

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Автоматики и Процессов Управления

ОТЧЕТ
по «Автоматизация тестирования» практике
Тема: Тестирование на основе PyTest

Студент гр. 2307

Руководитель

Подберёзский А. Д.

Турнецкая Е.Л.

Санкт-Петербург

2024

ЗАДАНИЕ
НА «АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ» ПРАКТИКУ

Студент Подберёзский А.Д.

Группа 2307

Тема практики: Тестирование на основе PyTest

Задание на практику:

Получение практических навыков по тестированию с помощью PyTest.

Для достижения поставленной цели требуется решить следующие задачи:

1. Установить программное окружение проекта по автоматизированному тестированию с фреймворком PyTest.
2. Реализовать тестовые функции для проведения модульного тестирования на основе программных инструментов PyTest.
3. Зафиксировать результаты тестирования в отчете.

Сроки прохождения практики: 02.09.2024 – 21.12.2024

Дата сдачи отчета: 24.11.2024

Дата защиты отчета: 24.11.2024

Студент		Подберёзский А.Д.
Руководитель		Турнецкая Е.Л.

АННОТАЦИЯ

В данном отчете рассмотрен процесс создания автоматизированного тестирования кода программ на языке программирования python с использованием инструмента PyTest. В рамках практической работы была разработана программа проверки палиндрома и протестирована правильность её исполнения и фиксация результатов тестирования. Цель работы — получить практические навыки работы с PyTest и применить их для решения задач автоматизации тестирования программ. Полученные результаты демонстрируют эффективность подхода, а также основные методы тестирования программ на python.

SUMMARY

This report examines the process of creating automated testing of program code in the python programming language using the PyTest tool. As part of the practical work, a palindrome verification program was developed and the correctness of its execution and fixation of test results were tested. The purpose of the work is to gain practical skills in working with PyTest and apply them to solve the tasks of automating software testing. The results obtained demonstrate the effectiveness of the approach, as well as the basic methods of testing python programs.

ВВЕДЕНИЕ

Получение практических навыков по тестированию с помощью PyTest.

Для достижения поставленной цели требуется решить следующие задачи:

1. Установить программное окружение проекта по автоматизированному тестированию с фреймворком PyTest.
2. Реализовать тестовые функции для проведения модульного тестирования на основе программных инструментов PyTest.
3. Зафиксировать результаты тестирования в отчете.

1. Установка программного окружения

Так как в ходе предыдущей практической работы по тестированию с помощью Selenium Webdriver и Python мы уже устанавливали Pycharm Community Edition, то пропускаем шаг установки IDE и сразу переходим к настройке нового проекта.

Создаём новый проект и проверяем чтобы в поле “Interpreter type” было выбрано “Project venv”.

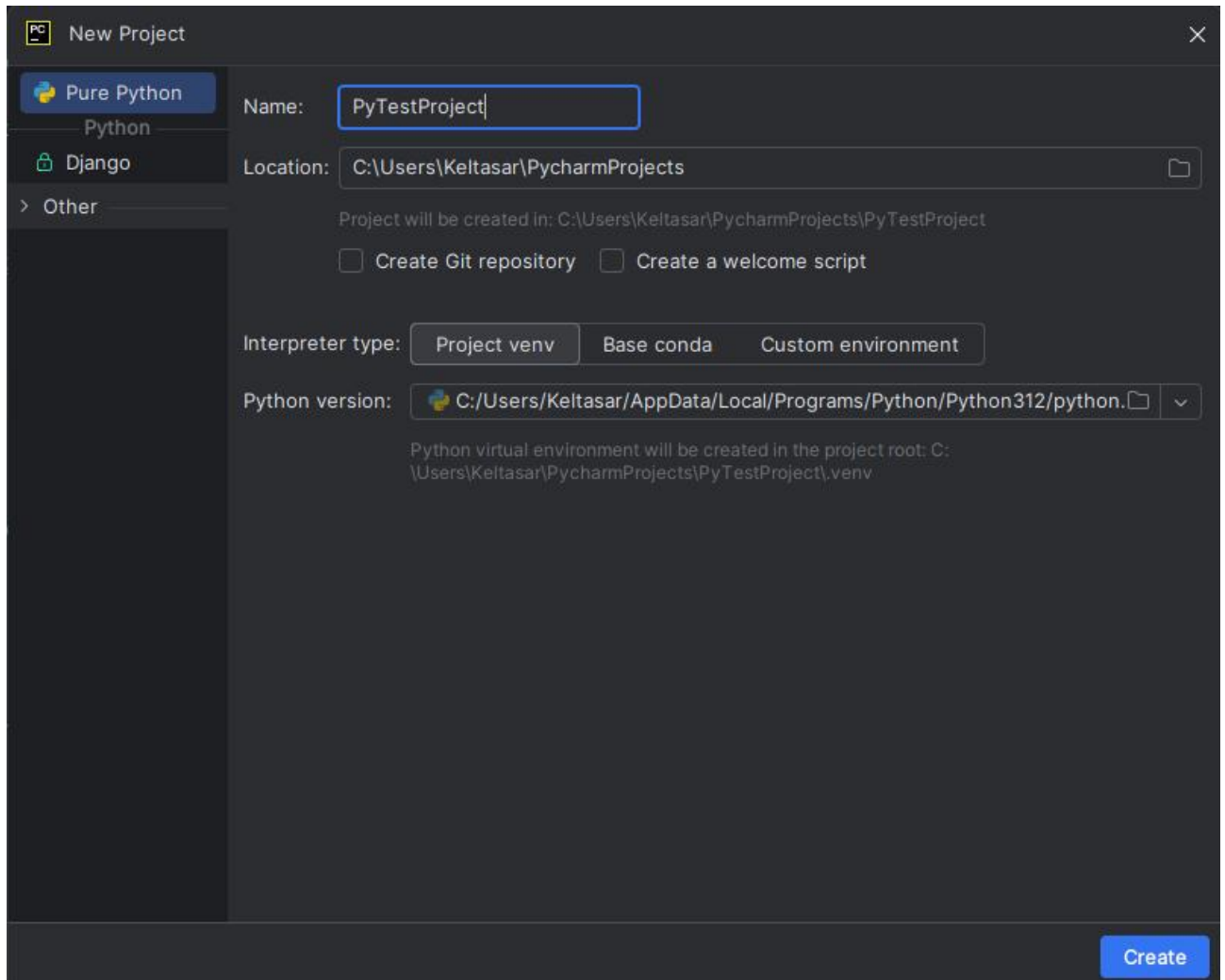


Рис. 1.1 - Первичная настройка проекта

После создания проекта проверяем наличие виртуального окружения

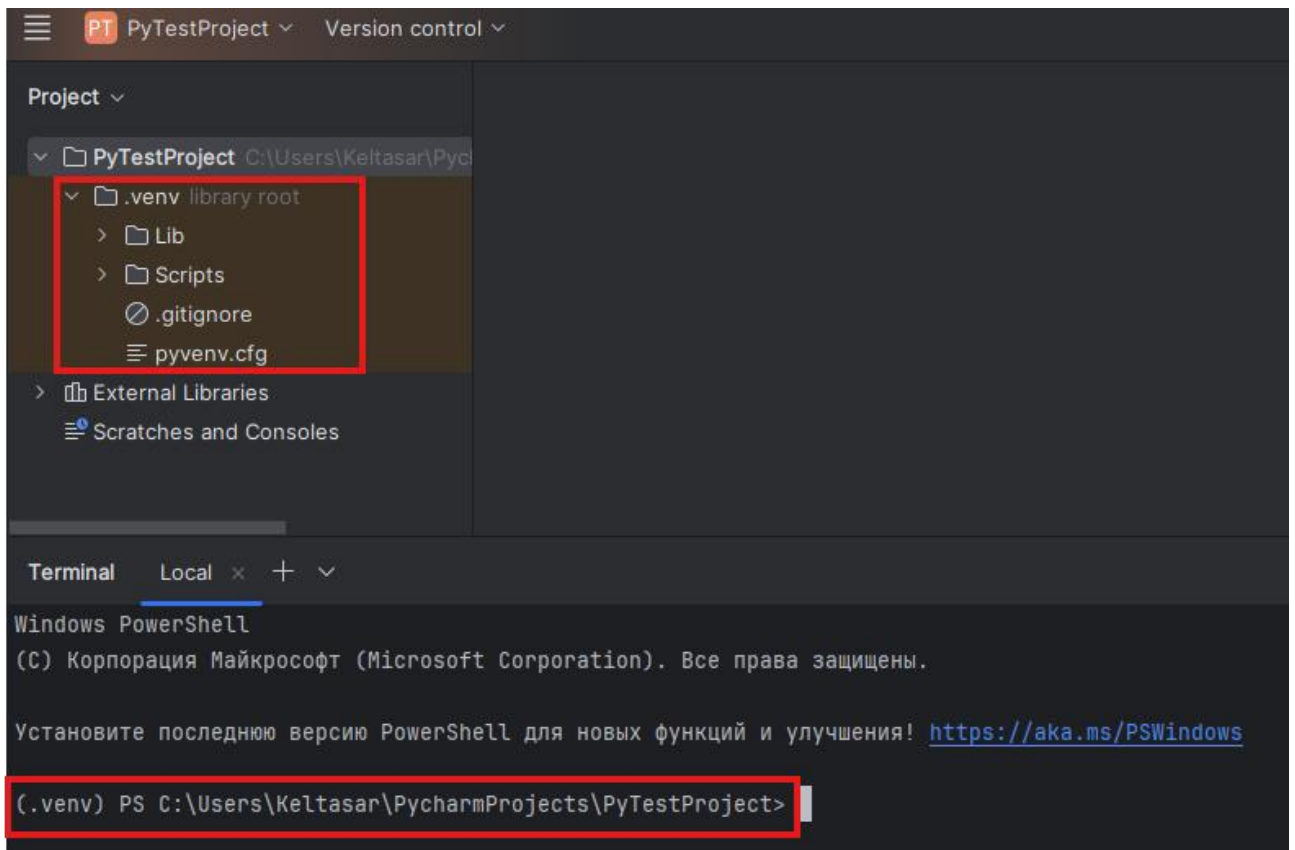


Рис. 1.2 - Проверка нахождения виртуального окружения

Далее проверяю установленные библиотеки командой “pip list”. В случае, если не установлена библиотека “pytest”, пишем в терминал команду “pip install <имя библиотеки>”.

```
(.venv) PS C:\Users\Keltasar\PycharmProjects\PyTestProject> pip list
Package    Version
-----
colorama   0.4.6
iniconfig  2.0.0
packaging  24.2
pip        23.2.1
pluggy     1.5.0
pytest     8.3.3
```

Рис. 1.3 - Проверка наличия PyTest

2. Выполнение практической части

Согласно моему варианту практического задания(вариант №5), требуется написать программу, которая принимает на вход число и проверяет, является ли оно палиндромом. Палиндромом называется число, которое одинаково читается слева направо и справа налево.

2.1. Проведение тестирования

Создаём 2 файла “palindrom.py” и “test_palindrom.py”. Далее добавляем код программы согласно практическому заданию, а в файле “test_palindrom.py” тесты для неё.

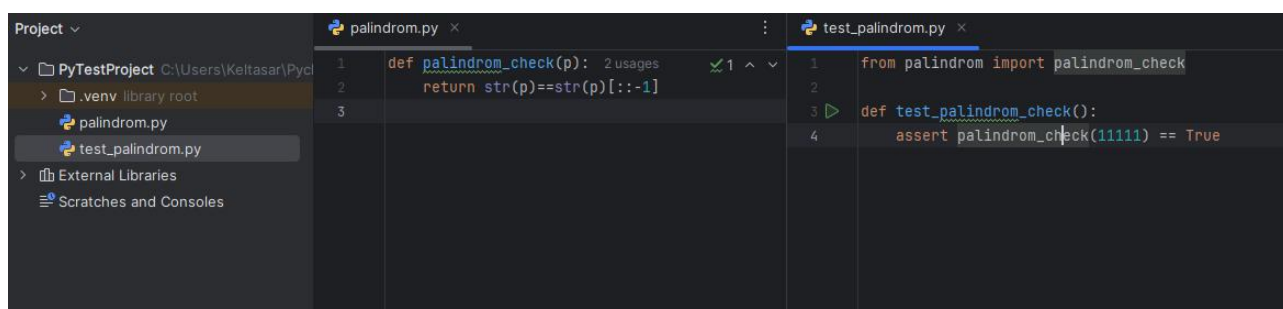


Рис. 2.1 - Файлы “palindrom.py” и “test_palindrom.py” и их код

Запускаем позитивное тестирование через терминал при помощи команды “pytest -v” , где “-v” флаг для вывода подробной информации о тестировании.

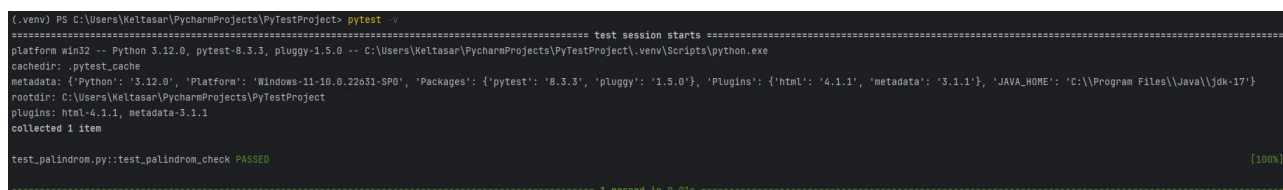


Рис. 2.2 - Результат позитивного тестирования “palindrom.py”

Для проведения негативного тестирования меняем в файле “test_palindrom.py” значение на неверное(с 11111 на 10110) и запускаем выполнение тестирования.

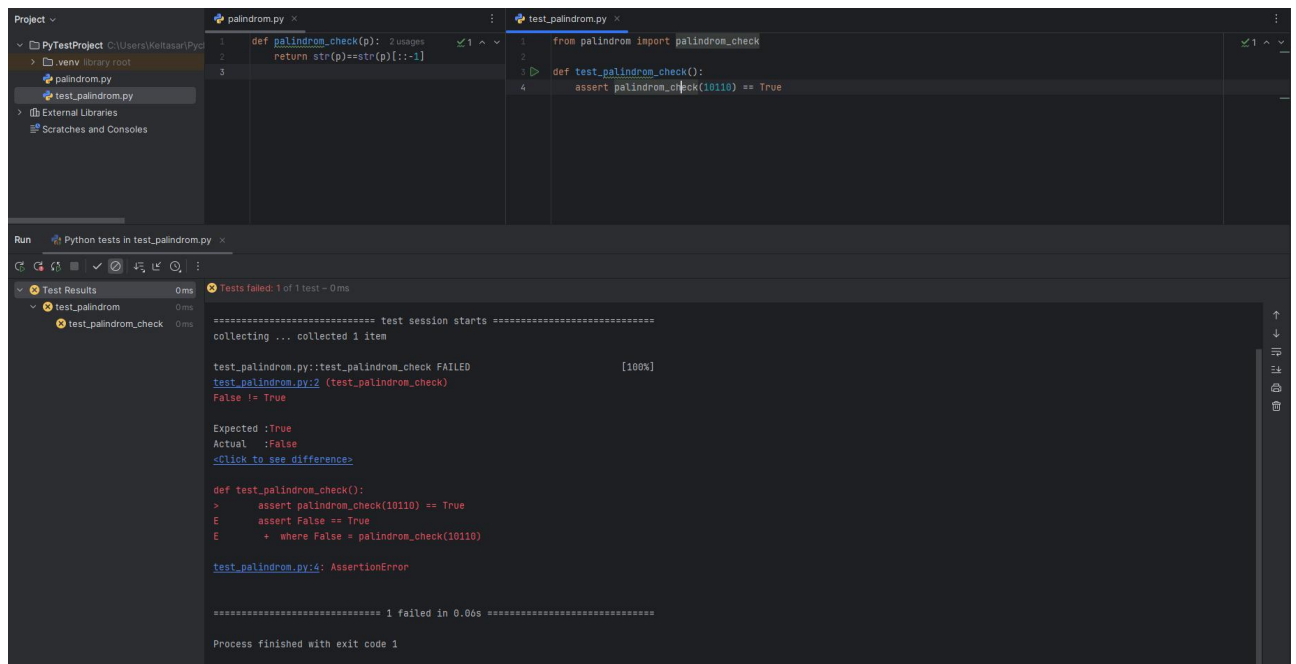


Рис. 2.3 - Результат негативного тестирования “palindrom.py”

Создадим каталоги *src* и *tests*, в которых будут находиться файлы “main.py” и “tests_main.py”(бывшие “palindrom.py” и “tests_palindrom.py”) соответственно.

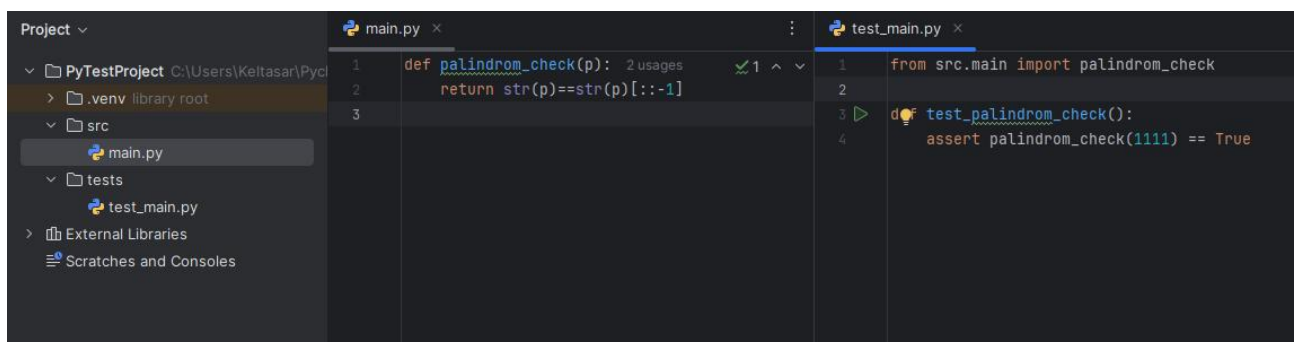


Рис. 2.4 - Файлы “main.py”, “test_main.py” и их код

Далее запускаем файл “test_main.py”, который должен вывести сообщение об успешном прохождении тестов.

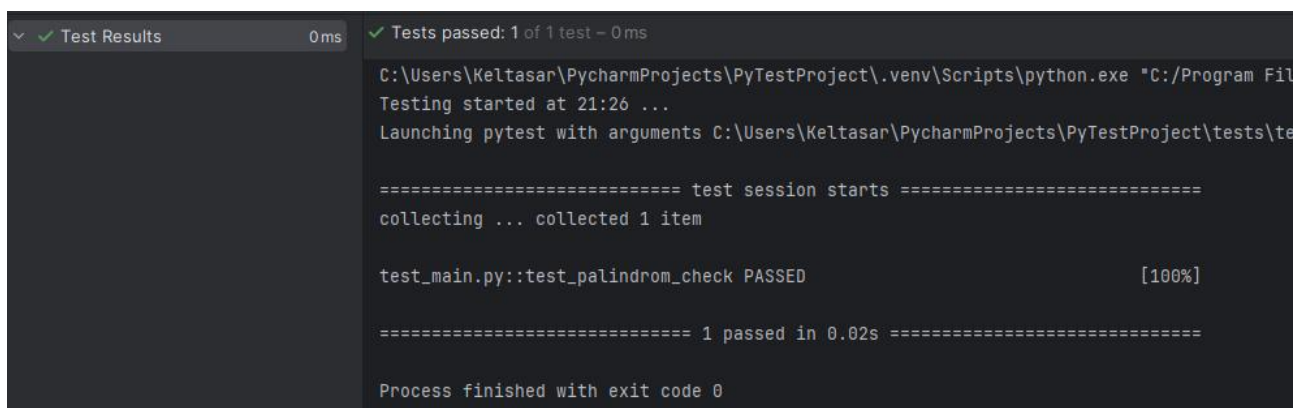


Рис. 2.5 - Результат запуска “test_main.py”

Так как файлы “main.py” и “test_main.py” находятся в разных директориях, то при запуске через терминал будет выдавать ошибку.

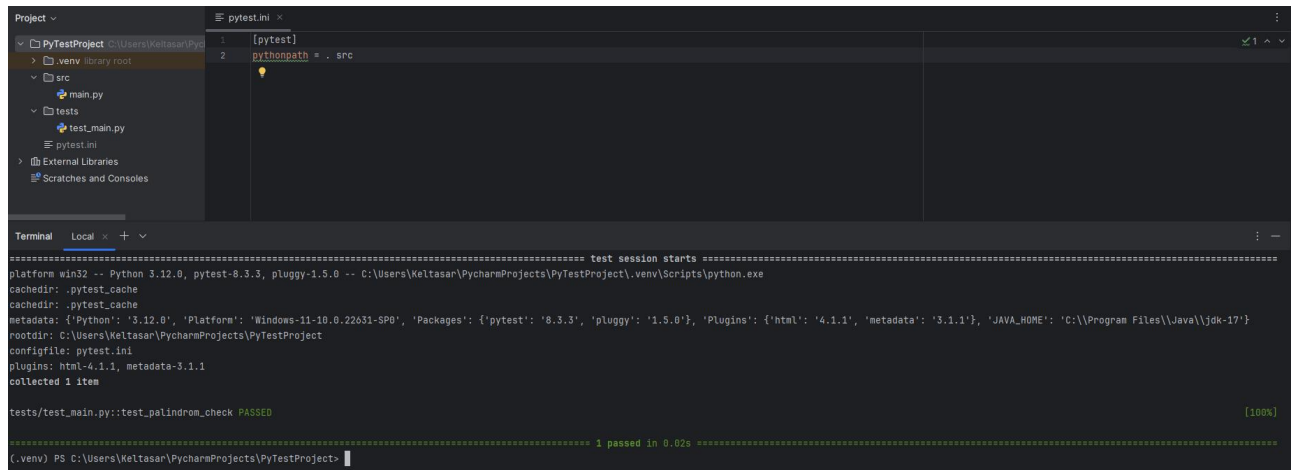
```
(.venv) PS C:\Users\Keltasar\PycharmProjects\PyTestProject> pytest -v
===== test session starts =====
platform win32 -- Python 3.12.0, pytest-8.3.3, pluggy-1.5.0 -- C:\Users\Keltasar\PycharmProjects\PyTestProject\.venv\Scripts\python.exe
cachedir: .pytest_cache
metadata: {'Python': '3.12.0', 'Platform': 'Windows-11-10.0.22631-SP0', 'Packages': {'pytest': '8.3.3', 'pluggy': '1.5.0'}, 'Plugins': {'html': '4.1.1', 'metadata': '3.1.1'}, 'JAVA_HOME': 'C:\\Program Files\\Java\\jdk-17'}
rootdir: C:\Users\Keltasar\PycharmProjects\PyTestProject
plugins: html-4.1.1, metadata-3.1.1
collected 0 items / 1 error

===== ERRORS =====
ERROR collecting tests/test_main.py
ImportError while importing test module 'C:\Users\Keltasar\PycharmProjects\PyTestProject\tests\test_main.py'.
Hint: make sure your test modules/packages have valid Python names.
Traceback:
..\.AppData\Local\Programs\Python\Python312\Lib\importlib\_init_.py:90: in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
tests\test_main.py:1: in <module>
    from src.main import palindrom_check
E ModuleNotFoundError: No module named 'src'

===== short test summary info =====
ERROR tests/test_main.py
!!!!!!!!!!!!!!!! Interrupted: 1 error during collection !!!!!!!!!!!!!!!!!
===== 1 error in 0.14s =====
```

Рис.2.6. - Результат запуска PyTest

Для устранения этой ошибки создадим в корневом каталоге файл “pytest.ini”, в котором явно укажем директорию *src*.



The screenshot shows the PyCharm IDE interface. On the left, the Project view displays the file structure: PyTestProject (root), .venv (library root), src (containing main.py), and tests (containing test_main.py and pytest.ini). The main editor window shows the content of pytest.ini:

```
[pytest]
pythonpath = . src
```

Below the editor, the Terminal window shows the output of a successful pytest run:

```
===== test session starts =====
platform win32 -- Python 3.12.0, pytest-8.3.3, pluggy-1.5.0 -- C:\Users\Keltasar\PycharmProjects\PyTestProject\.venv\Scripts\python.exe
cachedir: .pytest_cache
metadata: {'Python': '3.12.0', 'Platform': 'Windows-11-10.0.22631-SP0', 'Packages': {'pytest': '8.3.3', 'pluggy': '1.5.0'}, 'Plugins': {'html': '4.1.1', 'metadata': '3.1.1'}, 'JAVA_HOME': 'C:\\Program Files\\Java\\jdk-17'}
rootdir: C:\Users\Keltasar\PycharmProjects\PyTestProject
configfile: pytest.ini
plugins: html-4.1.1, metadata-3.1.1
collected 1 item

tests/test_main.py::test_palindrom_check PASSED [100%]

===== 1 passed in 0.02s =====
(.venv) PS C:\Users\Keltasar\PycharmProjects\PyTestProject>
```

Рис. 2.7 - Результат запуска Pytest с указанным src

Добавим в файл “test_main.py” маркер пропуска и маркер неверного выполнения теста, а также функции .

```
test_main.py x
1 import pytest
2 from src.main import palindrom_check
3
4 @pytest.mark.skip
5 def test_palindrom_check():
6     assert palindrom_check(1111) == True
7
8 @pytest.mark.xfail
9 def test_palindrom_check2():
10     assert palindrom_check(123454321) == True
11
12 @pytest.mark.xfail
13 def test_palindrom_check_false():
14     assert palindrom_check(123465432) == True
```

Рис. 2.8 - Код test_main.py с маркерами

Далее запустим PyTest и проверим выполнение тестов.

```
===== test session starts =====
platform win32 -- Python 3.12.0, pytest-8.3.3, pluggy-1.5.0 -- C:\Users\Keltasar\PycharmProjects\PyTestProject\.venv\Scripts\python.exe
cachedir: .pytest_cache
metadata: {'Python': '3.12.0', 'Platform': 'Windows-11-10.0.22631-SP0', 'Packages': {'pytest': '8.3.3', 'pluggy': '1.5.0'}, 'Plugins': {'html': '4.1.1', 'metadata': '3.1.1'}, 'JAVA_HOME': 'C:\\Program Files\\Java\\jdk-17'}
rootdir: C:\Users\Keltasar\PycharmProjects\PyTestProject
configfile: pytest.ini
plugins: html-4.1.1, metadata-3.1.1
collected 3 items

tests/test_main.py::test_palindrom_check SKIPPED (unconditional skip)
tests/test_main.py::test_palindrom_check2 XPASS
tests/test_main.py::test_palindrom_check_false XFAIL

===== 1 skipped, 1 xfailed, 1 xpassed in 0.07s =====
```

Рис. 2.9 - Запуск PyTest с маркерами

В результате выполнения PyTest можно заметить, что test_palindrom_check() было пропущено, так как мы добавили ему маркер пропуска, а также test_palindrom_check2() и test_palindrom_check_false(), которым был добавлен маркер XFAIL, отчитались как XPASS и XFAIL, что означает, что первый тест пройден успешно, а второй неверный.

Отметим пользовательским маркером mymark функции test_palindrom_check() и test_palindrom_check_false() и запустим тесты с маркером mymark командой pytest -m mymark.

```

(.venv) PS C:\Users\Keltasar\PycharmProjects\PyTestProject> pytest --m mymark
===== test session starts =====
platform win32 -- Python 3.12.0, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\Keltasar\PycharmProjects\PyTestProject
configfile: pytest.ini
plugins: html-4.1.1, metadata-3.1.1
collected 3 items / 1 deselected / 2 selected

tests\test_main.py sx [100%]
tests\test_main.py sx [100%]

===== warnings summary =====
tests\test_main.py:5
C:\Users\Keltasar\PycharmProjects\PyTestProject\tests\test_main.py:5: PytestUnknownMarkWarning: Unknown pytest.mark.mymark - is this a typo? You can register custom marks to avoid this warning - for details, see https://docs.pytest.org/en/stable/how-to/mark.html
  @pytest.mark.mymark

tests\test_main.py:14
C:\Users\Keltasar\PycharmProjects\PyTestProject\tests\test_main.py:14: PytestUnknownMarkWarning: Unknown pytest.mark.mymark - is this a typo? You can register custom marks to avoid this warning - for details, see https://docs.pytest.org/en/stable/how-to/mark.html
  @pytest.mark.mymark

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
===== 1 skipped, 1 deselected, 1 xfailed, 2 warnings in 0.07s =====

```

Рис. 2.10 - Запуск PyTest с пользовательскими маркерами

В ходе выполнения тестов PyTest вывел предупреждение, что маркер является пользовательским и не объявлен в “pytest.ini”. Для этого объявляем пользовательский маркер в файле конфигурации и запускаем PyTest заново.

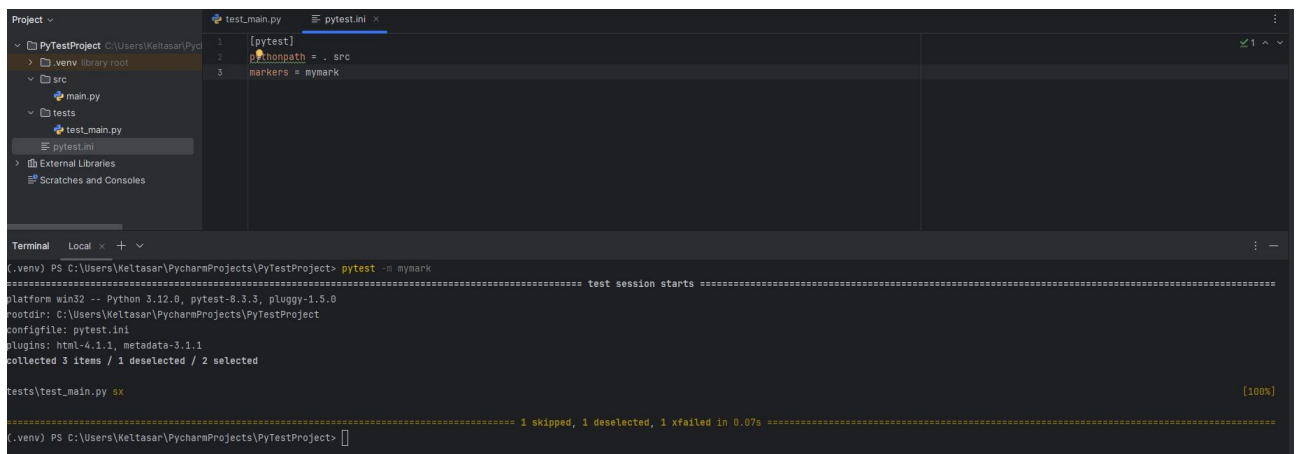


Рис. 2.11 - Объявление маркера mymark и запуск PyTest

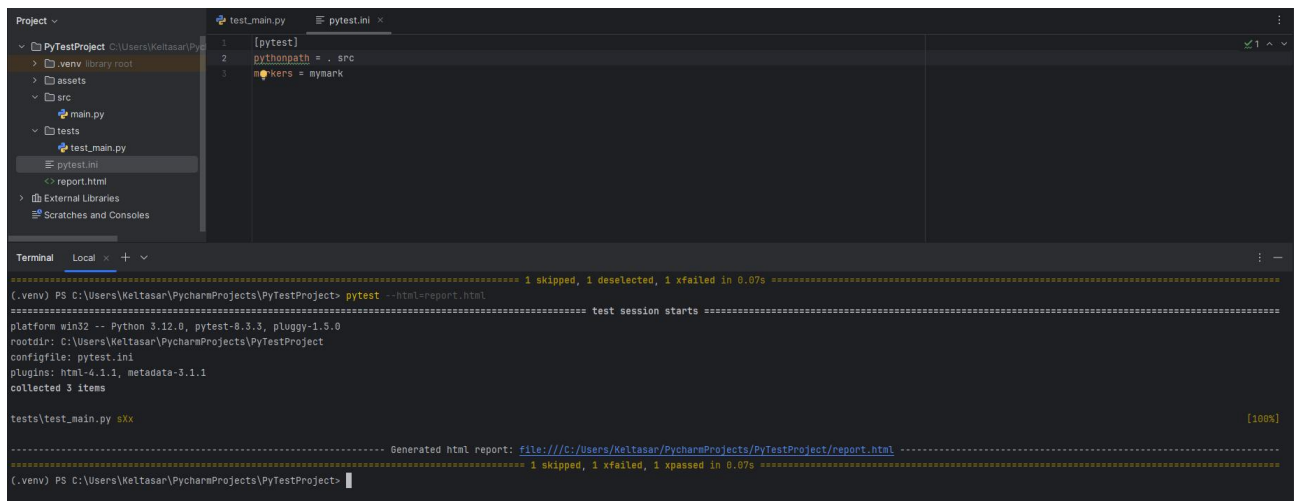
В результате выполнения PyTest мы видим, что 2 теста не проверялись, так как они не были отмечены маркером *mymark*, а остальные 2, которые были отмечены маркером *mymark*, были выполнены также, как и в предыдущем пункте.

Для составления отчётов нам понадобится плагин *pytest-html*. Для этого прописываем в терминал команду “pip install pytest-html” и проверяем установку командой “pip list”.

```
(.venv) PS C:\Users\Keltasar\PycharmProjects\PyTestProject> pip list
Package            Version
-----
colorama           0.4.6
iniconfig          2.0.0
Jinja2             3.1.4
MarkupSafe         3.0.2
packaging          24.2
pip               23.2.1
pluggy            1.5.0
pytest            8.3.3
pytest-html       4.1.1
pytest-metadata   3.1.1
```

Рис. 2.12 - Установка pytest-html

Для формирования отчета необходимо запустить pytest, добавив аргумент `--html` с указанием пути к файлу для сохранения отчета.



```
Project: PyTestProject
test_main.py
pytest.ini
report.html
External Libraries
Scratches and Consoles

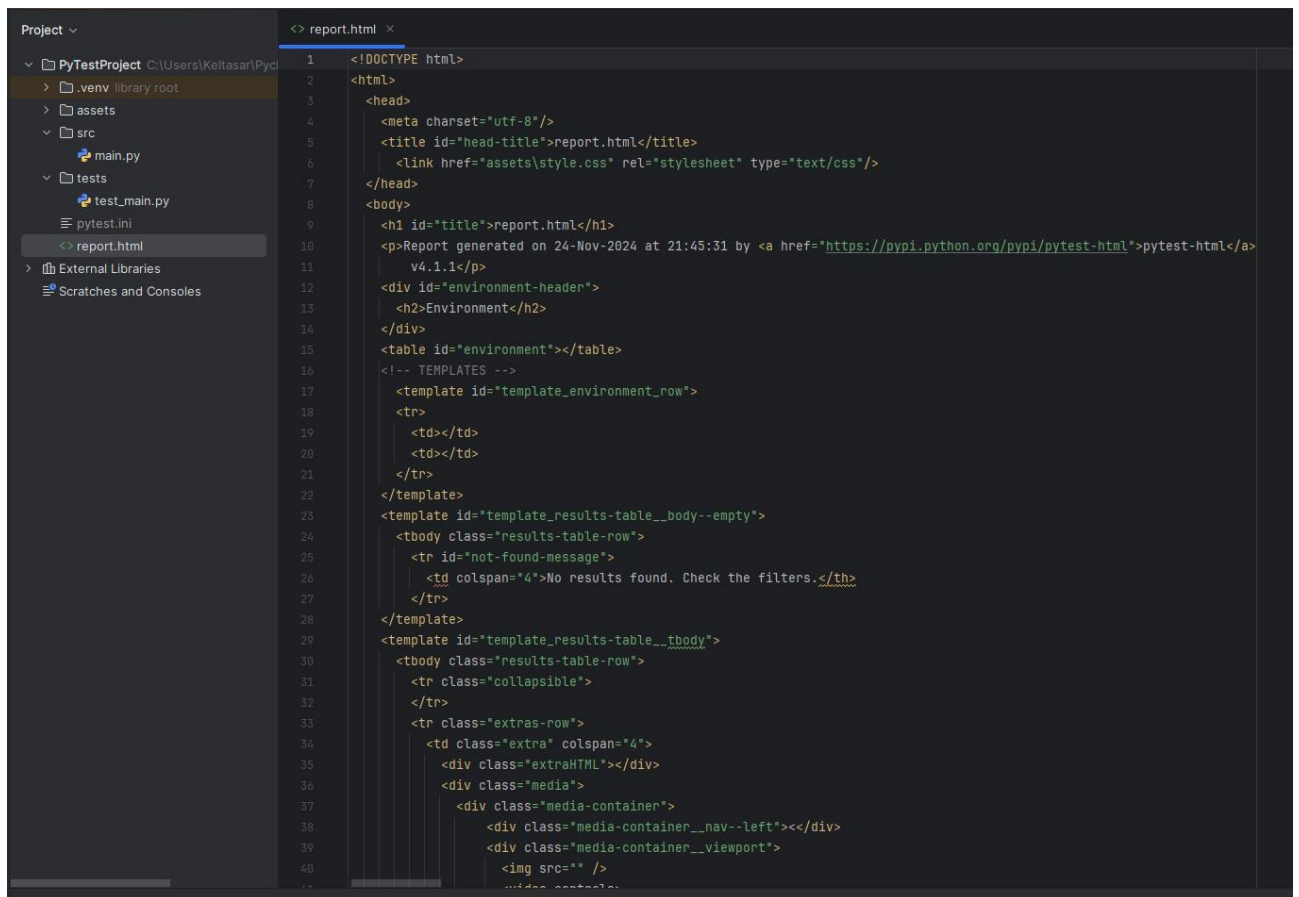
Terminal
(.venv) PS C:\Users\Keltasar\PycharmProjects\PyTestProject> pytest --html-report.html
===== 1 skipped, 1 deselected, 1 xfailed in 0.07s =====
test session starts
platform win32 -- Python 3.12.0, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\Keltasar\PycharmProjects\PyTestProject
configfile: pytest.ini
plugins: html-4.1.1, metadata-3.1.1
collected 3 items

tests\test_main.py sxx

===== Generated html report: file:///C:/Users/Keltasar/PycharmProjects/PyTestProject/report.html =====
===== 1 skipped, 1 xfailed, 1 xpassed in 0.07s =====
(.venv) PS C:\Users\Keltasar\PycharmProjects\PyTestProject>
```

Рис.2.13 - Результат выполнения PyTest с формированием отчёта

После этого у нас будет создан файл “report.html”, в котором и сформирован отчёт, однако при открытии его в редакторе будет только лишь код на HTML, так что открывать его придётся через браузер.



```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<title id="head-title">report.html</title>
<link href="assets/style.css" rel="stylesheet" type="text/css"/>
</head>
<body>
<h1 id="title">report.html</h1>
<p>Report generated on 24-Nov-2024 at 21:45:31 by <a href="https://pypi.python.org/pypi/pytest-html">pytest-html</a> v4.1.1</p>
<div id="environment-header">
<h2>Environment</h2>
</div>
<table id="environment">
<!-- TEMPLATES -->
<template id="template_environment_row">
<tr>
<td></td>
<td></td>
</tr>
</template>
<template id="template_results-table__body--empty">
<tbody class="results-table-row">
<tr id="not-found-message">
<td colspan="4">No results found. Check the filters.</td>
</tr>
</template>
<template id="template_results-table__tbody">
<tbody class="results-table-row">
<tr class="collapsible">
</tr>
<tr class="extras-row">
<td class="extra" colspan="4">
<div class="extraHTML">
<div class="media">
<div class="media-container">
<div class="media-container__nav--left">
<div class="media-container__viewport">
<img src="" />
</div>
</div>
</div>
</div>
</td>
</tr>
</tbody>
</table>
```

Рис. 2.14 - Код отчёта PyTest

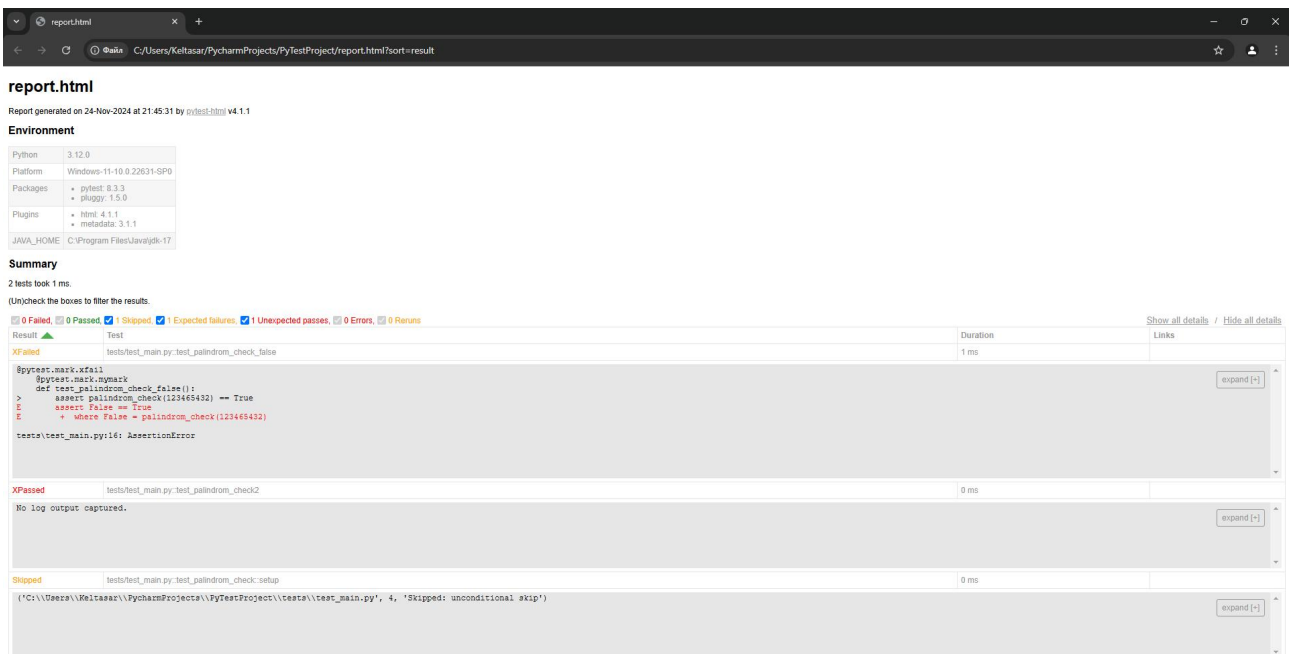


Рис.2.15 - Отчёт PyTest открытый через браузер

Вывод

В ходе практики были успешно выполнены все поставленные задачи: установлено и настроено программное окружение, написана и протестирована программа для проверки числа на предмет того, является ли оно палиндромом. Также были изучены основные методы работы с PyTest: создание и написание файла конфигурации, написание и использование фикстур и маркеров. PyTest хорошо показал себя в тестировании программ написанных на языке программирования python методом «белого» ящика. По итогам практики были освоены и применены принципы автоматизации тестирования программ на языке python, которые пригодятся в профессиональной деятельности.

Листинг кода main.py

```
def palindrom_check(p):  
    return str(p)==str(p)[::-1]
```

Листинг кода test_main.py

```
import pytest
from src.main import palindrom_check
```

```
@pytest.mark.skip
@pytest.mark.mymark
def test_palindrom_check():
    assert palindrom_check(1111) == True
```

```
@pytest.mark.xfail
def test_palindrom_check2():
    assert palindrom_check(123454321) == True
```

```
@pytest.mark.xfail
@pytest.mark.mymark
def test_palindrom_check_false():
    assert palindrom_check(123465432) == True
```


Листинг кода pytest.ini

```
[pytest]  
pythonpath = . src  
markers = mymark
```

Листинг кода report.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <title id="head-title">report.html</title>
    <link href="assets\style.css" rel="stylesheet" type="text/css"/>
  </head>
  <body>
    <h1 id="title">report.html</h1>
    <p>Report generated on 24-Nov-2024 at 21:45:31 by <a
href="https://pypi.python.org/pypi/pytest-html">pytest-html</a>
v4.1.1</p>
    <div id="environment-header">
      <h2>Environment</h2>
    </div>
    <table id="environment"></table>
    <!-- TEMPLATES -->
    <template id="template_environment_row">
      <tr>
        <td></td>
        <td></td>
      </tr>
    </template>
    <template id="template_results-table_body--empty">
      <tbody class="results-table-row">
        <tr id="not-found-message">
          <td colspan="4">No results found. Check the filters.</td>
        </tr>
      </tbody>
    </template>
    <template id="template_results-table_tbody">
      <tbody class="results-table-row">
        <tr class="collapsible">
          <td></td>
          <td></td>
          <td></td>
          <td></td>
        </tr>
      </tbody>
    </template>
  </body>
</html>
```

```

<tr class="extras-row">
  <td class="extra" colspan="4">
    <div class="extraHTML"></div>
    <div class="media">
      <div class="media-container">
        <div class="media-container_nav--left"></div>
        <div class="media-container_viewport">
          <img src="" />
          <video controls>
            <source src="" type="video/mp4">
          </video>
        </div>
        <div class="media-container_nav--right"></div>
      </div>
      <div class="media_name"></div>
      <div class="media_counter"></div>
    </div>
    <div class="logwrapper">
      <div class="logexpander"></div>
      <div class="log"></div>
    </div>
  </td>
</tr>
</tbody>
</template>
<!-- END TEMPLATES -->
<div class="summary">
  <div class="summary_data">
    <h2>Summary</h2>
    <div class="additional-summary prefix">
    </div>
    <p class="run-count">2 tests took 1 ms.</p>
    <p class="filter">(Un)check the boxes to filter the results.</p>
    <div class="summary_reload">

```

```
<div class="summary_reload_button hidden"
onclick="location.reload()">
    <div>There are still tests running. <br />Reload this page to get the
latest results!</div>
</div>
</div>
<div class="summary_spacer"></div>
<div class="controls">
    <div class="filters">
        <input checked="true" class="filter" name="filter_checkbox"
type="checkbox" data-test-result="failed" disabled/>
        <span class="failed">0 Failed,</span>
        <input checked="true" class="filter" name="filter_checkbox"
type="checkbox" data-test-result="passed" disabled/>
        <span class="passed">0 Passed,</span>
        <input checked="true" class="filter" name="filter_checkbox"
type="checkbox" data-test-result="skipped" />
        <span class="skipped">1 Skipped,</span>
        <input checked="true" class="filter" name="filter_checkbox"
type="checkbox" data-test-result="xfailed" />
        <span class="xfailed">1 Expected failures,</span>
        <input checked="true" class="filter" name="filter_checkbox"
type="checkbox" data-test-result="xpassed" />
        <span class="xpassed">1 Unexpected passes,</span>
        <input checked="true" class="filter" name="filter_checkbox"
type="checkbox" data-test-result="error" disabled/>
        <span class="error">0 Errors,</span>
        <input checked="true" class="filter" name="filter_checkbox"
type="checkbox" data-test-result="rerun" disabled/>
        <span class="rerun">0 Reruns</span>
    </div>
    <div class="collapse">
        <button id="show_all_details">Show all
details</button>&nbsp;/&nbsp;<button id="hide_all_details">Hide all
```

```

details</button>
    </div>
</div>
</div>
<div class="additional-summary summary">
</div>
<div class="additional-summary postfix">
</div>
</div>
<table id="results-table">
  <thead id="results-table-head">
    <tr>
      <th class="sortable" data-column-type="result">Result</th>
      <th class="sortable" data-column-type="testId">Test</th>
      <th class="sortable" data-column-type="duration">Duration</th>
      <th>Links</th>
    </tr>
  </thead>
</table>
</body>
<footer>
  <div id="data-container" data-jsonblob="{&#34;environment&#34;:
{&#34;Python&#34;: &#34;3.12.0&#34;, &#34;Platform&#34;:
&#34;Windows-11-10.0.22631-SPO&#34;, &#34;Packages&#34;:
{&#34;pytest&#34;: &#34;8.3.3&#34;, &#34;pluggy&#34;:
&#34;1.5.0&#34;}, &#34;Plugins&#34;: {&#34;html&#34;:
&#34;4.1.1&#34;, &#34;metadata&#34;: &#34;3.1.1&#34;},
&#34;JAVA_HOME&#34;: &#34;C:\\Program Files\\Java\\jdk-17&#34;},
&#34;tests&#34;: {&#34;tests/test_main.py::test_palindrom_check&#34;:
[&#34;extras&#34;: [], &#34;result&#34;: &#34;Skipped&#34;,
&#34;testId&#34;:
&#34;tests/test_main.py::test_palindrom_check::setup&#34;,
&#34;duration&#34;: &#34;0 ms&#34;, &#34;resultsTableRow&#34;:
[&#34;&lt;td class=\\&#34;col-result\\&#34;&gt;Skipped&lt;/td&#34;,

```

```

<td class=
testId>tests/test_main.py::test_palindrom_check::setup</td>
, <td class=
ms</td>, <td class=
links><</td>], <log>
<C:\\\\Users\\\\Keltasar\\\\PycharmProjects\\\\PyTestP
project\\\\tests\\\\test_main.py&#x27;, 4, &#x27;Skipped:
unconditional skip&#x27;)\n<]>,
<tests/test_main.py::test_palindrom_check2>: [{<extras>:
[], <result>: <XPassed>, <testId>:
<tests/test_main.py::test_palindrom_check2>,
<duration>: <0 ms>, <resultsTableRow>:
[<td class=
<td class=
testId>>tests/test_main.py::test_palindrom_check2</td>>,
<td class=
<td class=
links><</td>],
<log>: <No log output captured.>]>,
<tests/test_main.py::test_palindrom_check_false>:
[<extras>: [], <result>: <XFailed>,
<testId>:
<tests/test_main.py::test_palindrom_check_false>,
<duration>: <1 ms>, <resultsTableRow>:
[<td class=
<td class=
testId>>tests/test_main.py::test_palindrom_check_false</td>>
, <td class=
ms</td>, <td class=
links><</td>], <log>:
<@pytest.mark.xfail\n @pytest.mark.mymark\n def
test_palindrom_check_false():\n&#gt;    assert
palindrom_check(123465432) == True\nE    assert False == True\nE
+ where False = palindrom_check(123465432)\n\ntests\test_main.py:16:
AssertionError\n<]>], <renderCollapsed>:

```

```
[&#34;passed&#34;], &#34;initialSort&#34;: &#34;result&#34;,
&#34;title&#34;: &#34;report.html&#34;}"></div>
```

```
<script>
  (function(){function r(e,n,t){function o(i,f){if(!n[i]){if(!e[i]){var
c="function"==typeof require&&require;if(!f&&c)return c(i,!0);if(u)return
u(i,!0);var a=new Error("Cannot find module '"+i+"'");throw
a.code="MODULE_NOT_FOUND",a}var
p=n[i]={exports:{}};e[i][0].call(p.exports,function(r){var n=e[i][1][r];return
o(n||r)},p,p.exports,r,e,n,t)}return n[i].exports}for(var u="function"==typeof
require&&require,i=0;i<t.length;i++)o(t[i]);return o}r})({1:[function(require,module,exports){
const { getCollapsedCategory, setCollapsedIds } = require('./storage.js')
```

```
class DataManager {
  setManager(data) {
    const collapsedCategories =
[...getCollapsedCategory(data.renderCollapsed)]
    const collapsedIds = []
    const tests = Object.values(data.tests).flat().map((test, index) => {
      const collapsed =
collapsedCategories.includes(test.result.toLowerCase())
      const id = `test_${index}`
      if (collapsed) {
        collapsedIds.push(id)
      }
      return {
        ...test,
        id,
        collapsed,
      }
    })
    const dataBlob = { ...data, tests }
    this.data = { ...dataBlob }
    this.renderData = { ...dataBlob }
```

```
    setCollapsedIds(collapsedIds)  
  }
```

```
  get allData() {  
    return { ...this.data }  
  }
```

```
  resetRender() {  
    this.renderData = { ...this.data }  
  }
```

```
  setRender(data) {  
    this.renderData.tests = [...data]  
  }
```

```
  toggleCollapsedItem(id) {  
    this.renderData.tests = this.renderData.tests.map((test) =>  
      test.id === id ? { ...test, collapsed: !test.collapsed } : test,  
    )  
  }
```

```
  set allCollapsed(collapsed) {  
    this.renderData = { ...this.renderData, tests:  
      [...this.renderData.tests.map((test) => (  
        { ...test, collapsed }  
      ))] }  
  }
```

```
  get testSubset() {  
    return [...this.renderData.tests]  
  }
```

```
  get environment() {  
    return this.renderData.environment
```



```
}
```

```
get initialSort() {  
  return this.data.initialSort  
}
```

```
module.exports = {  
  manager: new DataManager(),  
}
```

```
}, {"/storage.js": 8}], 2: [function (require, module, exports) {  
  const mediaViewer = require('./mediaviewer.js')  
  const templateEnvRow =  
  document.getElementById('template_environment_row')  
  const templateResult = document.getElementById('template_results-  
table_tbody')
```

```
function htmlToElements(html) {  
  const temp = document.createElement('template')  
  temp.innerHTML = html  
  return temp.content.childNodes  
}
```

```
const find = (selector, elem) => {  
  if (!elem) {  
    elem = document  
  }  
  return elem.querySelector(selector)  
}
```

```
const findAll = (selector, elem) => {  
  if (!elem) {  
    elem = document
```

```

    }
    return [...elem.querySelectorAll(selector)]
  }
}

```

```

const dom = {
  getStaticRow: (key, value) => {
    const envRow = templateEnvRow.content.cloneNode(true)
    const isObj = typeof value === 'object' && value !== null
    const values = isObj ? Object.keys(value).map((k) => `${k}: ${value[k]}`) :
    null
  }
}

```

```

    const valuesElement = htmlToElements(
      values ? `<ul>${values.map((val) => `<li>${val}</li>`).join("")}</ul>` :
      `<div>${value}</div>`)[0]
    const td = findAll('td', envRow)
    td[0].textContent = key
    td[1].appendChild(valuesElement)
  }
}

```

```

    return envRow
  },
  getResultTBody: ({ testId, id, log, extras, resultsTableRow, tableHtml, result,
  collapsed }) => {
    const resultBody = templateResult.content.cloneNode(true)
    resultBody.querySelector('tbody').classList.add(result.toLowerCase())
    resultBody.querySelector('tbody').id = testId
    resultBody.querySelector('.collapsible').dataset.id = id
  }
}

```

```

    resultsTableRow.forEach((html) => {
      const t = document.createElement('template')
      t.innerHTML = html
      resultBody.querySelector('.collapsible').appendChild(t.content)
    })
  }
}

```

```

    if (log) {

```

```

        // Wrap lines starting with "E" with span.error to color those lines
red
        const wrappedLog = log.replace(/^E.*$/gm, (match) => `<span
class="error">${match}</span>`)
        resultBody.querySelector('.log').innerHTML = wrappedLog
    } else {
        resultBody.querySelector('.log').remove()
    }
}

```

```

    if (collapsed) {
        resultBody.querySelector('.collapsible > td')?.classList.add('collapsed')
        resultBody.querySelector('.extras-row').classList.add('hidden')
    } else {
        resultBody.querySelector('.collapsible >
td')?.classList.remove('collapsed')
    }
}

```

```

const media = []
extras?.forEach(({ name, format_type, content }) => {
    if (['image', 'video'].includes(format_type)) {
        media.push({ path: content, name, format_type })
    }
})

```

```

        if (format_type === 'html') {
            resultBody.querySelector('.extraHTML').insertAdjacentHTML('beforeend',
`<div>${content}</div>`)
        }
    })
    mediaViewer.setup(resultBody, media)
}

```

```

// Add custom html from the pytest_html_results_table_html hook
tableHtml?.forEach((item) => {

```

```
resultBody.querySelector('td[class="extra"]').insertAdjacentHTML('beforeend',  
item)  
  })  
}
```

```
    return resultBody  
  },  
}
```

```
module.exports = {  
  dom,  
  htmlToElements,  
  find,  
  findAll,  
}
```

```
},["./mediaviewer.js":6}],3:[function(require,module,exports){  
const { manager } = require('./datamanager.js')  
const { doSort } = require('./sort.js')  
const storageModule = require('./storage.js')
```

```
const getFilteredSubSet = (filter) =>  
  manager.allData.tests.filter(({ result }) =>  
filter.includes(result.toLowerCase()))
```

```
const doInitFilter = () => {  
  const currentFilter = storageModule.getVisible()  
  const filteredSubset = getFilteredSubSet(currentFilter)  
  manager.setRender(filteredSubset)  
}
```

```
const doFilter = (type, show) => {  
  if (show) {  
    storageModule.showCategory(type)  
  } else {
```

```
storageModule.hideCategory(type)  
}
```

```
const currentFilter = storageModule.getVisible()  
const filteredSubset = getFilteredSubSet(currentFilter)  
manager.setRender(filteredSubset)
```

```
const sortColumn = storageModule.getSort()  
doSort(sortColumn, true)  
}
```

```
module.exports = {  
  doFilter,  
  doInitFilter,  
}
```

```
}, {"/datamanager.js": 1, "/sort.js": 7, "/storage.js": 8}], 4: [function(require, module,  
exports) {  
  const { redraw, bindEvents, renderStatic } = require('./main.js')  
  const { doInitFilter } = require('./filter.js')  
  const { doInitSort } = require('./sort.js')  
  const { manager } = require('./datamanager.js')  
  const data = JSON.parse(document.getElementById('data-  
container').dataset.jsonblob)
```

```
function init() {  
  manager.setManager(data)  
  doInitFilter()  
  doInitSort()  
  renderStatic()  
  redraw()  
  bindEvents()  
}
```

```
init()
```

```
}, { './datamanager.js': 1, './filter.js': 3, './main.js': 5, './sort.js': 7 }], 5: [function (require, module, exports) {  
  const { dom, find, findAll } = require('./dom.js')  
  const { manager } = require('./datamanager.js')  
  const { doSort } = require('./sort.js')  
  const { doFilter } = require('./filter.js')  
  const {  
    getVisible,  
    getCollapsedIds,  
    setCollapsedIds,  
    getSort,  
    getSortDirection,  
    possibleFilters,  
  } = require('./storage.js')
```

```
const removeChildren = (node) => {  
  while (node.firstChild) {  
    node.removeChild(node.firstChild)  
  }  
}
```

```
const renderStatic = () => {  
  const renderEnvironmentTable = () => {  
    const environment = manager.environment  
    const rows = Object.keys(environment).map((key) =>  
      dom.getStaticRow(key, environment[key]))  
    const table = document.getElementById('environment')  
    removeChildren(table)  
    rows.forEach((row) => table.appendChild(row))  
  }  
  renderEnvironmentTable()  
}
```

```
const addItemToggleListener = (elem) => {  
  elem.addEventListener('click', ({ target }) => {  
    const id = target.parentElement.dataset.id  
    manager.toggleCollapsedItem(id)
```

```
    const collapsedIds = getCollapsedIds()  
    if (collapsedIds.includes(id)) {  
      const updated = collapsedIds.filter((item) => item !== id)  
      setCollapsedIds(updated)  
    } else {  
      collapsedIds.push(id)  
      setCollapsedIds(collapsedIds)  
    }  
    redraw()  
  })  
}
```

```
const renderContent = (tests) => {  
  const sortAttr = getSort(manager.initialSort)  
  const sortAsc = JSON.parse(getSortDirection())  
  const rows = tests.map(dom.getResultTBody)  
  const table = document.getElementById('results-table')  
  const tableHeader = document.getElementById('results-table-head')
```

```
  const newTable = document.createElement('table')  
  newTable.id = 'results-table'
```

```
  // remove all sorting classes and set the relevant  
  findAll('.sortable', tableHeader).forEach((elem) => elem.classList.remove('asc',  
'desc'))  
  tableHeader.querySelector(`.sortable[data-column-  
type="${sortAttr}"]`)?.classList.add(sortAsc ? 'desc' : 'asc')  
  newTable.appendChild(tableHeader)
```

```

    if (!rows.length) {
      const emptyTable = document.getElementById('template_results-
table__body--empty').content.cloneNode(true)
      newTable.appendChild(emptyTable)
    } else {
      rows.forEach((row) => {
        if (!!row) {
          findAll('.collapsible td:not(.col-links',
row).forEach(addItemToggleListener)
          find('.logexpander', row).addEventListener('click',
            (evt) => evt.target.parentNode.classList.toggle('expanded'),
          )
          newTable.appendChild(row)
        }
      })
    }
  }
}

```

```

    table.replaceWith(newTable)
  }
}

```

```

const renderDerived = () => {
  const currentFilter = getVisible()
  possibleFilters.forEach((result) => {
    const input = document.querySelector(`input[data-test-
result="${result}"]`)
    input.checked = currentFilter.includes(result)
  })
}

```

```

const bindEvents = () => {
  const filterColumn = (evt) => {
    const { target: element } = evt
    const { testResult } = element.dataset
  }
}

```



```

doFilter(testResult, element.checked)
const collapsedIds = getCollapsedIds()
const updated = manager.renderData.tests.map((test) => {
  return {
    ...test,
    collapsed: collapsedIds.includes(test.id),
  }
})
manager.setRender(updated)
redraw()
}

```

```

const header = document.getElementById('environment-header')
header.addEventListener('click', () => {
  const table = document.getElementById('environment')
  table.classList.toggle('hidden')
  header.classList.toggle('collapsed')
})

```

```

findAll('input[name="filter_checkbox"]').forEach((elem) => {
  elem.addEventListener('click', filterColumn)
})

```

```

findAll('.sortable').forEach((elem) => {
  elem.addEventListener('click', (evt) => {
    const { target: element } = evt
    const { columnType } = element.dataset
    doSort(columnType)
    redraw()
  })
})

```

```

document.getElementById('show_all_details').addEventListener('click', () => {

```

```

        manager.allCollapsed = false
        setCollapsedIds([])
        redraw()
    })
    document.getElementById('hide_all_details').addEventListener('click', () => {
        manager.allCollapsed = true
        const allIds = manager.renderData.tests.map((test) => test.id)
        setCollapsedIds(allIds)
        redraw()
    })
}

```

```

const redraw = () => {
    const { testSubset } = manager

```

```

    renderContent(testSubset)
    renderDerived()
}

```

```

module.exports = {
    redraw,
    bindEvents,
    renderStatic,
}

```

```

}, {"/datamanager.js":1, "/dom.js":2, "/filter.js":3, "/sort.js":7, "/storage.js":8}], 6:
[function(require,module,exports){
class MediaViewer {
    constructor(assets) {
        this.assets = assets
        this.index = 0
    }
}

```

```

    nextActive() {

```

```
    this.index = this.index === this.assets.length - 1 ? 0 : this.index + 1  
    return [this.activeFile, this.index]  
  }  
}
```

```
prevActive() {  
    this.index = this.index === 0 ? this.assets.length - 1 : this.index - 1  
    return [this.activeFile, this.index]  
}  
}
```

```
get currentIndex() {  
    return this.index  
}  
}
```

```
get activeFile() {  
    return this.assets[this.index]  
}  
}
```

```
const setup = (resultBody, assets) => {  
    if (!assets.length) {  
        resultBody.querySelector('.media').classList.add('hidden')  
        return  
    }  
}
```

```
const mediaViewer = new MediaViewer(assets)  
const container = resultBody.querySelector('.media-container')  
const leftArrow = resultBody.querySelector('.media-container_nav--left')  
const rightArrow = resultBody.querySelector('.media-container_nav--right')  
const mediaName = resultBody.querySelector('.media_name')  
const counter = resultBody.querySelector('.media_counter')  
const imageEl = resultBody.querySelector('img')  
const sourceEl = resultBody.querySelector('source')
```

```
const videoEl = resultBody.querySelector('video')
```

```
const setImg = (media, index) => {  
  if (media?.format_type === 'image') {  
    imageEl.src = media.path
```

```
    imageEl.classList.remove('hidden')  
    videoEl.classList.add('hidden')  
  } else if (media?.format_type === 'video') {  
    sourceEl.src = media.path
```

```
    videoEl.classList.remove('hidden')  
    imageEl.classList.add('hidden')  
  }  
}
```

```
  mediaName.innerText = media?.name  
  counter.innerText = `${index + 1} / ${assets.length}`  
}  
setImg(mediaViewer.activeFile, mediaViewer.currentIndex)
```

```
const moveLeft = () => {  
  const [media, index] = mediaViewer.prevActive()  
  setImg(media, index)  
}  
const doRight = () => {  
  const [media, index] = mediaViewer.nextActive()  
  setImg(media, index)  
}  
const openImg = () => {  
  window.open(mediaViewer.activeFile.path, '_blank')  
}  
if (assets.length === 1) {  
  container.classList.add('media-container--fullscreen')  
} else {
```

```

    leftArrow.addEventListener('click', moveLeft)
    rightArrow.addEventListener('click', doRight)
  }
  imageEl.addEventListener('click', openImg)
}

```

```

module.exports = {
  setup,
}

```

```

},{}],7:[function(require,module,exports){
const { manager } = require('./datamanager.js')
const storageModule = require('./storage.js')

```

```

const genericSort = (list, key, ascending, customOrder) => {
  let sorted
  if (customOrder) {
    sorted = list.sort((a, b) => {
      const aValue = a.result.toLowerCase()
      const bValue = b.result.toLowerCase()

```

```

      const aIndex = customOrder.findIndex((item) => item.toLowerCase()
=== aValue)
      const bIndex = customOrder.findIndex((item) => item.toLowerCase()
=== bValue)

```

```

      // Compare the indices to determine the sort order
      return aIndex - bIndex
    })
  } else {
    sorted = list.sort((a, b) => a[key] === b[key] ? 0 : a[key] > b[key] ? 1 : -
1)
  }
}

```

```

    if (ascending) {
      sorted.reverse()
    }
    return sorted
  }
}

```

```

const durationSort = (list, ascending) => {
  const parseDuration = (duration) => {
    if (duration.includes(':')) {
      // If it's in the format "HH:mm:ss"
      const [hours, minutes, seconds] = duration.split(':').map(Number)
      return (hours * 3600 + minutes * 60 + seconds) * 1000
    } else {
      // If it's in the format "nnn ms"
      return parseInt(duration)
    }
  }
  const sorted = list.sort((a, b) => parseDuration(a['duration']) -
    parseDuration(b['duration']))
  if (ascending) {
    sorted.reverse()
  }
  return sorted
}

```

```

const doInitSort = () => {
  const type = storageModule.getSort(manager.initialSort)
  const ascending = storageModule.getSortDirection()
  const list = manager.testSubset
  const initialOrder = ['Error', 'Failed', 'Rerun', 'XFailed', 'XPassed', 'Skipped',
    'Passed']

```

```

    storageModule.setSort(type)
    storageModule.setSortDirection(ascending)

```

```

    if (type?.toLowerCase() === 'original') {
      manager.setRender(list)
    } else {
      let sortedList
      switch (type) {
        case 'duration':
          sortedList = durationSort(list, ascending)
          break
        case 'result':
          sortedList = genericSort(list, type, ascending, initialOrder)
          break
        default:
          sortedList = genericSort(list, type, ascending)
          break
      }
      manager.setRender(sortedList)
    }
  }
}

```

```

const doSort = (type, skipDirection) => {
  const newSortType = storageModule.getSort(manager.initialSort) !== type
  const currentAsc = storageModule.getSortDirection()
  let ascending
  if (skipDirection) {
    ascending = currentAsc
  } else {
    ascending = newSortType ? false : !currentAsc
  }
  storageModule.setSort(type)
  storageModule.setSortDirection(ascending)
}

```

```

const list = manager.testSubset
const sortedList = type === 'duration' ? durationSort(list, ascending) :

```

```
genericSort(list, type, ascending)
  manager.setRender(sortedList)
}
```

```
module.exports = {
  doInitSort,
  doSort,
}
```

```
}, ['./datamanager.js': 1, './storage.js': 8]], 8: [function(require, module, exports) {
const possibleFilters = [
  'passed',
  'skipped',
  'failed',
  'error',
  'xfailed',
  'xpassed',
  'rerun',
]
}
```

```
const getVisible = () => {
  const url = new URL(window.location.href)
  const settings = new URLSearchParams(url.search).get('visible')
  const lower = (item) => {
    const lowerItem = item.toLowerCase()
    if (possibleFilters.includes(lowerItem)) {
      return lowerItem
    }
  }
  return null
}
return settings === null ?
  possibleFilters :
  [...new Set(settings?.split(',').map(lower).filter((item) => item))]
}
```



```
const hideCategory = (categoryToHide) => {
  const url = new URL(window.location.href)
  const visibleParams = new URLSearchParams(url.search).get('visible')
  const currentVisible = visibleParams ? visibleParams.split(',') :
[...possibleFilters]
  const settings = [...new Set(currentVisible)].filter((f) => f !==
categoryToHide).join(',')

```

```
url.searchParams.set('visible', settings)
window.history.pushState({}, null, unescape(url.href))
}

```

```
const showCategory = (categoryToShow) => {
  if (typeof window === 'undefined') {
    return
  }
  const url = new URL(window.location.href)
  const currentVisible = new
URLSearchParams(url.search).get('visible')?.split(',').filter(Boolean) ||
[...possibleFilters]
  const settings = [...new Set([categoryToShow, ...currentVisible])]
  const noFilter = possibleFilters.length === settings.length || !settings.length

```

```
noFilter ? url.searchParams.delete('visible') : url.searchParams.set('visible',
settings.join(','))
window.history.pushState({}, null, unescape(url.href))
}

```

```
const getSort = (initialSort) => {
  const url = new URL(window.location.href)
  let sort = new URLSearchParams(url.search).get('sort')
  if (!sort) {
    sort = initialSort || 'result'
  }

```

```

    }
    return sort
  }
}

```

```

const setSort = (type) => {
  const url = new URL(window.location.href)
  url.searchParams.set('sort', type)
  window.history.pushState({}, null, unescape(url.href))
}

```

```

const getCollapsedCategory = (renderCollapsed) => {
  let categories
  if (typeof window !== 'undefined') {
    const url = new URL(window.location.href)
    const collapsedItems = new URLSearchParams(url.search).get('collapsed')
    switch (true) {
      case !renderCollapsed && collapsedItems === null:
        categories = ['passed']
        break
      case collapsedItems?.length === 0 || /^["']{2}$/.test(collapsedItems):
        categories = []
        break
      case /^all$/.test(collapsedItems) || collapsedItems === null &&
/^all$/.test(renderCollapsed):
        categories = [...possibleFilters]
        break
      default:
        categories = collapsedItems?.split(',').map((item) =>
item.toLowerCase()) || renderCollapsed
        break
    }
  } else {
    categories = []
  }
}

```

```
    return categories;
}
```

```
const getSortDirection = () => JSON.parse(sessionStorage.getItem('sortAsc')) ||
false
const setSortDirection = (ascending) => sessionStorage.setItem('sortAsc',
ascending)
```

```
const getCollapsedIds = () => JSON.parse(sessionStorage.getItem('collapsedIds'))
|| []
const setCollapsedIds = (list) => sessionStorage.setItem('collapsedIds',
JSON.stringify(list))
```

```
module.exports = {
  getVisible,
  hideCategory,
  showCategory,
  getCollapsedIds,
  setCollapsedIds,
  getSort,
  setSort,
  getSortDirection,
  setSortDirection,
  getCollapsedCategory,
  possibleFilters,
}
```

```
}, {}, [4]);
</script>
</footer>
</html>
```