

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**
**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №2.2

Дисциплина: «Основы кроссплатформенного программирования»

Тема: «Условные операторы и циклы в языке Python»

Выполнил: студент 2 курса
группы ПИЖ-б-о-21-1
Пуценко Иван Алексеевич

Ставрополь 2022

Выполнение работы.

1. Создал репозиторий в GitHub «rep 2.2» в который добавил .gitignore, который дополнил правила для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на лок. сервер и организовал в соответствие с моделью ветвления git-flow.

2. Создал проект PyCharm в папке репозитория, проработал примеры ЛР.

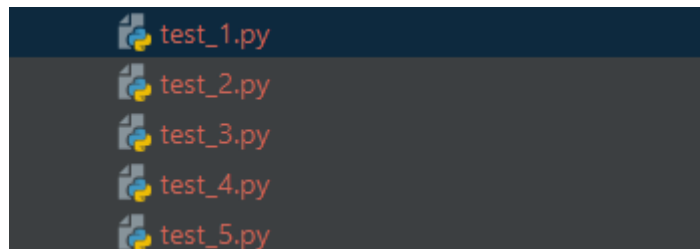


Рисунок 2.1 Примеры в проекте

3. Построил диаграммы к примерам

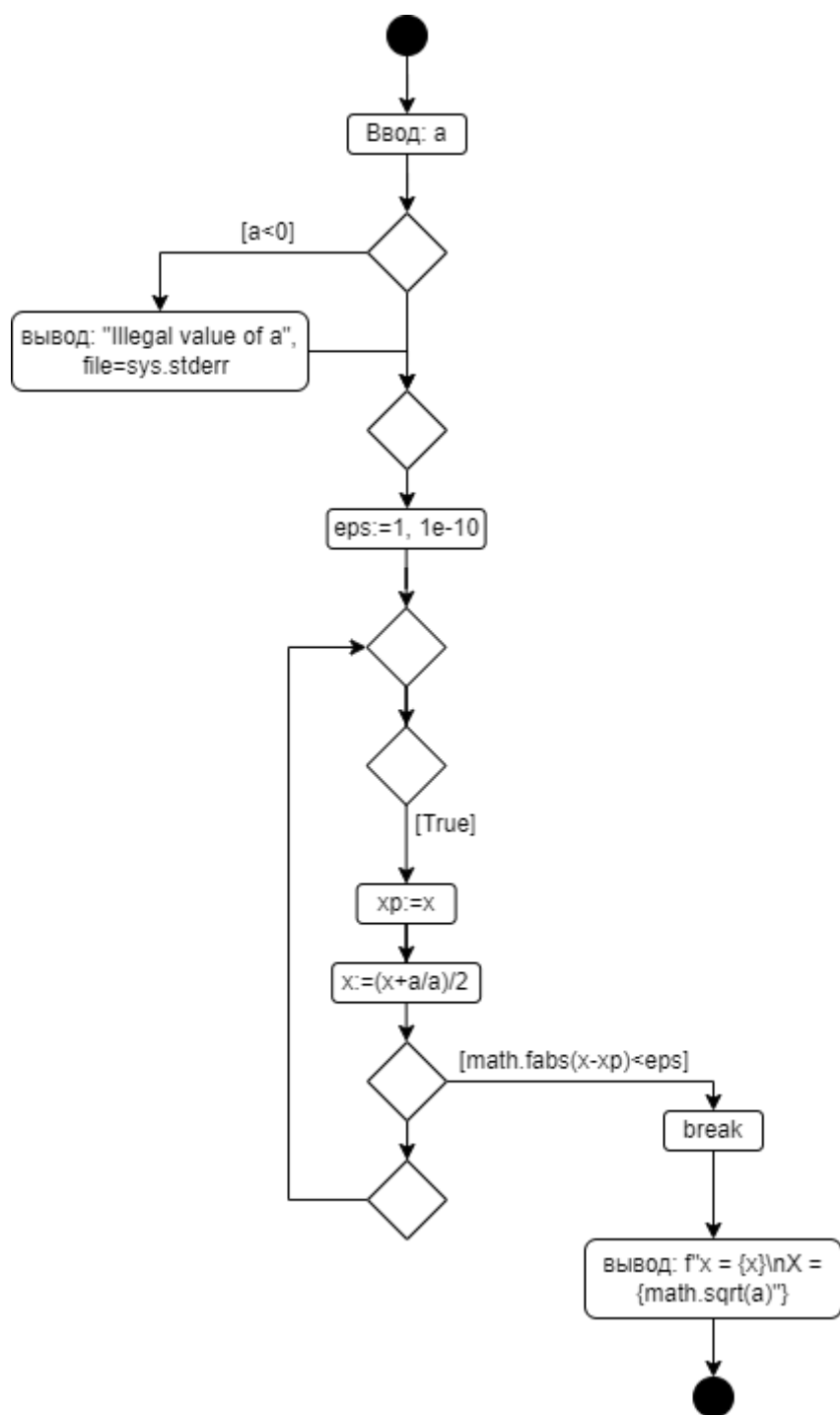


Рисунок 3.1 UML – диаграмма к примеру №4

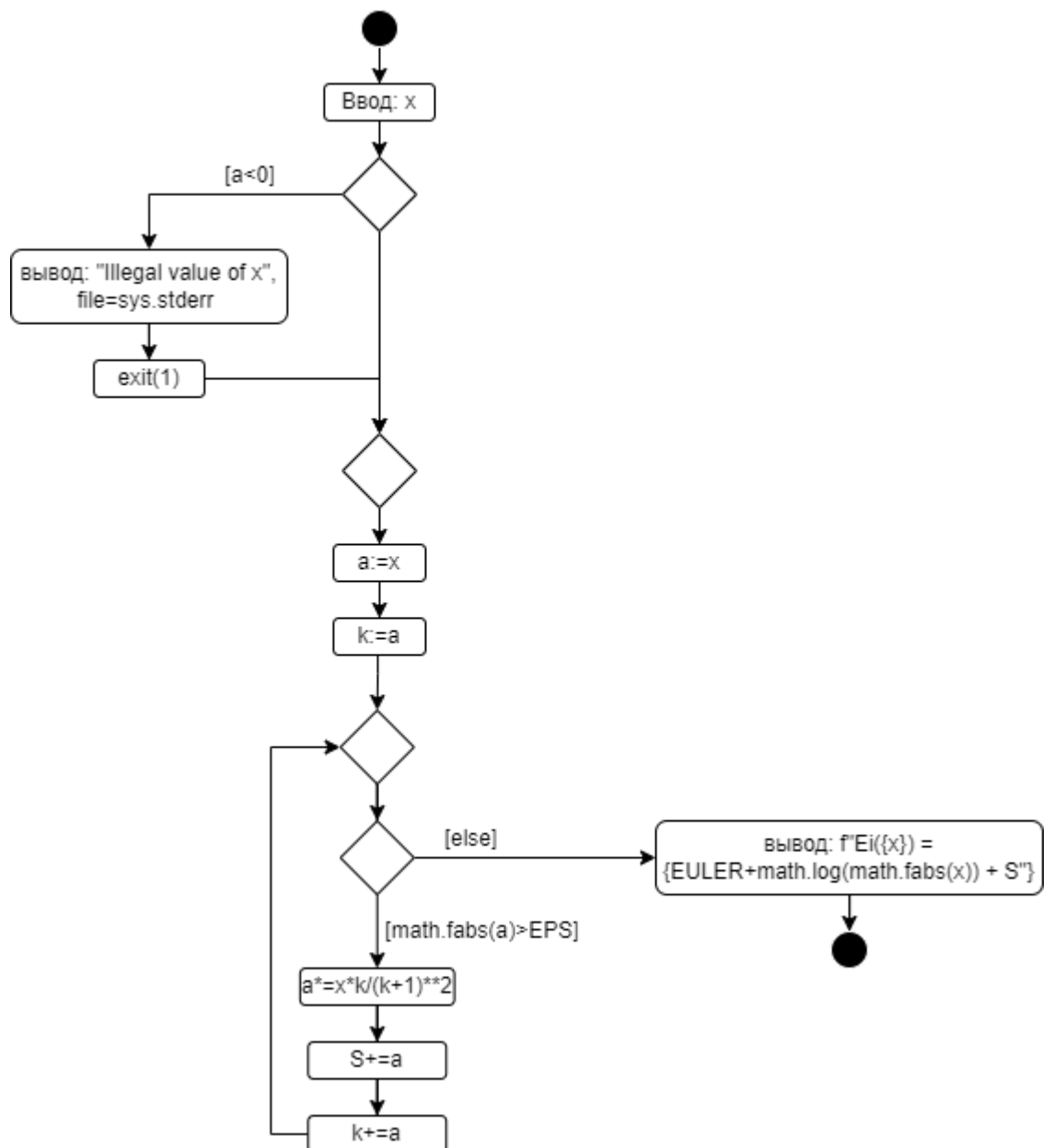


Рисунок 3.2 UML – диаграмма к примеру №5

4. Выполнил индивидуальные задания и задание повышенной сложности согласно своему варианту. Построил UML диаграммы программ.

The image shows a Python IDE with a file explorer on the left and a code editor on the right. The file explorer shows a directory named 'labRab_5' containing files like 'diagrams', '.gitignore', 'Hard_task.py', 'LICENSE', 'README.md', 'task_1.py', 'task_2.py', 'task_3.py', 'test_1.py', 'test_2.py', 'test_3.py', 'test_4.py', 'test_5.py', 'External Libraries', and 'Scratches and Consoles'. The code editor displays a Python script for task 1. The script prompts the user for the number of pencils bought and prints the corresponding phrase based on the count. The console at the bottom shows the execution of 'task_1.py', where the user entered '4' and the program outputted 'я купил 4 карандаша'.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 # Вводится число карандашей. Вывести фразу Я купил N карандашей, согласовав слов
5
6 if __name__ == '__main__':
7     N = int(input("сколько карандашей вы купили? "))
8     if N == 1:
9         print("я купил", N, "карандаш")
10    elif (N >= 2) and (N <= 4):
11        print("я купил", N, "карандаша")
12    elif (N >= 5) and (N <= 10):
13        print("я купил", N, "карандашей")
14    else:
15        print("неверно введенное число")
16
```

if __name__ == '__main__': if N == 1

test_1 x task_1 x

C:\Users\FonK\AppData\Local\Programs\Python\Python39\python.exe C:/Users/FonK/Desktop/ОПИ/LabRab_5/task_1.py

сколько карандашей вы купили? 4

я купил 4 карандаша

Process finished with exit code 0

Рисунок 4.1 Программа к инд. заданию №1

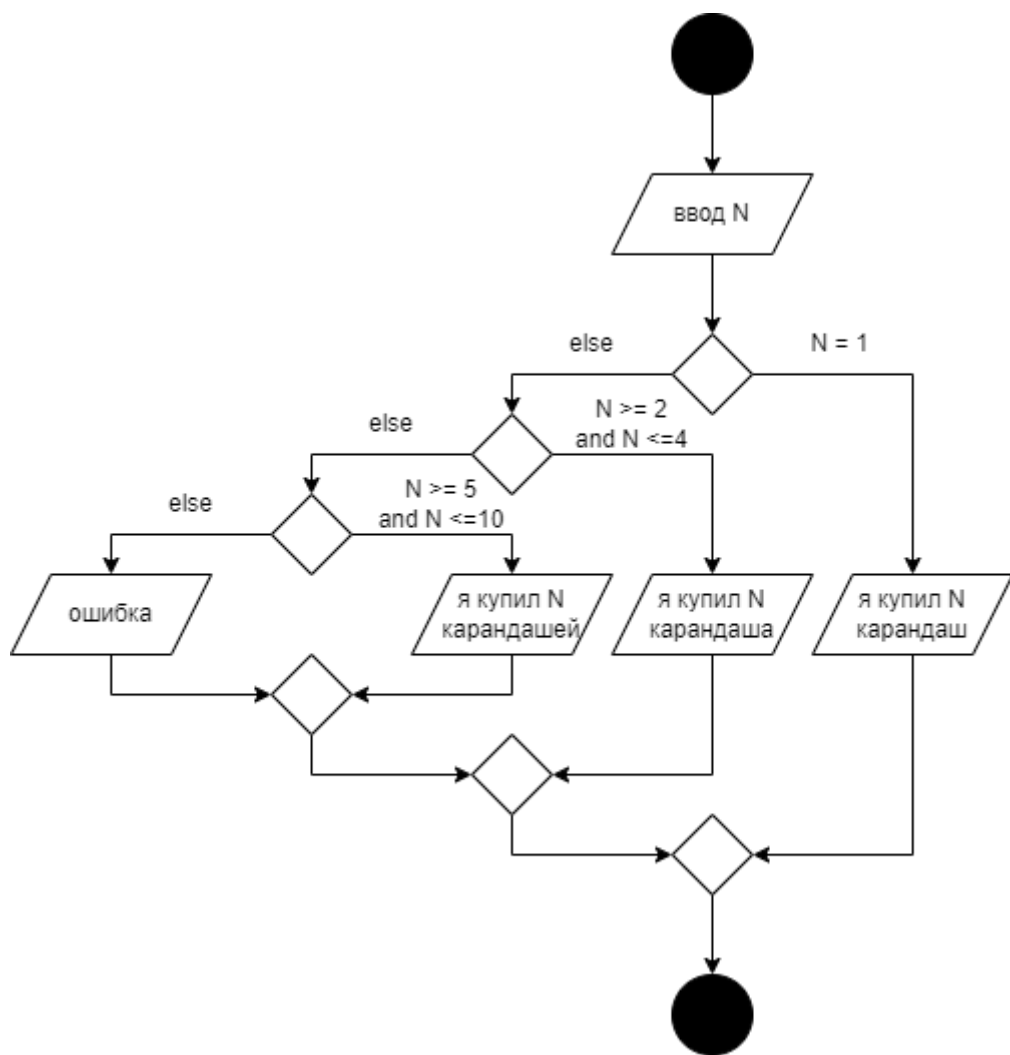


Рисунок 4.2 UML – диаграмма к программе инд. задания 1

```
labRab_5 C:\Users\FonK\Desktop\ОПИ\labRab_5
> diagrams
> .gitignore
> Hard_task.py
> LICENSE
> README.md
> task_1.py
> task_2.py
> task_3.py
> test_1.py
> test_2.py
> test_3.py
> test_4.py
> test_5.py
> External Libraries
> Scratches and Consoles

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 # Симметричны ли точки M1(x1, y1) и M2(x2, y2) относительно оси Ox или отно
5 if __name__ == '__main__':
6     x1 = int(input("введите координаты точки M1 \n"))
7     y1 = int(input())
8     x2 = int(input("введите координаты точки M2 \n"))
9     y2 = int(input())
10
11     if x1 == x2 and y1 == -y2:
12         print("точки симметричны оси Ox")
13     else:
14         print("точки не симметричны оси Ox")
15     if x1 == -x2 and y1 == y2:
16         print("точки симметричны оси Oy")
17     else:
18         print("точки не симметричны оси Oy")
19
```

test_1 x task_2 x

C:\Users\FonK\AppData\Local\Programs\Python\Python39\python.exe C:/Users/FonK/Desktop/ОПИ/labRab_5/task_2.py

введите координаты точки M1

3

-3

введите координаты точки M2

2

2

точки не симметричны оси Ox

точки не симметричны оси Oy

Process finished with exit code 0

Рисунок 4.3 Программа к инд. заданию №2

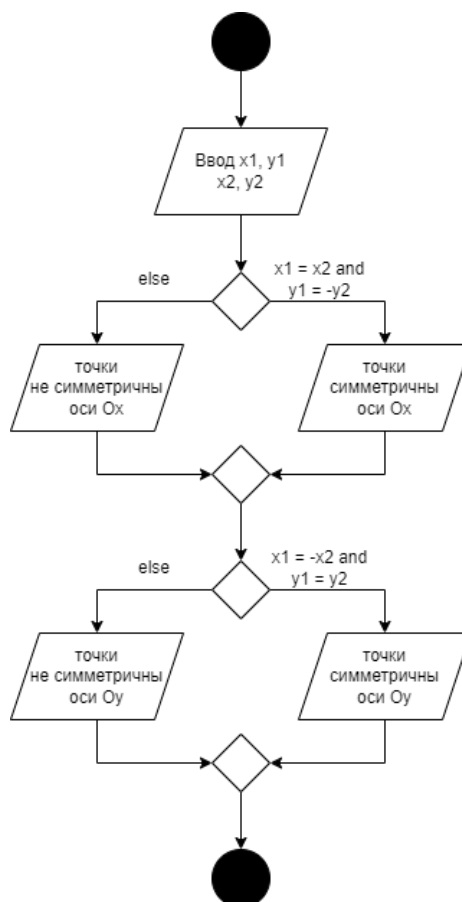


Рисунок 4.4 UML – диаграмма к программе инд. задания 2

The screenshot shows a Python IDE with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'labRab_5' with various files including 'task_3.py'. The code editor displays the following Python code:

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Найти все трехзначные натуральные числа, сумма цифр которых равна их произведению
5  if __name__ == '__main__':
6      for i in range(100, 1000):
7          _sum = 0
8          _mul = 1
9          num = i
10         while num != 0:
11             _sum += num % 10
12             _mul *= num % 10
13             num //= 10
14         if _sum == _mul:
15             print(i)
16

```

Below the code editor, the execution output is shown in a console window. It lists the numbers 123, 132, 213, 231, 312, and 321, which are the three-digit numbers whose sum of digits equals their product. The console also shows the command used to run the script and the exit code 0.

Рисунок 4.5 Программа к инд. заданию №1

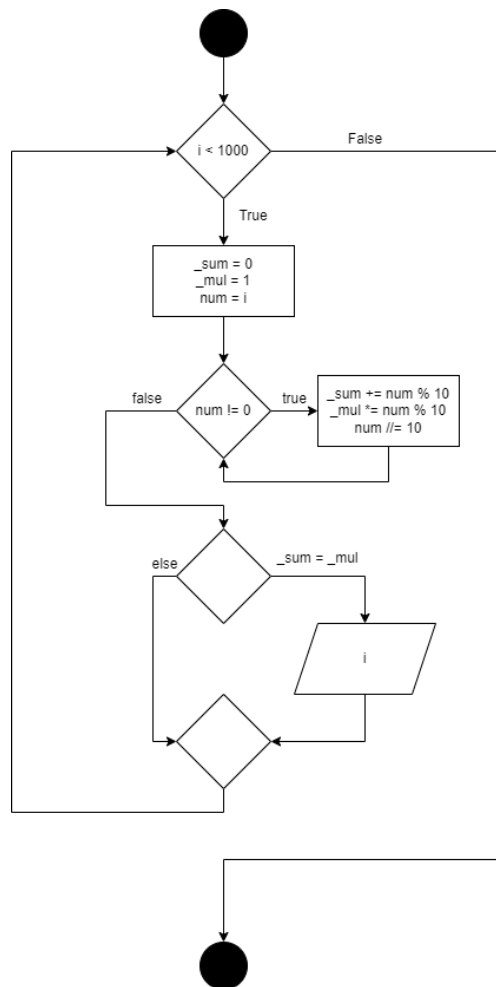


Рисунок 4.6 UML – диаграмма к программе инд. задания 3


```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import math
4  import sys
5
6  EPS = 1e-10
7
8  # вариант 5
9  if __name__ == '__main__':
10     x = float(input("введите x: "))
11     if x == 0:
12         print("неверно введенное x", file=sys.stderr)
13         exit(1)
14
15     a = -((math.pi ** 2) / 40)
16     S, n = a, 1
17
18     while math.fabs(a) < EPS:
19
20         a = -((math.pi ** 2) * (4 * n - 1)) / 4 * (2 * n + 2) * (2 * n + 1) * (4 * n + 5)
21         S += a
22         n += 1
23
24     print(S)
25
26 if __name__ == '__main__':

```

Hard_task (1) ×

C:\Users\FonK\AppData\Local\Programs\Python\Python39\python.exe C:\Users\FonK\Desktop\ОПИ\LabRab_5/task/Hard_task.py

введите x:
 -0.24674011002723395

Process finished with exit code 0

Рисунок 4.7 Программа для задачи повышенной сложности.

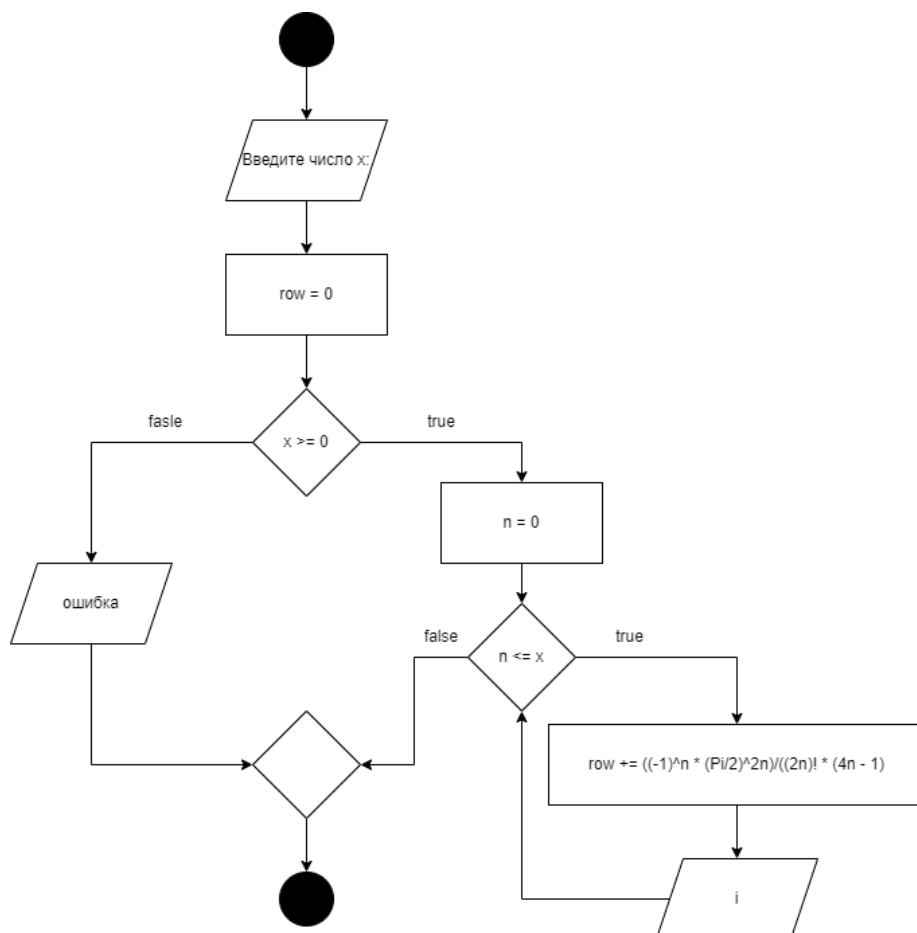


Рисунок 4.8 UML – диаграмма деятельности программы для усложненного задания

5. Сделал коммит, выполнил слияние с веткой main, и запустил изменения в уд. репозиторий.

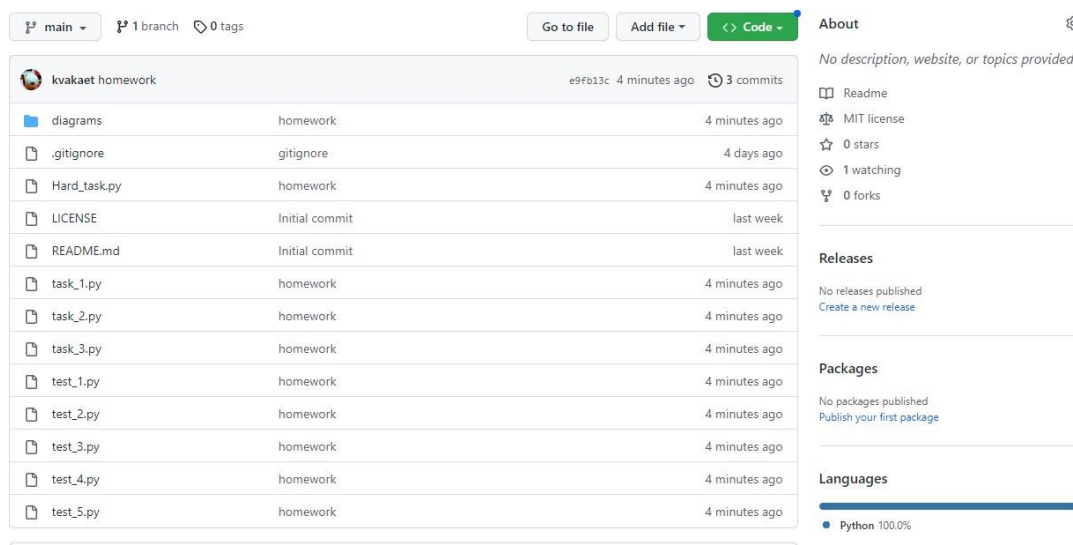


Рисунок 5.1 Изменения на уд. сервере

1. Для чего нужны диаграммы деятельности UML?

Позволяет наглядно визуализировать алгоритм программы.

2. Что такое состояние действия и состояние деятельности?

Состояние действия - частный вид состояния деятельности, а конкретнее – такое состояние, которое не может быть подвергнуто дальнейшей декомпозиции.

Состояние деятельности можно представлять себе как составное состояние, поток управления которого включает только другие состояния деятельности и действий.

3. Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?

Переходы, ветвление, алгоритм разветвляющейся структуры, алгоритм циклической структуры.

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм разветвляющейся структуры - это алгоритм, в котором вычислительный процесс осуществляется либо по одной, либо по другой ветви, в зависимости от выполнения некоторого условия.

5. Чем отличается разветвляющийся алгоритм от линейного?

Линейный алгоритм - алгоритм, все этапы которого выполняются однократно и строго последовательно.

Разветвляющийся алгоритм - алгоритм, содержащий хотя бы одно условие, в результате проверки которого ЭВМ обеспечивает переход на один из нескольких возможных шагов.

6. Что такое условный оператор? Какие существуют его формы?

Оператор, конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд.

Условный оператор имеет полную и краткую формы.

7. Какие операторы сравнения используются в Python?

If, elif, else

8. Что называется простым условием? Приведите примеры.

Простым условием называется выражение, составленное из двух арифметических выражений или двух текстовых величин.

Пример: `a == b`

9. Что такое составное условие? Приведите примеры.

Составное условие – логическое выражение, содержащее несколько простых условий объединённых логическими операциями. Это операции `not`, `and`, `or`.

Пример: `(a == b or a == c)`

10. Какие логические операторы допускаются при составлении сложных условий?

`not`, `and`, `or`.

11. Может ли оператор ветвления содержать внутри себя другие ветвления?

Может.

12. Какой алгоритм является алгоритмом циклической структуры?

Циклический алгоритм — это вид алгоритма, в процессе выполнения которого одно или несколько действий нужно повторить.

13. Типы циклов в языке Python.

В Python есть 2 типа циклов: - цикл while, - цикл for.

14. Назовите назначение и способы применения функции range.

Функция range генерирует серию целых чисел, от значения start до stop, указанного пользователем. Мы можем использовать его для цикла for и обходить весь диапазон как список.

15. Как с помощью функции range организовать перебор значений от 15 до 0 с шагом 2?

```
range(15, 0, 2)
```

16. Могут ли быть циклы вложенными?

Могут.

17. Как образуется бесконечный цикл и как выйти из него?

Бесконечный цикл в программировании — цикл, написанный таким образом, что условие выхода из него никогда не выполняется.

18. Для чего нужен оператор break?

Используется для выхода из цикла.

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

Оператор continue используется только в циклах. В операторах for , while , do while , оператор continue выполняет пропуск оставшейся части кода тела цикла и переходит к следующей итерации цикла.

20. Для чего нужны стандартные потоки stdout и stderr?

Ввод и вывод распределяется между тремя стандартными потоками: stdin — стандартный ввод (клавиатура), stdout — стандартный вывод (экран), stderr — стандартная ошибка (вывод ошибок на экран)

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

Указать в `print(..., file=sys.stderr)`.

22. Каково назначение функции `exit`?

Функция `exit()` модуля `sys` - выход из Python.