Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №5
дисциплины «Программирование на Python»
Вариант 23

Выполнил:
Мотовилов Вадим Борисович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Информатика и вычислительная
техника», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Роман Александрович
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Порядок выполнения работы:

1. Создал репозиторий и скопировал его

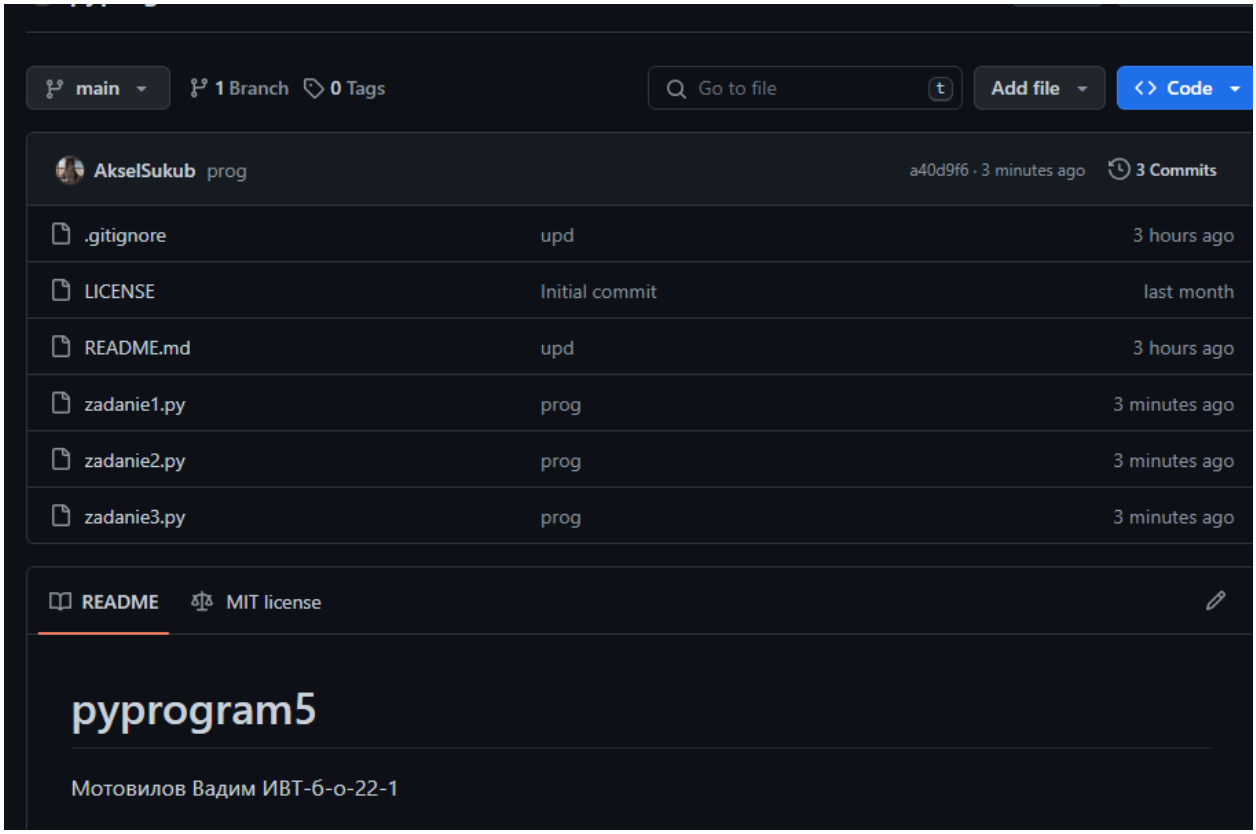

Рисунок 1. Созданный репозиторий

2. Изменил файл .gitignore и README.rm и добавил git flow

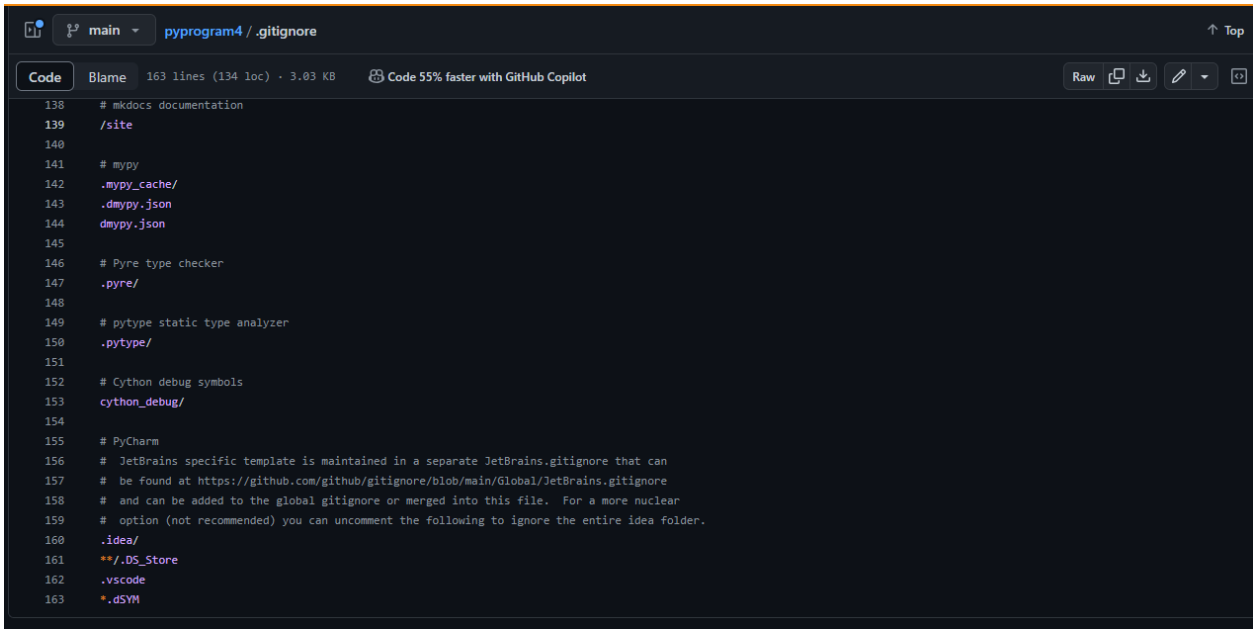

Рисунок 2. Измененный файл .gitignore

3. Выполнил задания

Вводится число карандашей $N \leq 10$. Вывести фразу `Я купил N карандашей`, согласовав слово "карандаш" с числом N.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


def main():
    n = int(input("Введите число карандашей (N <= 10): "))

    if n <= 0 or n > 10:
        print("Пожалуйста, введите число от 1 до 10.")
    else:
        if n == 1:
            print(f"Я купил {n} карандаш.")
        elif 2 <= n <= 4:
            print(f"Я купил {n} карандаша.")
        else:
            print(f"Я купил {n} карандашей.")

if __name__ == "__main__":
    main()
```

```
Введите число карандашей (N <= 10): 3
Я купил 3 карандаша.
```

Рисунок 3 . Выполнение 1 задания

Симметричны ли точки $M_1(x_1, y_1)$ и $M_2(x_2, y_2)$ относительно оси ox или относительно оси oy?

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


def are_points_symmetric(m1, m2):
    x1, y1 = m1
    x2, y2 = m2

    symmetric_x = x1 == x2
    symmetric_y = y1 == y2

    if symmetric_x and symmetric_y:
        return "Точки симметричны относительно обеих осей."
    elif symmetric_x:
        return "Точки симметричны относительно оси Oy."
    elif symmetric_y:
        return "Точки симметричны относительно оси Ox."
    else:
        return "Точки не симметричны относительно осей."

def main():
    m1 = tuple(map(float, input("Введите координаты точки M1 (x1 y1): ").split()))
    m2 = tuple(map(float, input("Введите координаты точки M2 (x2 y2): ").split()))

    result = are_points_symmetric(m1, m2)
    print(result)

if __name__ == "__main__":
    main()
```

```
Введите координаты точки M1 (x1 y1): 3 5
Введите координаты точки M2 (x2 y2): 3 6
Точки симметричны относительно оси Oy.
```

Рисунок 4 . Выполнение 2 задания

Найти все трехзначные натуральные числа, сумма цифр которых равна их произведению.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


def find_special_numbers():
    special_numbers = []

    for number in range(100, 1000):
        digit_sum = sum(int(digit) for digit in str(number))
        digit_product = 1
        for digit in str(number):
            digit_product *= int(digit)

        if digit_sum == digit_product:
            special_numbers.append(number)

    return special_numbers

def main():
    result = find_special_numbers()

    if result:
        print("Трехзначные натуральные числа, сумма цифр которых равна их произведению:")
        print(result)
    else:
        print("Таких чисел нет.")

if __name__ == "__main__":
    main()
```

```
Трехзначные натуральные числа, сумма цифр которых равна их произведению:
[123, 132, 213, 231, 312, 321]
```

Рисунок 5 . Выполнение 3 задания

Ответы на контрольные вопросы:

1. 1. **Для чего нужны диаграммы деятельности UML?** Диаграммы деятельности в UML используются для визуализации и описания бизнес-процессов и деятельности системы. Они позволяют моделировать последовательность действий, участников и потоки управления в процессе.

2. **Что такое состояние действия и состояние деятельности?**

- *Состояние действия* - это состояние, в котором объект выполняет какое-то действие, и оно представляет собой шаг или этап в деятельности.

- *Состояние деятельности* - это более общее состояние, описывающее область, в которой объект может находиться в течение определенного периода.

3. **Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?**
В диаграммах деятельности используются стрелки для обозначения переходов. Для ветвлений используются ромбы с условиями.

4. **Какой алгоритм является алгоритмом разветвляющейся структуры?**
Алгоритм разветвляющейся структуры - это условный оператор (if-else).

5. **Чем отличается разветвляющийся алгоритм от линейного?**
Разветвляющийся алгоритм содержит условия и ветвления, позволяющие выбирать различные пути выполнения программы в зависимости от условий. Линейный алгоритм выполняется последовательно без ветвлений.

6. **Что такое условный оператор? Какие существуют его формы?**
Условный оператор (if-else) позволяет выполнять определенные блоки кода в зависимости от условия. Есть формы:

- Простой условный оператор: if условие: блок_кода
- Условный оператор с ветвлением: if условие: блок_кода1 else: блок_кода2

7. **Какие операторы сравнения используются в Python?**
Операторы сравнения: == (равно), != (не равно), < (меньше), > (больше), <= (меньше или равно), >= (больше или равно).

8. **Что называется простым условием? Приведите примеры.**
Простое условие - это выражение, возвращающее булево значение. Пример: x > 0.

9. **Что такое составное условие? Приведите примеры.**
Составное условие - это комбинация нескольких простых условий с использованием логических операторов. Пример: (x > 0) and (y < 10).

10. **Какие логические операторы допускаются при составлении сложных условий?**
Логические операторы: and (логическое И), or (логическое ИЛИ), not (логическое НЕ).

11. **Может ли оператор ветвления содержать внутри себя другие ветвления?**
Да, оператор ветвления может содержать внутри себя другие ветвления, создавая вложенные условия.

12. **Какой алгоритм является алгоритмом циклической структуры?**
Алгоритм циклической структуры - это цикл, например, цикл while или for.

13. **Типы циклов в языке Python.**
В Python есть циклы for и while.

14. **Назовите назначение и способы применения функции range.**
Функция range создает последовательность чисел и часто используется в циклах. Пример: range(10) создает последовательность от 0 до 9.

15. **Как с помощью функции range организовать перебор значений от 15 до 0 с шагом 2?**
range(15, -1, -2)

16. **Могут ли быть циклы вложенными?**
Да, циклы могут быть вложенными, т.е. один цикл может находиться внутри другого.

17. **Как образуется бесконечный цикл и как выйти из него?**
Бесконечный цикл может возникнуть, если условие цикла всегда истинно. Для выхода используется оператор break.

18. **Для чего нужен оператор break?**

Оператор break используется для выхода из цикла досрочно, прерывая его выполнение.

19. **Где употребляется оператор continue и для чего он используется?**

Оператор continue используется внутри цикла для перехода к следующей итерации, пропуская оставшуюся часть кода в текущей итерации.

20. **Для чего нужны стандартные потоки stdout и stderr?**

21. **Как в Python организовать вывод в стандартный поток stderr?**

В Python можно организовать вывод в стандартный поток ошибок (stderr) с помощью модуля sys.

22. **Каково назначение функции exit?**

Функция exit используется для немедленного завершения программы. Она прерывает выполнение программы и возвращает код завершения.