

Федеральное государственное автономное
образовательное учреждение
Высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Информатика

кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №6

Документирование кода

тема

Преподаватель

Пересунько П.В.

подпись, дата

инициалы, фамилия

Студент

КИ21-17/1Б, 032155487

номер группы, зачётной книжки

подпись, дата

Гавриленко С.И.

инициалы, фамилия

Красноярск 2021

СОДЕРЖАНИЕ

1 Цель.....	3
2 Задачи	3
3 Ход выполнения	3
4 Выводы	8
Список использованных источников	9

1 Цель

Изучить способы документирования кода на языке Python при помощи пакета Sphinx [1] и docstring [2], работу с удалённым репозиторием, а также получить необходимые навыки по оформлению README файла [3] проекта.

2 Задачи

Для выполнения практической работы необходимо выполнить следующие задачи:

- оформить docstring для функций (или методов) из практической работы №4 в любом из приглянувшихся для студента стилей;
- создать удалённый репозиторий на GitLab [4] или GitHub [5] (сервис студент выбирает по своему усмотрению) и создать там все необходимые файлы (README, gitignore и прочие, по усмотрению студента, однако советуем сначала разместить там свой проект, а только потом дополнять отсутствующими файлами);
- установить связь с удалённым репозиторием, после чего туда же загрузить все локальные коммиты;
- оформить README файл проекта с обязательным использованием примеров кода, разных заголовков, рисунков, гиперссылок (можно добавить ссылки на ресурсы, связанные с темой по варианту из практической №4);
- оформить отчет по практической работе;
- ответить на вопросы и при желании сделать задание на дополнительный балл (ответы на вопросы по этому заданию также дают дополнительный балл);

3 Ход выполнения

Вначале оформим docstring для функций (или методов) из практической работы №4. Содержание проекта представлено на Рисунке 1.

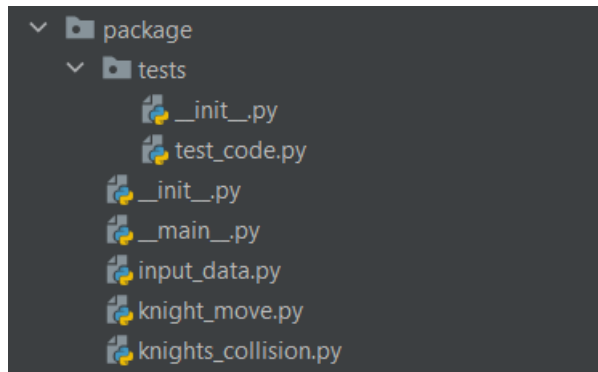


Рисунок 1 – Структура проекта

В листингах 1-4 представлена документация к созданным функциям.

Листинг 1 – Документация к __main__.py

```
"""
Основной модуль пакета, выступает в роли точки входа

Ф-я reenter - повторный ввод с возможностью завершения,
и вывода соответствующей информации в случае некорректного ввода
Ф-я main - командный линейный интерфейс (выбор режима запуска)
"""
```

Листинг 2 – Документация к input_data

```
"""
Функция ввода данных для первой и второй функций.

:return: 4 кортежа с целыми числами по 2 числа в каждом (координаты точек)
"""
```

Листинг 3 – Документация к knight_move

```
"""
Эта функция считает количество ходов, необходимое коню, чтобы
добраться из одной клетки шахматной доски до другой.

:param start: кортеж (два целых числа - начальная координата коня)
:param finish: кортеж (два целых числа - конечная координата коня)
:return: целое число (кол-во ходов, которое необходимо сделать коню)
"""
```

Листинг 4 – Документация к knights_collision

```
"""
Эта функция считает, через какое минимальное количество ходов могут
встретиться два коня, расположенных на двух разных клетках доски.

:param first: кортеж (два целых числа - координата первого коня)
:param second: кортеж (два целых числа - координата второго коня)
:return: целое число (кол-во ходов, которые должны проделать оба коня)
"""
```

У меня уже была учетная запись в GitHub. Теперь создадим удалённый репозиторий на GitHub и загрузим туда все файлы. README и gitignore файлы уже присутствовали в проекте.

```

Софья@DESKTOP-EGBA5GV MINGW64 /c/Labs/VVPD/lab6 (master)
$ git remote add master https://github.com/green1937/VVPD6.git

Софья@DESKTOP-EGBA5GV MINGW64 /c/Labs/VVPD/lab6 (master)
$ git branch -M master

Софья@DESKTOP-EGBA5GV MINGW64 /c/Labs/VVPD/lab6 (master)
$ git push -u origin master
Enumerating objects: 26, done.
Counting objects: 100% (26/26), done.
Delta compression using up to 4 threads
Compressing objects: 100% (22/22), done.
Writing objects: 100% (26/26), 7.00 KiB | 1.75 MiB/s, done.
Total 26 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/green1937/VVPD6.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

Софья@DESKTOP-EGBA5GV MINGW64 /c/Labs/VVPD/lab6 (master)
$

```

Рисунок 2 - Выгружаем на GitHub

Связь с удалённым репозиторием была успешно установлена и все локальные коммиты загрузились - рисунок 3.

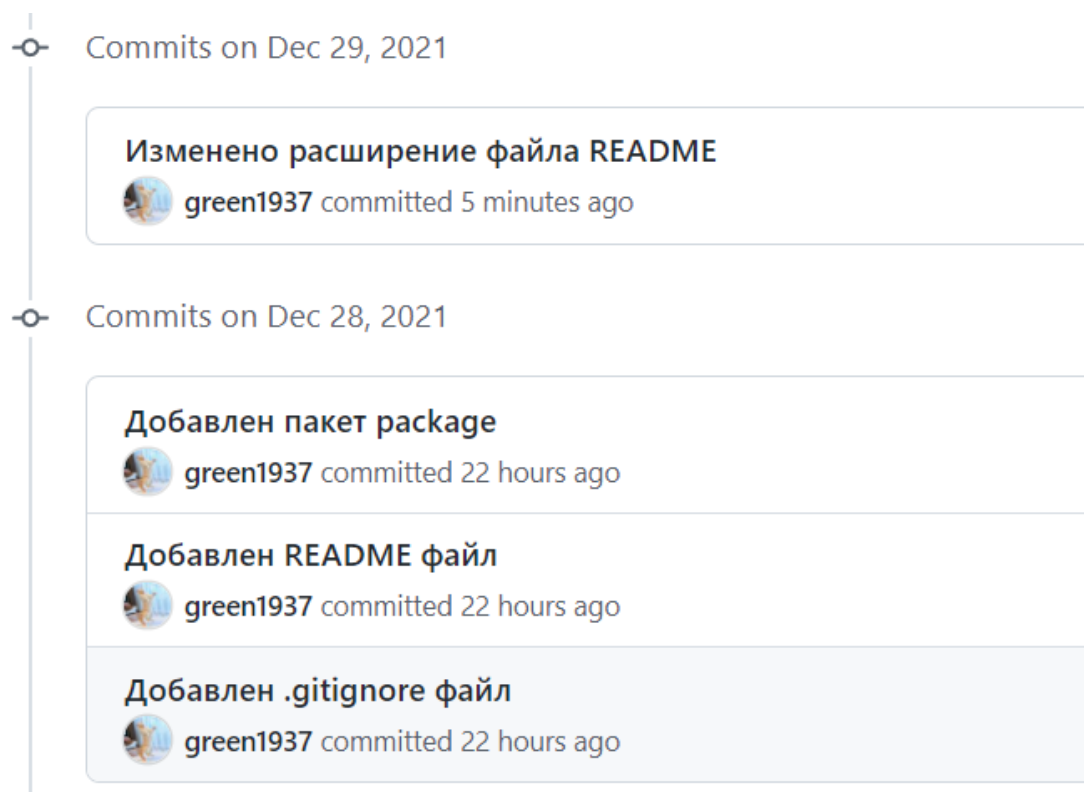


Рисунок 3 – Коммиты

Переоформим README файл проекта с обязательным использованием примеров кода, разных заголовков, рисунков – листинг 5.

Листинг 5 – README.md

```
# **Пакет, решающий задачу "Про шахматного коня"**


## **Описание задачи**
В шахматах конь является весьма уникальной фигурой, а всё из-за схемы его хода
(рисунок 10.1). За один ход конь может преодолеть маршрут напоминающий букву
«Г». Также математически доказано, что конь за некоторое количество ходов может
из любой клетки попасть в любую другую.

Функция ``knight_move`` считает количество ходов, необходимое коню, чтобы
добраться из одной клетки шахматной доски до другой.
Код функции:
```python
def knight_move(start: tuple[int, int], finish: tuple[int, int]) -> int:
 step = [[100 for _ in range(0,13)] for _ in range(0,13)]
 step[start[0]+1][start[1]+1] = 0
 for _ in range(0,9):
 for i in range(2, 10):
 for j in range(2, 10):
 if step[i][j] > 0:
 step[i][j] = min(step[i+2][j+1], step[i-2][j-1],
 step[i-1][j-2], step[i+1][j+2],
 step[i+2][j-1], step[i-2][j+1],
 step[i+1][j-2], step[i-1][j+2])+1
 answer = step[finish[0]+1][finish[1]+1]
 return answer
```

### **Пример 1**
При (1, 1), (6, 1) - количество ходов равно 3, а при (2, 2), (2, 2) - количество
ходов равно 1.

Функция ``knights_collision`` считает, через какое минимальное количество
ходов могут встретиться два коня, расположенных на двух разных клетках доски.
Код функции:
```python
def knights_collision(first: tuple[int, int], second: tuple[int, int]) -> int:
 step = [[100 for _ in range(0, 13)] for _ in range(0, 13)]
 step[first[0]+1][first[1]+1] = 0
 for _ in range(0, 9):
 for i in range(2, 10):
 for j in range(2, 10):
 if step[i][j] > 0:
 step[i][j] = min(step[i+2][j+1], step[i-2][j-1],
 step[i-1][j-2], step[i+1][j+2],
 step[i+2][j-1], step[i-2][j+1],
 step[i+1][j-2], step[i-1][j+2])+1
 answer = step[second[0]+1][second[1]+1]
 sum1 = first[0]+first[1]
 sum2 = second[0] + second[1]
 if (sum1 % 2 == 0 and sum2 % 2 == 0) or (sum1 % 2 == 1 and sum2 % 2 == 1):
 return answer // 2
 else:
 answer = 0
```

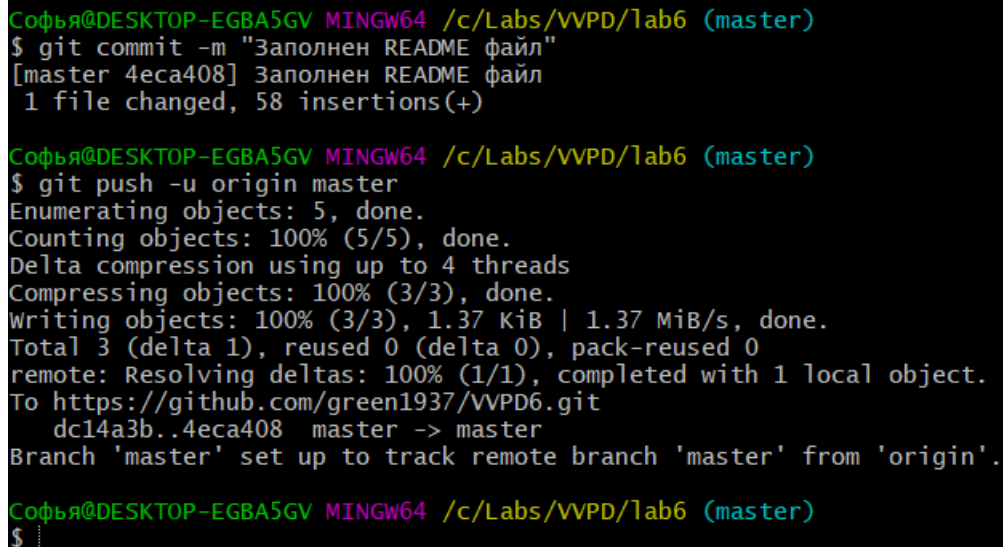
## Продолжение листинга 5

```
... return answer
```

```
Пример 2
```

При (1, 2), (3, 4) – количество ходов равно 2, а при (1, 2), (4, 4) – кони никогда не встретятся.

Теперь загрузим новый README файл в наш удаленный репозиторий.



```
Софья@DESKTOP-EGBA5GV MINGW64 /c/Labs/VVPD/lab6 (master)
$ git commit -m "Заполнен README файл"
[master 4eca408] Заполнен README файл
1 file changed, 58 insertions(+)

Софья@DESKTOP-EGBA5GV MINGW64 /c/Labs/VVPD/lab6 (master)
$ git push -u origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.37 KiB | 1.37 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/green1937/VVPD6.git
 dc14a3b..4eca408 master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

Софья@DESKTOP-EGBA5GV MINGW64 /c/Labs/VVPD/lab6 (master)
$
```

Рисунок 3 - Загрузка README.md

## **4 Выводы**

В ходе работы были изучены способы документирования кода на языке Python при помощи пакета Sphinx [1] и docstring [2], работы с удалённым репозиторием, а также получены необходимые навыки по оформлению README файла [3] проекта.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Sphinx 4.0.0. Python Documentation Generator [Электронный ресурс]: official website of tool that makes it easy to create intelligent and beautiful documentation. – 2020. – Режим доступа: <https://www.sphinxdoc.org/en/master/index.html>.
2. PEP 257 – Docstring Conventions [Электронный ресурс]: Python developer guide. PEP Index. – 2020. – Режим доступа: <https://www.python.org/dev/peps/pep-0257/>.
3. GitLab Markdown – the Markdown guide [Электронный ресурс]: GitLab user docs. – 2020. – Режим доступа: <https://docs.gitlab.com/ee/user/markdown.html>.
4. GitLab [Электронный ресурс]: official GitLab documentation. – 2020. – Режим доступа: <https://docs.gitlab.com/>.
5. GitHub [Электронный ресурс]: official GitHub documentation. – 2020. – Режим доступа: <https://docs.github.com/en>.
6. GitLab Pages [Электронный ресурс]: official GitLab documentation. – 2020. – Режим доступа: <https://docs.gitlab.com/ee/user/project/pages/>.
7. GitHub Pages [Электронный ресурс]: official GitHub documentation. – 2020. – Режим доступа: <https://pages.github.com/>
8. СТО 4.2-07-2014 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности. – Взамен СТО 4.2-07-2012; введ. 30.12.2013. – Красноярск: СФУ, 2014. – 60с.