if they were not marked
  -iF, --last-failed    rerun only the tests that failed at the last run (or all if none failed)
  --ff, --failed-first  run all tests, but run the last failures first.
    This may re-order tests and thus lead to repeated fixture setup/teardown.
  --nf, --new-first     run tests from new files first, then the rest of the tests sorted by file mtime
  --cache-show=[CACHESHOW]
    show cache contents, don't perform collection or tests. Optional argument: glob (default: '*').
  --cache-clear          remove all cache contents at start of test run.
  --lfmf={all,none}, --last-failed-no-failures={all,none}
    which tests to run with no previously (known) failures.
```

Figure 1: Pytest Options



```
[Select Anaconda Prompt (anaconda3)]
coverage reporting with distributed testing support:
  --cov=[SOURCE]          Path or package name to measure during execution (multi-allowed). Use --cov= to not do any source filtering and record everything.
  --cov-reset              Reset cov sources accumulated in options so far.
  --cov-report=TYPE        Type of report to generate: term, term-missing, annotate, html, xml, lcov (multi-allowed). term, term-missing may be followed by ':skip-covered'. annotate, html, xml and lcov may be followed by ':DEST' where DEST specifies the output location. Use --cov-report= to not generate any output.
  --cov-config=PATH        Config file for coverage. .coveragearc
  --no-cov-on-fail         Do not report coverage if test run fails. Default: False
  --no-cov                Disable coverage report completely (useful for debuggers). Default: False
  --cov-fail-under=MIN    Fail if the total coverage is less than MIN.
  --cov-append             Do not delete coverage but append to current. Default: False
  --cov-branch             Enable branch coverage.
  --cov-context=CONTEXT   Dynamic contexts to use. "test" for now.

[pytest] ini-options in the first pytest.ini|tox.ini|setup.cfg file found:

  markers (linelist): markers for test functions
  empty_parameter_set_mark (string):
    default marker for empty parametersets
  norecursedirs (args): directory patterns to avoid for recursion
  testpaths (args): directories to search for tests when no files or directories are given in the command line.
  filterwarnings (linelist):
    Each line specifies a pattern for warnings.filterwarnings. Processed after -W--pythonwarnings.
  usefixtures (args): list of default fixtures to be used with this project
  python_files (args): glob-style file patterns for Python test module discovery
  python_classes (args):
    prefixes or glob names for Python test class discovery
  python_functions (args):
    prefixes or glob names for Python test function and method discovery
  disable_test_id_escaping_and_forfeit_all_rights_to_community_support (bool):
    disable string escape non-ascii characters, might cause unwanted side effects(use at your own risk)
  console_output_style (string):
    console output: "classic", or with additional progress information ("progress" (percentage) | "count")
  xfail_strict (bool): default for the strict parameter of xfail markers when not given explicitly (default: False)
  enable_assertion_pass_hook (bool):
    Enables the pytest_assertion_pass hook. Make sure to delete any previously generated pyc cache files.
```

Figure 2: Pytest Options

## Test the below programs in PyTest Tool

### 1.1 Write a program to generate a Factorial of numbers (where stack length should be at 3 (max) ). The numbers should be 5, 3, 8, and 15.

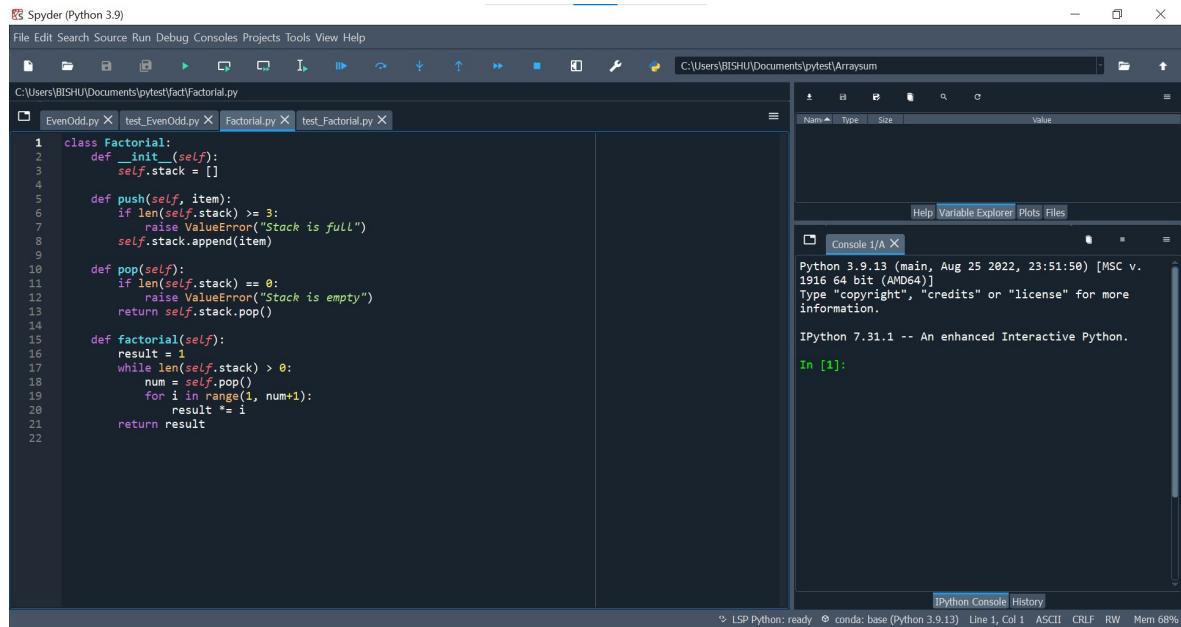


Figure 3: PyCOV Screenshot

A screenshot of the Anaconda Prompt window titled "Administrator: Anaconda Prompt (anaconda3)". It shows two sessions of PyTest execution and coverage analysis for Factorial.py and test\_Factorial.py.

```
(base) C:\Users\BISHU\Documents\pytest\fact>pytest
=====
test session starts =====
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\BISHU\Documents\pytest\fact
plugins: anyio-3.5.0, cov-4.0.0
collected 3 items

test_Factorial.py ... [100%]

=====
3 passed in 0.07s =====

(base) C:\Users\BISHU\Documents\pytest\fact>pytest --cov=C:\Users\BISHU\Documents\pytest\fact
=====
test session starts =====
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\BISHU\Documents\pytest\fact
plugins: anyio-3.5.0, cov-4.0.0
collected 3 items

test_Factorial.py ... [100%]

-----
coverage: platform win32, python 3.9.13-final-0 -----
Name      Stmts   Miss  Cover
-----
Factorial.py      18      0  100%
test_Factorial.py    21      0  100%
-----
TOTAL            39      0  100%

=====
3 passed in 0.15s =====

(base) C:\Users\BISHU\Documents\pytest\fact
```

Figure 4: Factorial Code

## 1.2 Write a program to generate Fibonacci numbers.

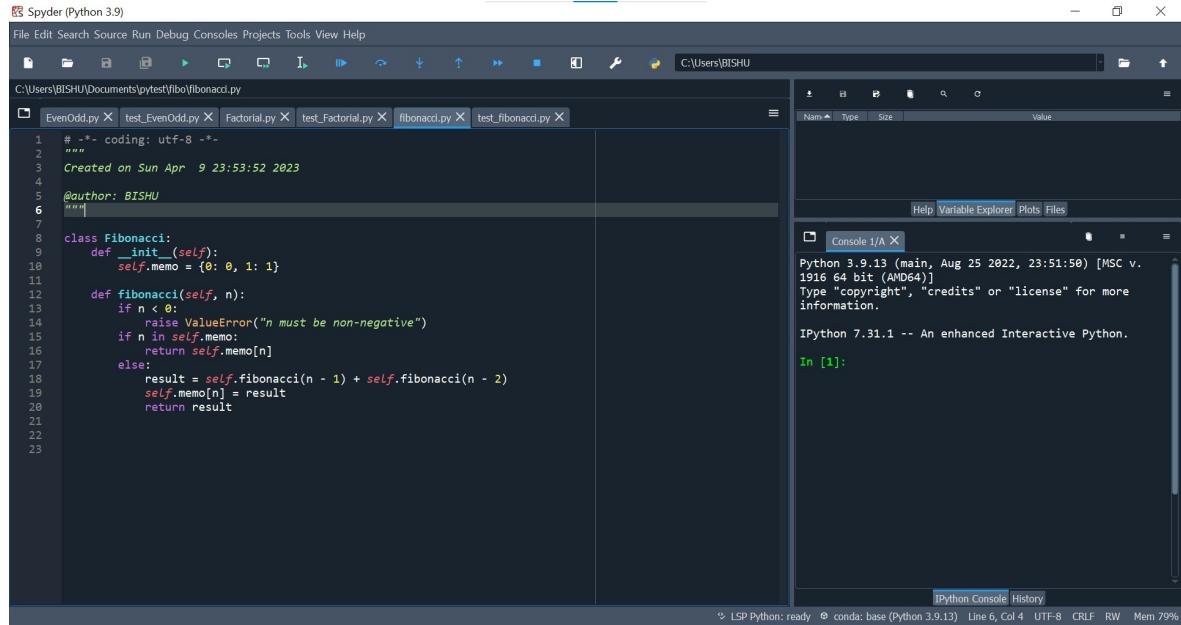


Figure 5: PyCOV Screenshot

A screenshot of the Anaconda Prompt window. It shows two separate sessions of the pytest command. The first session runs 'pytest' in the directory 'C:\Users\BISHU\Documents\pytest\fibo'. The output indicates a test session starts, platform details (win32, Python 3.9.13), root directory, and plugins (anyio-3.5.0, cov-4.0.0). It collected 1 item from 'test\_fibonacci.py' and passed in 0.13s. The second session runs 'pytest --cov=C:\Users\BISHU\Documents\pytest\fibo' to generate coverage reports. The coverage report shows the following data:

Name	Stmts	Miss	Cover
fibonacci.py	12	0	100%
test_fibonacci.py	20	0	100%
TOTAL	32	0	100%

Both sessions end with a message indicating 1 passed in 0.12s.

Figure 6: Fibonacci Code

### 1.3 Write a program that performs sorting of a group of integer values using the quick sort technique.

The screenshot shows the Spyder Python IDE interface. On the left, there are three tabs: 'quicksort.py' (active), 'test\_quicksort.py\*', and 'untitled0.py'. The 'quicksort.py' tab contains the following Python code:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 10 20:09:04 2023
@author: BISHU
"""

class Quicksort:
    def sort(self, arr):
        if len(arr) <= 1:
            return arr
        else:
            pivot = arr[0]
            less = [x for x in arr[1:] if x <= pivot]
            greater = [x for x in arr[1:] if x > pivot]
            return self.sort(less) + [pivot] + self.sort(greater)
```

The 'test\_quicksort.py\*' tab shows the following test code:

```
def test_quicksort():
    assert quicksort([3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]) == [1, 1, 2, 3, 3, 4, 5, 5, 6, 9]
```

The right side of the interface includes a 'Variable Explorer', 'Plots', and 'Files' panel. Below these is the 'Console I/A' tab, which displays the output of running the tests:

```
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v. 1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]:
```

At the bottom, status bars show 'LSP Python: ready', 'conda: base (Python 3.9.13)', and memory usage 'Line 8, Col 1 UTF-8 LF RW Mem 70%'. A progress bar at the bottom indicates 'Python Console History'.

Figure 7: PyCOV Screenshot

The screenshot shows the Anaconda Prompt window. It displays two sessions of pytest execution:

```
(base) C:\Users\BISHU\Documents\pytest\quicksort>pytest
=====
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\BISHU\Documents\pytest\quicksort
plugins: anyio-3.5.0, cov-4.0.0
collected 1 item

test_quicksort.py .

===== 1 passed in 0.07s =====
```

```
(base) C:\Users\BISHU\Documents\pytest\quicksort>pytest --cov=C:\Users\BISHU\Documents\pytest\quicksort\
=====
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\BISHU\Documents\pytest\quicksort
plugins: anyio-3.5.0, cov-4.0.0
collected 1 item

test_quicksort.py .

----- coverage: platform win32, python 3.9.13-final-0 -----
Name      Stmts  Miss  Cover
-----
quicksort.py      9      0   100%
test_quicksort.py  11      0   100%
-----
TOTAL          20      0   100%
```

Below the coverage report, another test run is shown:

```
===== 1 passed in 0.12s =====
```

```
(base) C:\Users\BISHU\Documents\pytest\quicksort>
```

Figure 8: Quicksort Code

## 1.4 Write a program that accepts elements of a matrix and displays its transpose.

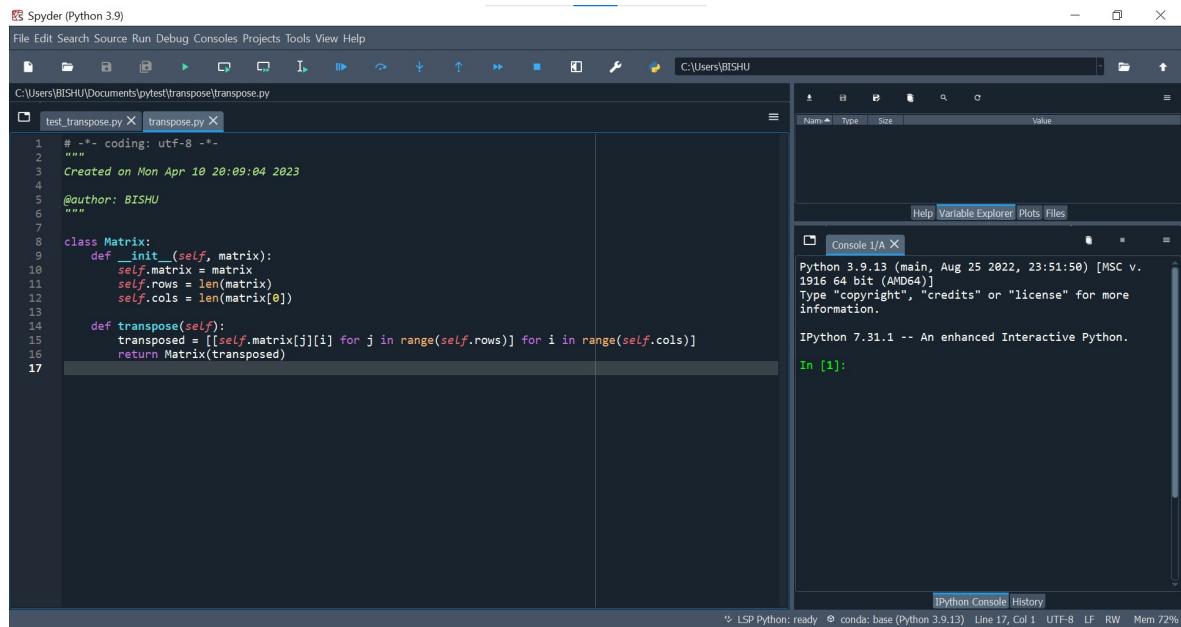
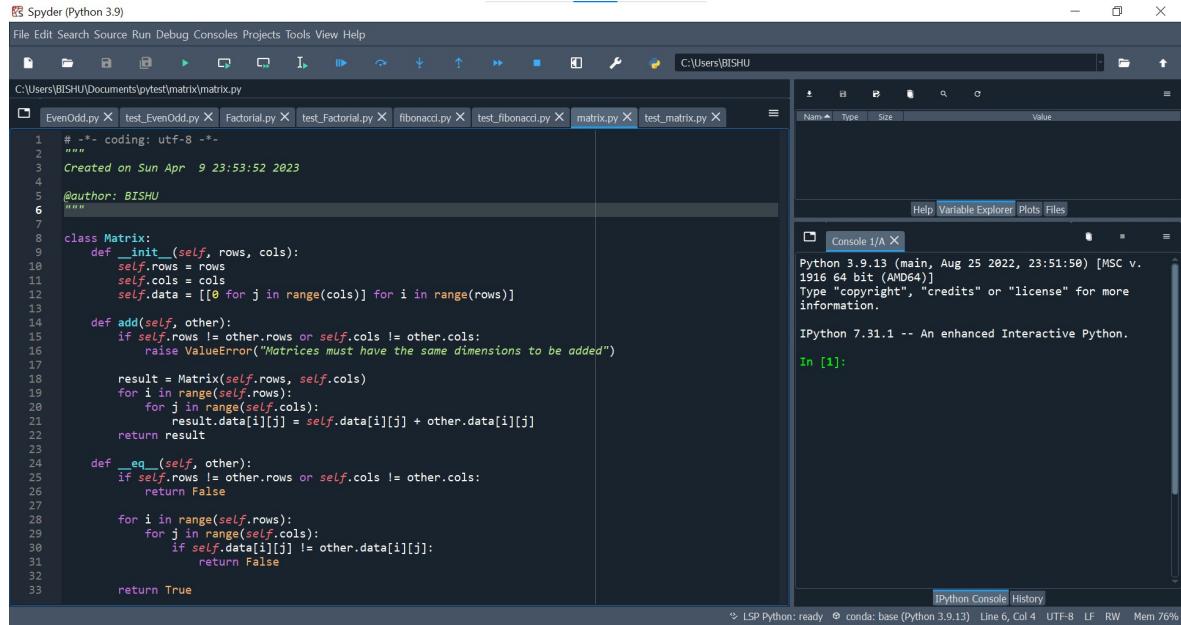


Figure 9: PyCOV Screenshot

A screenshot of the Anaconda Prompt window. It shows two sessions of pytest runs. The first session runs 'pytest' in the directory 'C:\Users\BISHU\Documents\pytest\transpose'. It collects 2 items and runs 'test\_transpose.py'. The second session runs 'pytest --cov=C:\Users\BISHU\Documents\pytest\transpose' to generate coverage reports. The coverage report for 'test\_transpose.py' shows 100% coverage with 15 statements, 0 misses, and 100% branch coverage. The total coverage for all files is 100% with 24 statements, 0 misses, and 100% branch coverage. Both sessions end with 2 passed tests.

Figure 10: Matrix Transpose Code

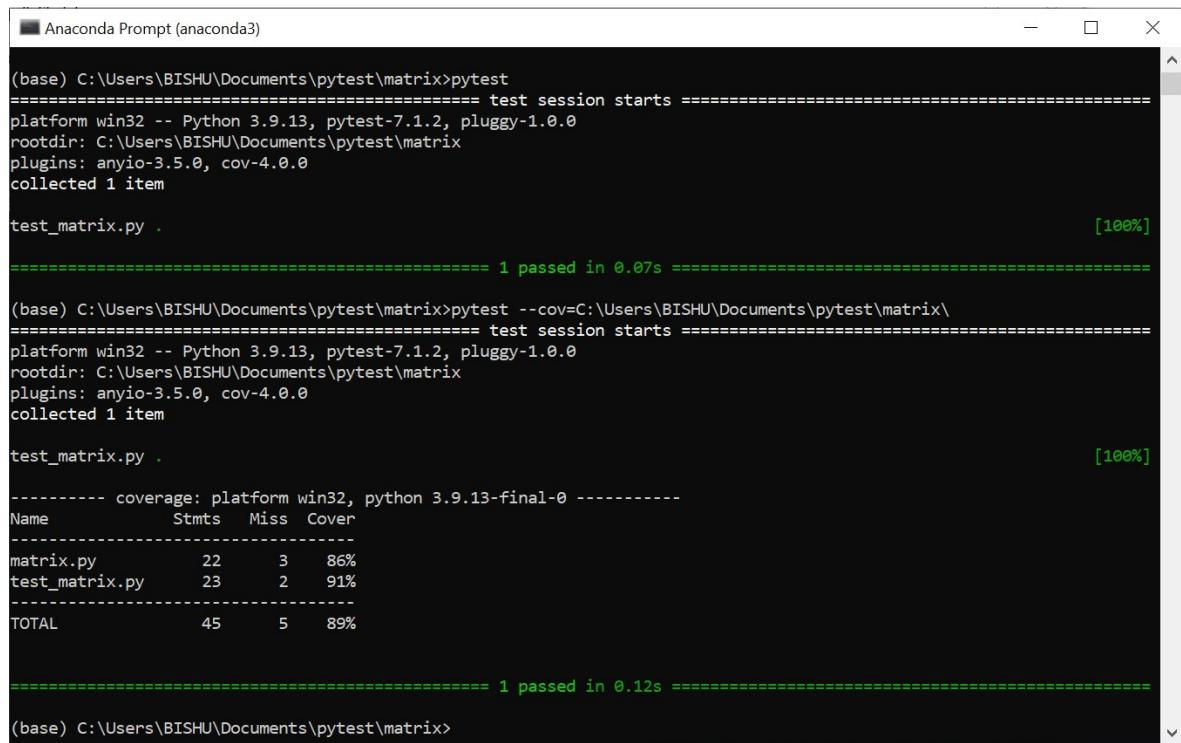
## 1.5 Write a program to add two matrices and display the sum matrix.



The screenshot shows the Spyder Python IDE interface. The left pane displays the source code for `matrix.py`, which defines a `Matrix` class with methods for initialization, addition, and equality comparison. The right pane shows the IPython console output, indicating a successful run of the test suite with 1 item collected and 1 passed in 0.07s. The status bar at the bottom right shows "LSP Python: ready" and "conda: base (Python 3.9.13) Line 6, Col 4 UTF-8 LF RW Mem 76%".

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun Apr  9 23:53:52 2023
4
5 @author: BISHU
6 """
7
8 class Matrix:
9     def __init__(self, rows, cols):
10         self.rows = rows
11         self.cols = cols
12         self.data = [[0 for j in range(cols)] for i in range(rows)]
13
14     def add(self, other):
15         if self.rows != other.rows or self.cols != other.cols:
16             raise ValueError("Matrices must have the same dimensions to be added")
17
18         result = Matrix(self.rows, self.cols)
19         for i in range(self.rows):
20             for j in range(self.cols):
21                 result.data[i][j] = self.data[i][j] + other.data[i][j]
22
23         return result
24
25     def __eq__(self, other):
26         if self.rows != other.rows or self.cols != other.cols:
27             return False
28
29         for i in range(self.rows):
30             for j in range(self.cols):
31                 if self.data[i][j] != other.data[i][j]:
32                     return False
33
34         return True
```

Figure 11: PyCOV Screenshot



The screenshot shows the Anaconda Prompt window. It displays the results of running `pytest` and `coverage` on the `matrix` module. The `pytest` session shows 1 item collected and 1 passed in 0.07s. The `coverage` session shows 89% coverage with 22 statements in `matrix.py` and 23 in `test_matrix.py`. The total coverage is 89%.

```
(base) C:\Users\BISHU\Documents\pytest\matrix>pytest
=====
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\BISHU\Documents\pytest\matrix
plugins: anyio-3.5.0, cov-4.0.0
collected 1 item

test_matrix.py . [100%]

===== 1 passed in 0.07s =====

(base) C:\Users\BISHU\Documents\pytest\matrix>pytest --cov=C:\Users\BISHU\Documents\pytest\matrix
=====
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\BISHU\Documents\pytest\matrix
plugins: anyio-3.5.0, cov-4.0.0
collected 1 item

test_matrix.py . [100%]

----- coverage: platform win32, python 3.9.13-final-0 -----
Name      Stmts  Miss  Cover
matrix.py    22      3    86%
test_matrix.py  23      2    91%
-----
TOTAL       45      5    89%
```

Figure 12: Matrix Addition Code

## 1.6 Write a program to Print Prime Numbers from 1 to 100 using Scanner Class and For Loop.

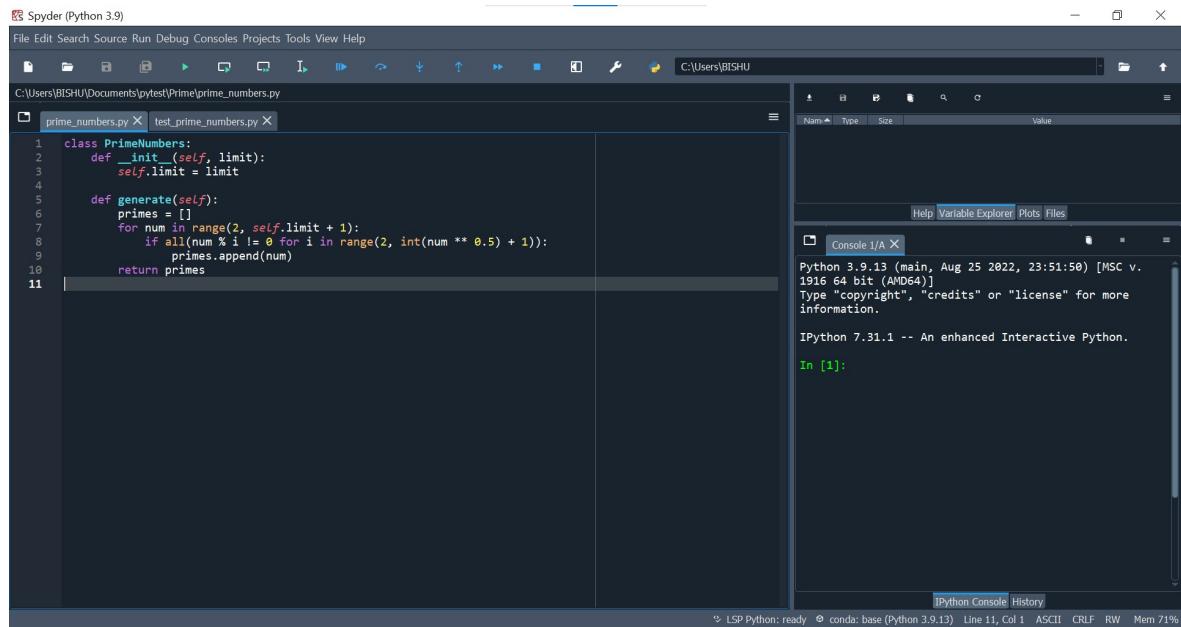


Figure 13: PyCOV Screenshot

A screenshot of the Anaconda Prompt window titled '(base) C:\Users\BISHU\Documents\pytest\Prime>'. It shows two separate pytest sessions. The first session runs 'prime\_numbers.py' and the second runs 'test\_prime\_numbers.py'. Both sessions show 4 passed tests in 0.06s and 0.14s respectively. After the tests, a coverage report is generated for 'prime\_numbers.py' and 'test\_prime\_numbers.py', showing 100% coverage for all statements and functions. The prompt ends with '(base) C:\Users\BISHU\Documents\pytest\Prime>'.

Figure 14: Prime Code

## 1.7 Write a program to generate a palindrome of numbers.

The screenshot shows the Spyder Python IDE interface. The left pane displays two files: `palindrome.py` and `test_palindrome.py`. The `palindrome.py` file contains the following code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun Apr  9 23:53:52 2023
4
5 @author: BISHU
6 """
7
8 class PalindromeGenerator:
9     def __init__(self, length):
10         self.length = length
11
12     def generate_palindrome(self):
13         if self.length <= 0:
14             return []
15
16         nums = list(range(1, self.length+1))
17         palindrome = nums + nums[-2::-1] if self.length > 1 else nums
18
19         return palindrome
```

The right pane shows the IPython console output:

```
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v. 1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]:
```

At the bottom of the interface, status information is displayed: LSP Python: ready, conda: base (Python 3.9.13), Line 19, Col 26, UTF-8, LF, RW, Mem 71%.

Figure 15: PyCOV Screenshot

The screenshot shows the Anaconda Prompt window. It displays the results of running pytest and coverage on the `palindrome.py` and `test_palindrome.py` files.

```
(base) C:\Users\BISHU\Documents\pytest\Palindrome>pytest
=====
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\BISHU\Documents\pytest\Palindrome
plugins: anyio-3.5.0, cov-4.0.0
collected 1 item

test_palindrome.py . [100%]

=====
1 passed in 0.43s =====

(base) C:\Users\BISHU\Documents\pytest\Palindrome>pytest --cov=C:\Users\BISHU\Documents\pytest\Palindrome
=====
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\BISHU\Documents\pytest\Palindrome
plugins: anyio-3.5.0, cov-4.0.0
collected 1 item

test_palindrome.py . [100%]

-----
coverage: platform win32, python 3.9.13-final-0 -----
Name           Stmts   Miss  Cover
-----
palindrome.py      10      0  100%
test_palindrome.py    14      1   93%
-----
TOTAL            24      1   96%

=====
1 passed in 0.74s =====

(base) C:\Users\BISHU\Documents\pytest\Palindrome>
```

Figure 16: Palindrome Code

## 1.8 Write a program to find out the sum of two arrays.

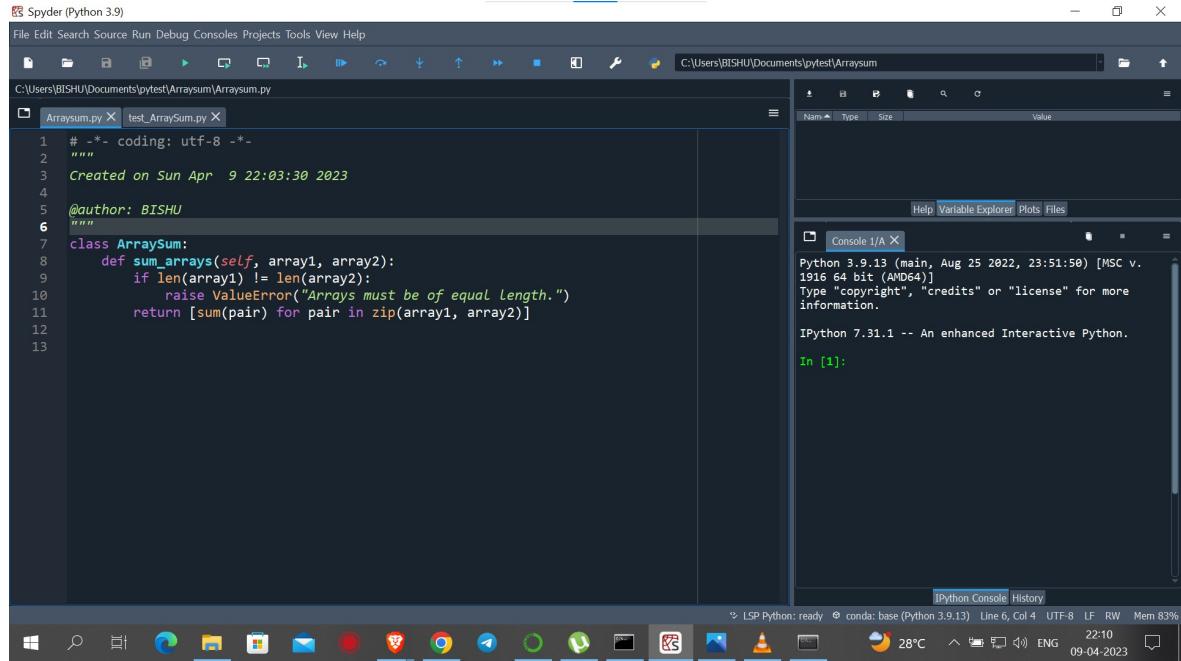


Figure 17: PyCOV Screenshot

A screenshot of the Anaconda Prompt window. It shows the following command-line session:

```
Administrator: Anaconda Prompt (anaconda3)
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\BISHU\Documents\pytest\Arraysum
plugins: anyio-3.5.0, cov-4.0.0
collected 1 item

test_ArraySum.py .

===== 1 passed in 0.07s =====

(base) C:\Users\BISHU\Documents\pytest\Arraysum>pytest --cov=C:\Users\BISHU\Documents\pytest\Arraysum\
===== test session starts =====
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\BISHU\Documents\pytest\Arraysum
plugins: anyio-3.5.0, cov-4.0.0
collected 1 item

test_ArraySum.py .

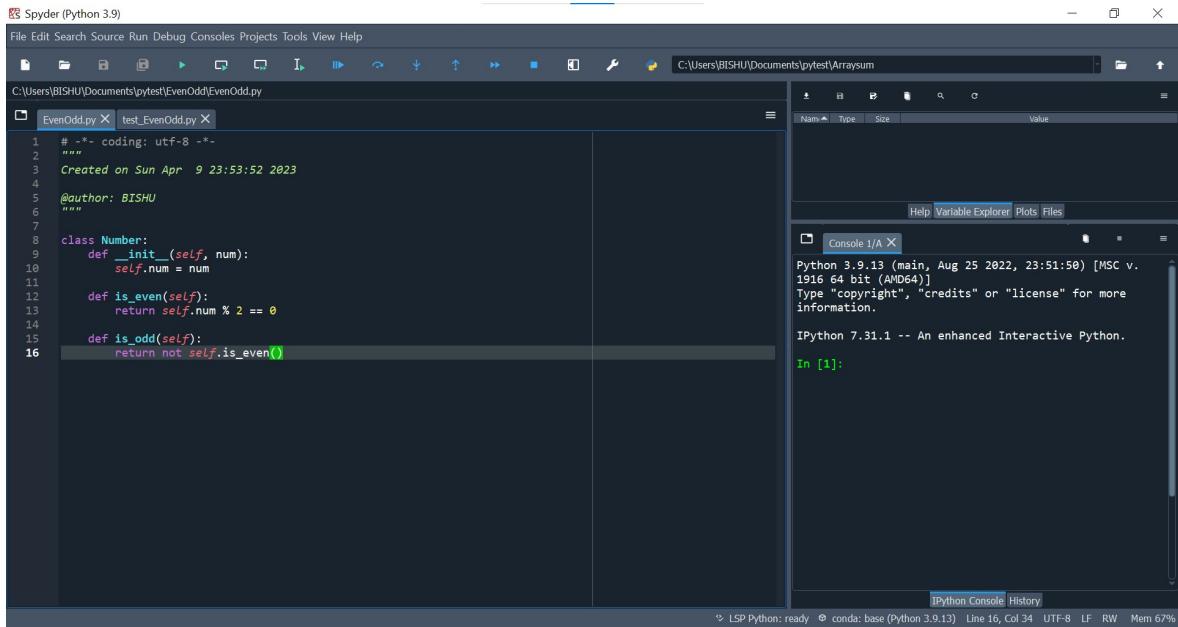
----- coverage: platform win32, python 3.9.13-final-0 -----
Name      Stmts  Miss  Cover
Arraysum.py       6      0   100%
test_ArraySum.py    14      2    86%
-----
TOTAL          20      2    90%

===== 1 passed in 0.14s =====

(base) C:\Users\BISHU\Documents\pytest\Arraysum>
```

Figure 18: Sum of Array Code

## 1.9 Write a program to check whether the number is even or odd.

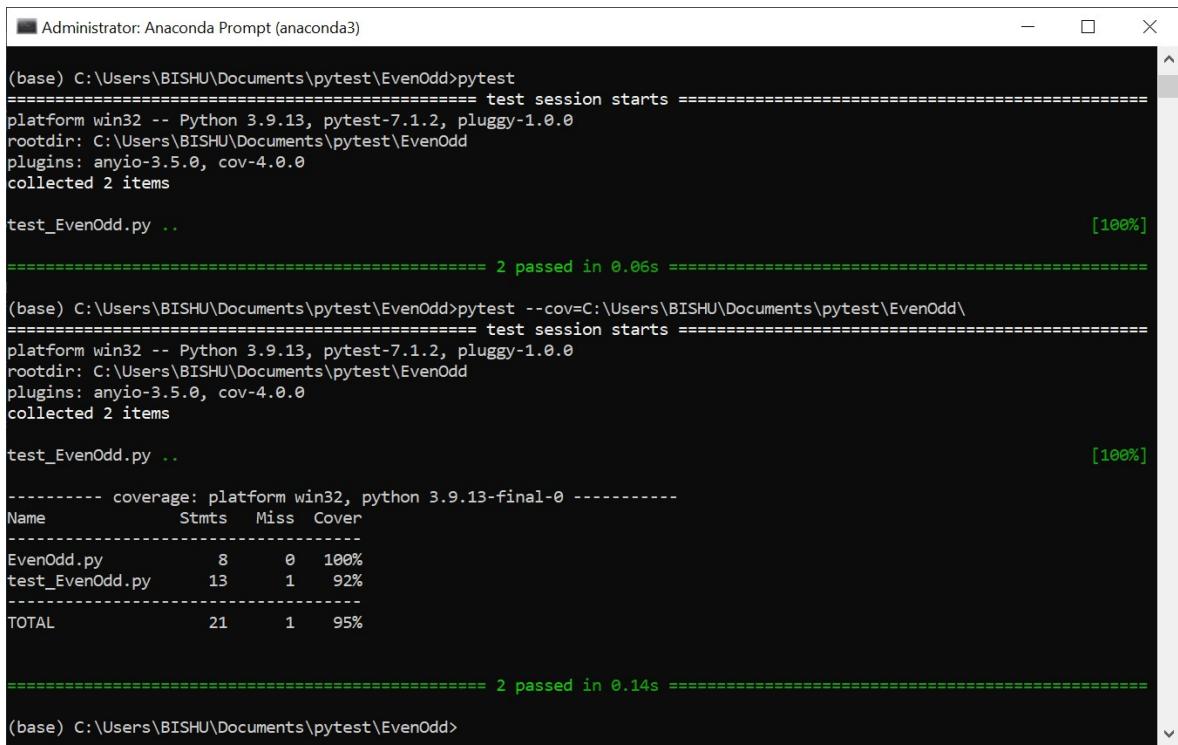


The screenshot shows the Spyder Python IDE interface. On the left, there are two tabs: 'EvenOdd.py' and 'test\_EvenOdd.py'. The 'EvenOdd.py' tab contains the following code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun Apr  9 23:53:52 2023
4
5 @author: BISHU
6 """
7
8 class Number:
9     def __init__(self, num):
10         self.num = num
11
12     def is_even(self):
13         return self.num % 2 == 0
14
15     def is_odd(self):
16         return not self.is_even()
```

The 'test\_EvenOdd.py' tab is currently active. On the right side of the interface, there is an IPython console window showing the Python version and license information.

Figure 19: PyCOV Screenshot



The screenshot shows an Anaconda Prompt window with the title 'Administrator: Anaconda Prompt (anaconda3)'. It displays the output of running pytest and coverage commands on the 'EvenOdd' project.

```
(base) C:\Users\BISHU\Documents\pytest\EvenOdd>pytest
=====
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\BISHU\Documents\pytest\EvenOdd
plugins: anyio-3.5.0, cov-4.0.0
collected 2 items

test_EvenOdd.py .. [100%]

=====
2 passed in 0.06s =====

(base) C:\Users\BISHU\Documents\pytest\EvenOdd>pytest --cov=C:\Users\BISHU\Documents\pytest\EvenOdd\
=====
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\BISHU\Documents\pytest\EvenOdd
plugins: anyio-3.5.0, cov-4.0.0
collected 2 items

test_EvenOdd.py .. [100%]

-----
coverage: platform win32, python 3.9.13-final-0 -----
Name      Stmts  Miss  Cover
-----
EvenOdd.py       8      0   100%
test_EvenOdd.py  13      1    92%
-----
TOTAL          21      1    95%

=====
2 passed in 0.14s =====

(base) C:\Users\BISHU\Documents\pytest\EvenOdd>
```

Figure 20: Odd or Even Code

## 1.10 Write a program for binary to hexadecimal conversion.

The screenshot shows the Spyder Python IDE interface. On the left, there are two tabs: 'binarytohex.py' and 'test\_binarytohex.py'. The 'binarytohex.py' tab contains the following code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun Apr  9 22:03:30 2023
4
5 @author: BISHU
6 """
7 class BinaryToHex:
8     def __init__(self, binary):
9         self.binary = binary
10
11    def convert(self):
12        decimal = int(self.binary, 2)
13        hexadecimal = hex(decimal)[2:].upper()
14        return hexadecimal
15
```

The 'test\_binarytohex.py' tab is currently active. On the right side of the interface, there is an IPython console window with the following output:

```
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.
1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]:
```

Figure 21: PyCOV Screenshot

The screenshot shows an Anaconda Prompt window titled 'Administrator: Anaconda Prompt (anaconda3)'. It displays the results of running pytest and coverage on the 'binarytohex.py' file.

```
(base) C:\Users\BISHU\Documents\pytest\binarytohex>pytest
=====
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\BISHU\Documents\pytest\binarytohex
plugins: anyio-3.5.0, cov-4.0.0
collected 1 item

test_binarytohex.py . [100%]

=====
1 passed in 0.05s =====

(base) C:\Users\BISHU\Documents\pytest\binarytohex>pytest --cov=C:\Users\BISHU\Documents\pytest\binarytohex\
=====
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\BISHU\Documents\pytest\binarytohex
plugins: anyio-3.5.0, cov-4.0.0
collected 1 item

test_binarytohex.py . [100%]

-----
coverage: platform win32, python 3.9.13-final-0 -----
Name          Stmts Miss Cover
-----
binarytohex.py      8     0  100%
test_binarytohex.py 12     0  100%
-----
TOTAL            20     0  100%

=====
1 passed in 0.18s =====

(base) C:\Users\BISHU\Documents\pytest\binarytohex>
```

Figure 22: Binary to Hexadecimal Code