**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 1 курса
группы ИВТ-б-о-21-1
Харченко Богдан Романович

Ставрополь 2022

**Ход работы:**

1. Создал репозиторий в GitHub «Lab_4» 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.

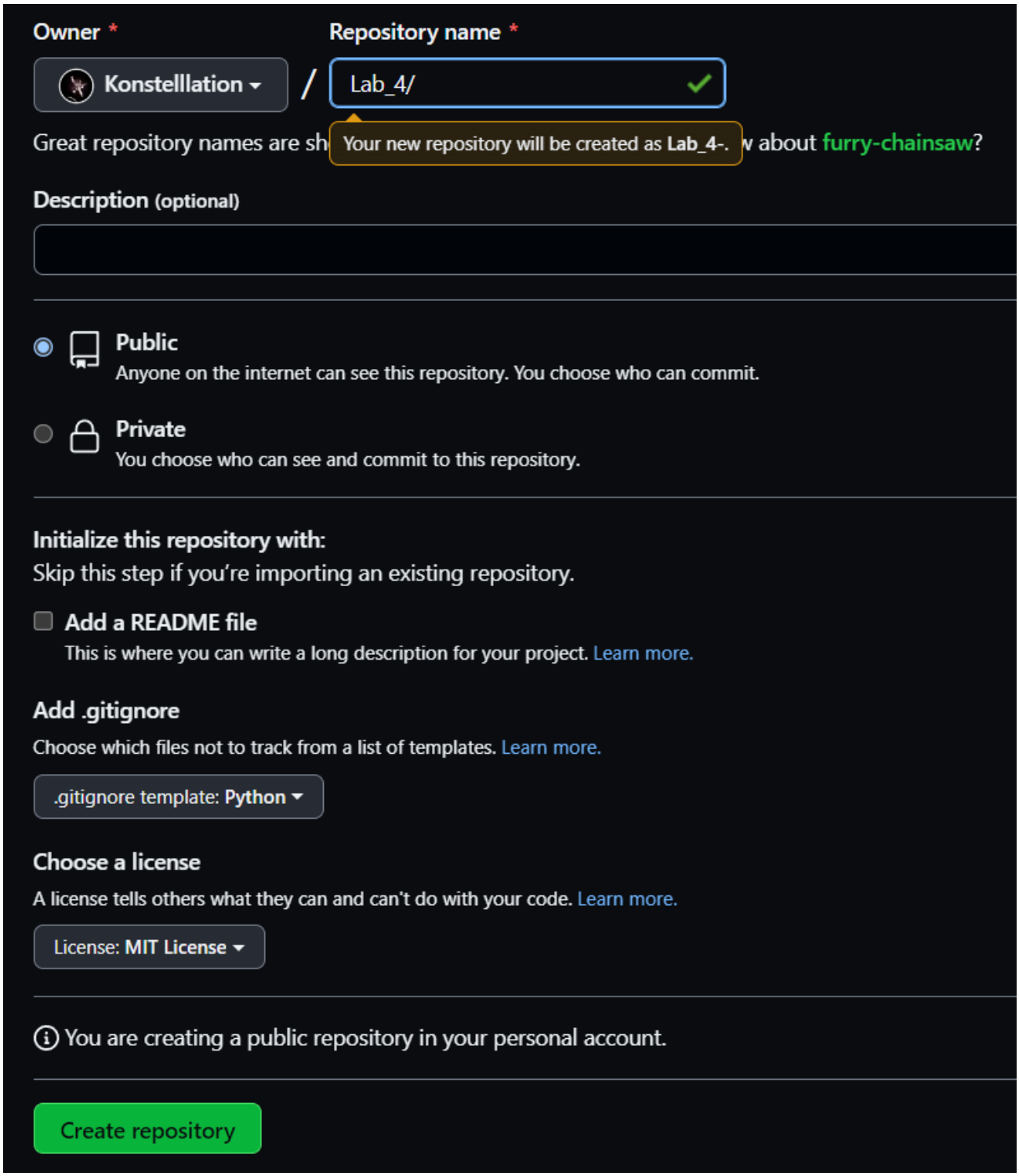


Рисунок 1.1 Создание репозитория

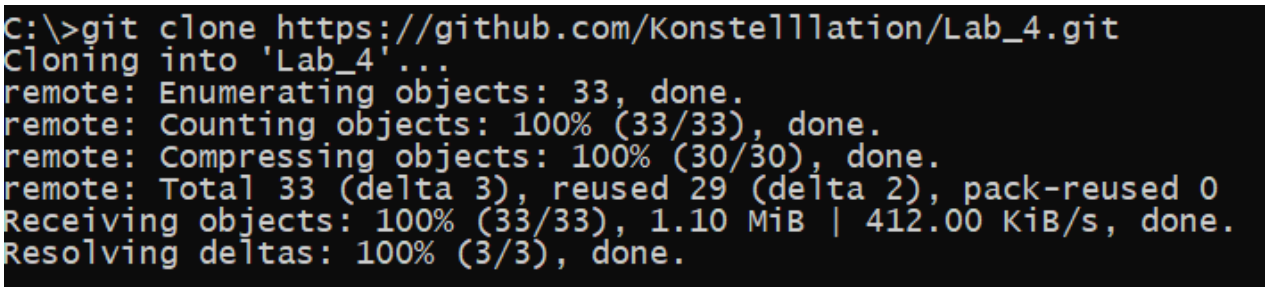
C:\>git clone https://github.com/Konstelllation/Lab_4.git
Cloning into 'Lab_4'...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 33 (delta 3), reused 29 (delta 2), pack-reused 0
Receiving objects: 100% (33/33), 1.10 MiB | 412.00 KiB/s, done.
Resolving deltas: 100% (3/3), done.

Рисунок 1.2 Клонирование репозитория

```
C:\LB_4>git flow init

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/LB_4/.git/hooks]
```

Рисунок 1.3 Организация репозитория в соответствии с моделью ветвления git-flow


Рисунок 1.4 Изменение .gitignore

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

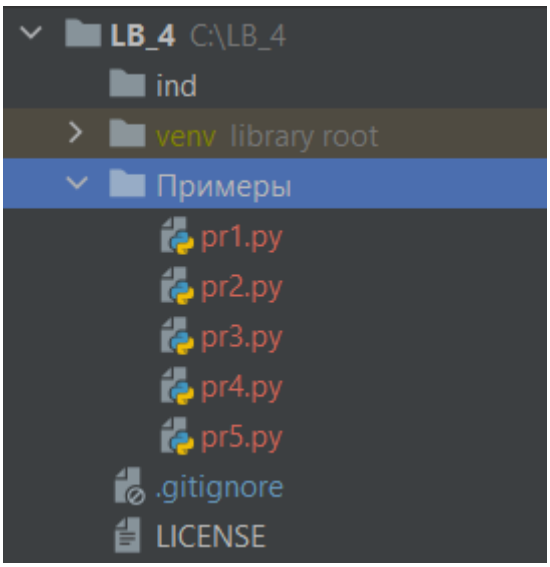


Рисунок 2.1 Примеры в проекте

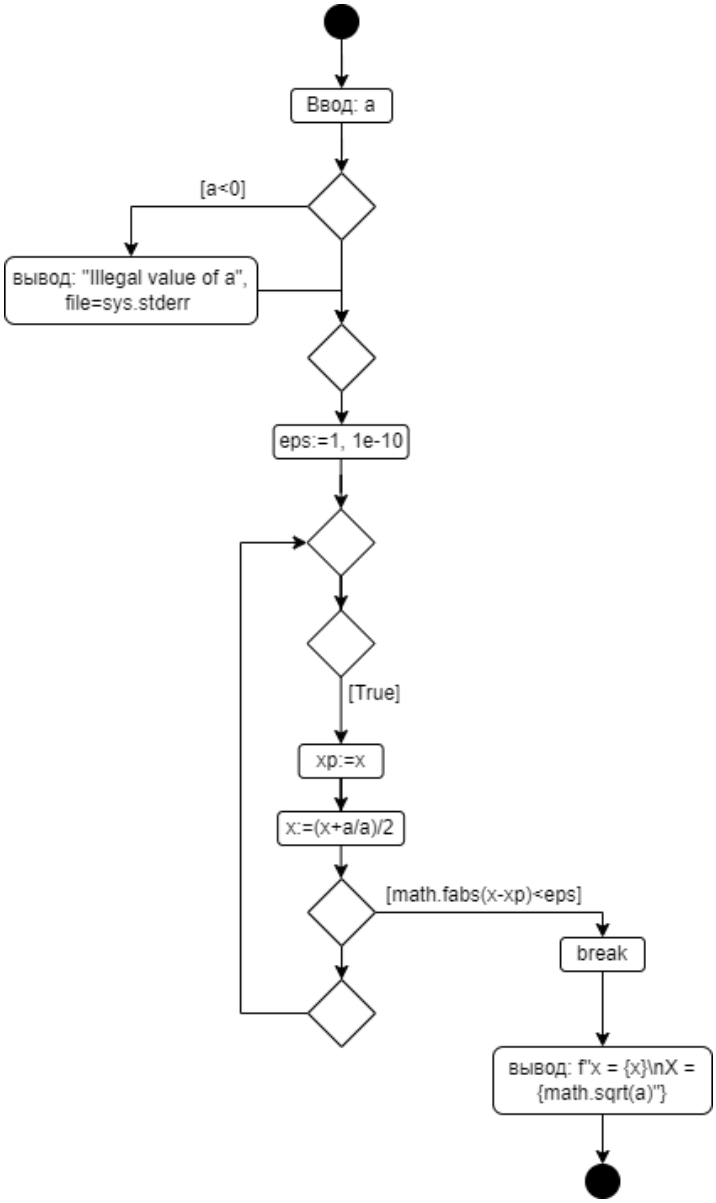


Рисунок 2.2 UML-диаграмма программы 4 примера

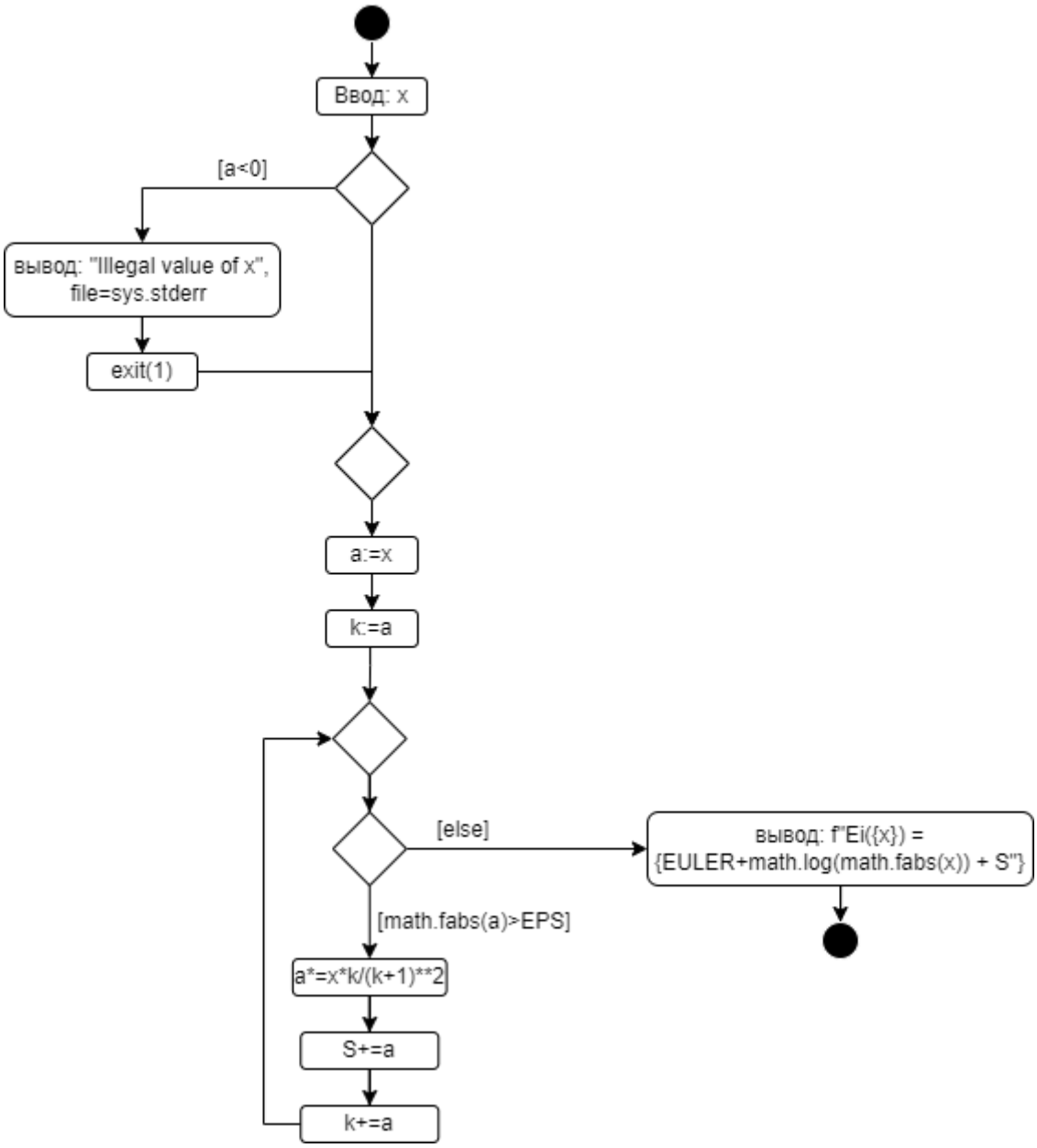


Рисунок 2.3 UML-диаграмма программы 5 примера

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

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

if __name__ == '__main__':
    # Ввод номера месяца
    m = int(input('Введите номер месяца: '))

    # Проверка на полугодие
    if m <= 6:
        print('Месяц приходится на первое полугодие.\n')
    else:
        print('Месяц приходится на второе полугодие.\n')

    # Провека на количество дней в месяце
    if m == 2:
        print('В месяце под номером ', m, ' 28 дней.')
    elif m == 1 or m == 3 or m == 5 or m == 7 or m == 8 or m == 10 or m == 12:
        print('В месяце под номером ', m, ' 31 дней.')
    else:
        print('В месяце под номером ', m, ' 30 дней.')

```

```
Введите номер месяца: 5
Месяц приходится на первое полугодие.

В месяце под номером  5  31 дней.
```

Рисунок 3.1 Программа к инд. заданию №1

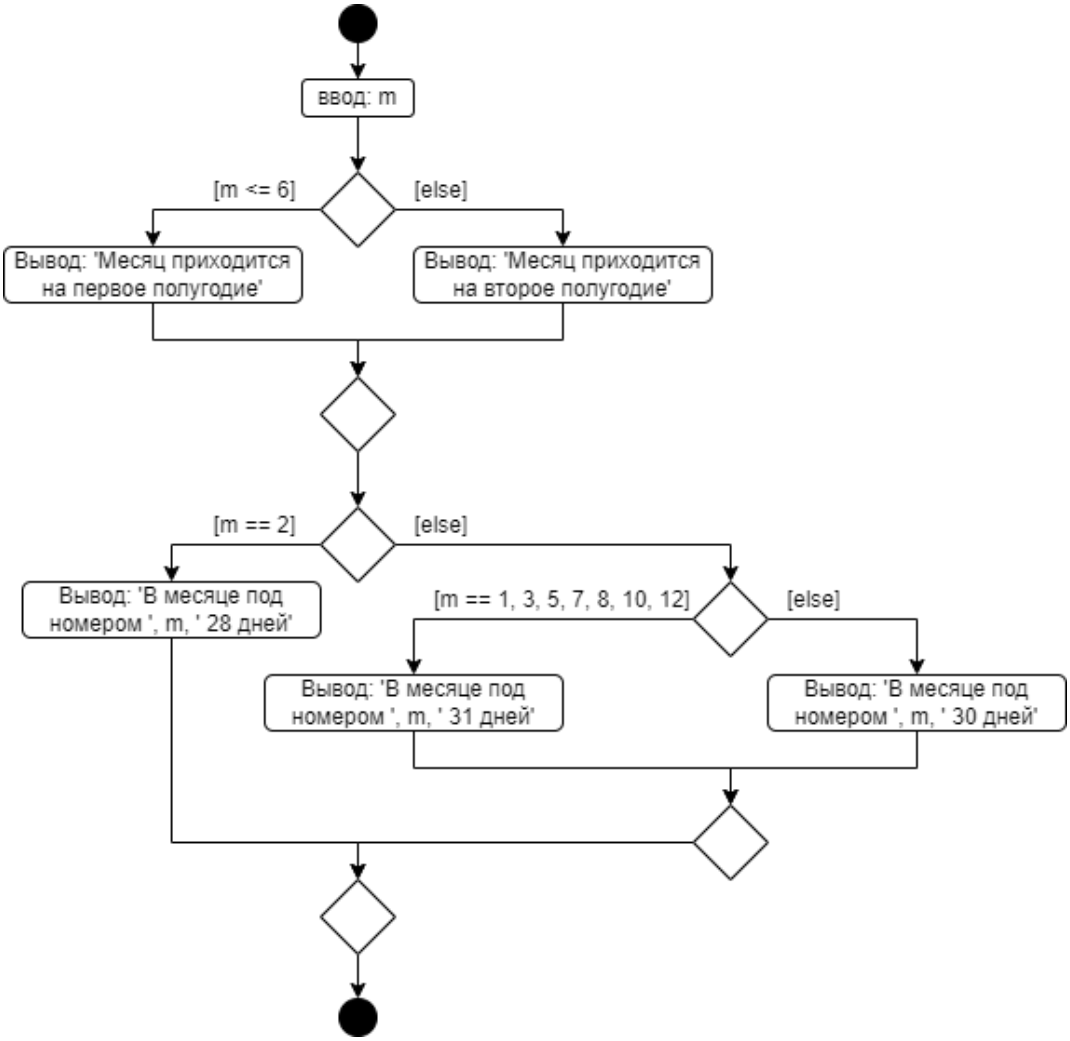


Рисунок 3.2 UML – диаграмма к программе инд. задания 1

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

if __name__ == '__main__':
    print('Введите координаты точки M1')
    x1 = int(input('x2 = '))
    y1 = int(input('y2 = '))

    print('Введите координаты точки M2')
    x2 = int(input('x2 = '))
    y2 = int(input('y2 = '))

    if x1 == x2:
        if y1 + y2 == 0:
            print('Точки M1 и M2 симметричны относительно начала координат')
    elif y1 == y2:
        if x1 + x2 == 0:
            print('Точки M1 и M2 симметричны относительно начала координат')
    else:
        print('Точки не симметричны относительно начала координат')
```

```
Введите координаты точки M1
x2 = 2
y2 = 3
Введите координаты точки M2
x2 = -2
y2 = 3
Точки M1 и M2 симметричны относительно начала координат
```

Рисунок 3.3 Программа к инд. заданию №2

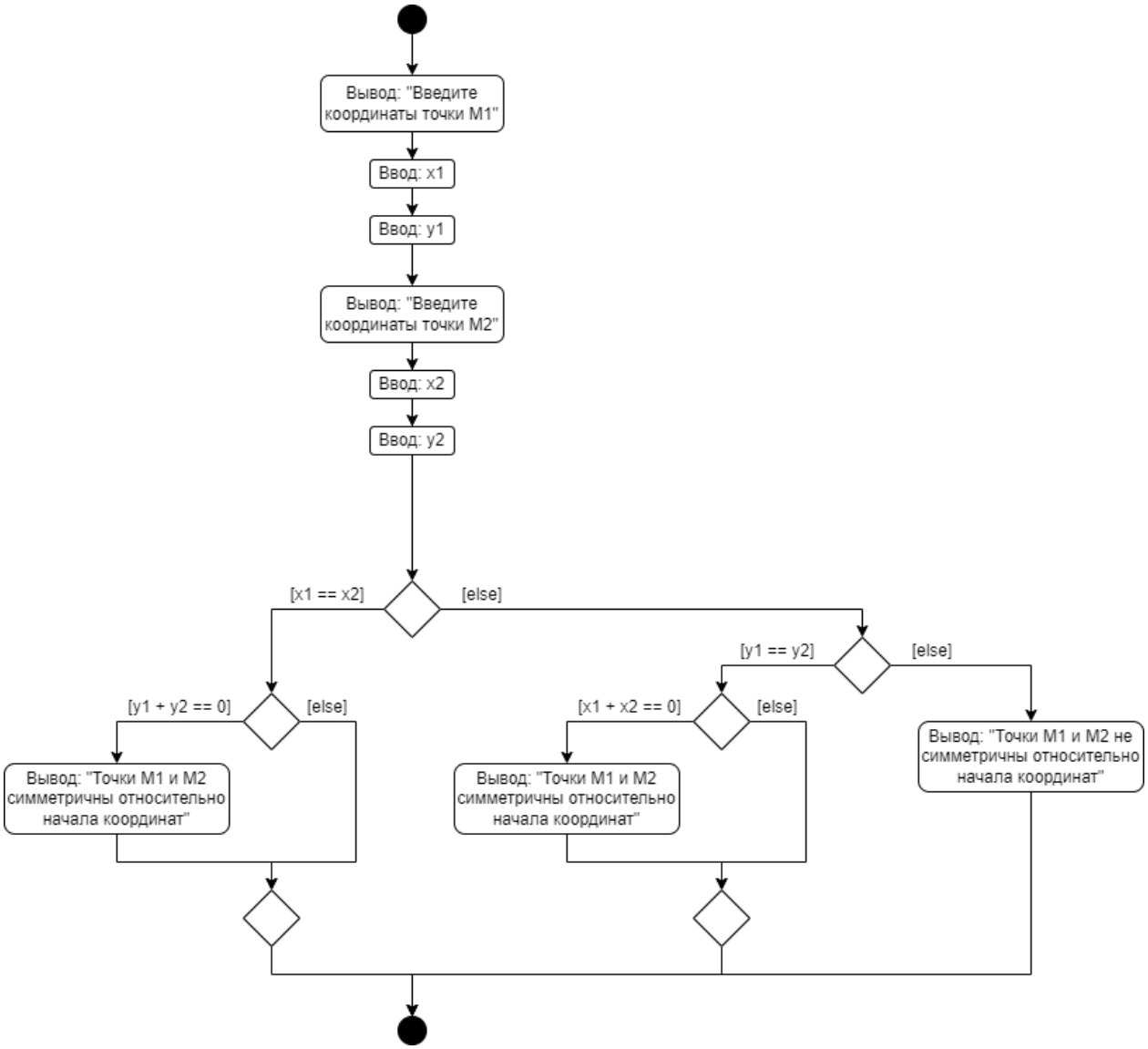


Рисунок 3.4 UML – диаграмма к программе инд. задания 2

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

if __name__ == '__main__':
    A = int(input('Введите стипендию: '))
    B = int(input('Введите расходы: '))
    s = B

    for i in range(10):
        B = (B*103) / 100
        s += B
    print(s)
    c = s - A*10
    print('Чтобы прожить учебный год, у родителей надо попросить ', c, ' р.')

```

```
Введите стипендию: 2000
Введите расходы: 10000
128077.95690814851
Чтобы прожить учебный год, у родителей надо попросить  108077.95690814851  р.
```

Рисунок 3.5 Программа к инд. заданию №3

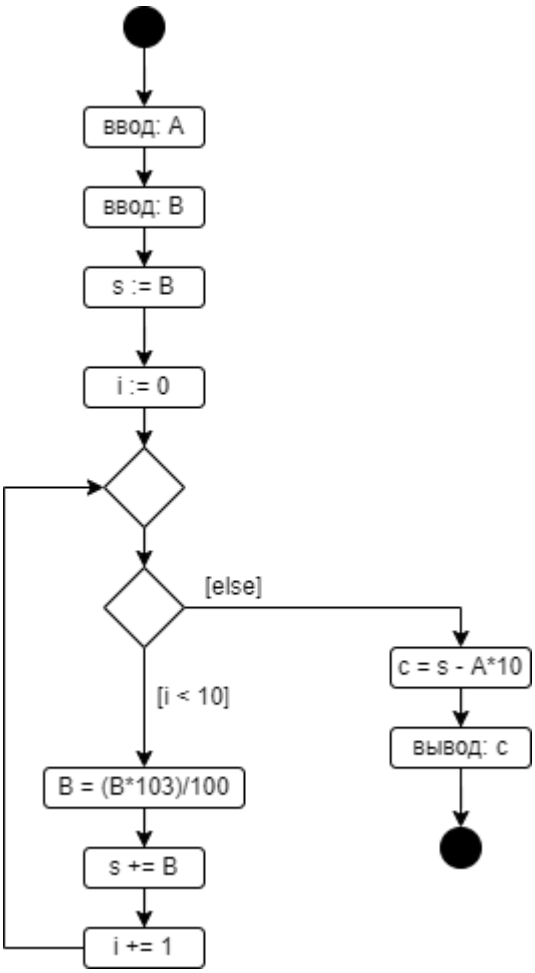

Рисунок 3.6 UML – диаграмма к программе инд. задания 3

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

import math
import sys


EULER = 0.5772156649015328606
EPS = 1e-10

if __name__ == '__main__':
    x = float(input("x =  "))
    if x == 0:
        print("Error", file=sys.stderr)
        exit(1)
    a = -x ** 2 / 4
    S, n = a, 1
    while math.fabs(a) > EPS:
        a *= (-1 * x ** 2 * 2 * n) / (2 * (n + 1)) ** 2
        S += a
        n += 1
    print(f"Ci({x}) = {EULER + math.log(math.fabs(x)) + S}")

```

Рисунок 3.7 Программа для задачи повышенной сложности.

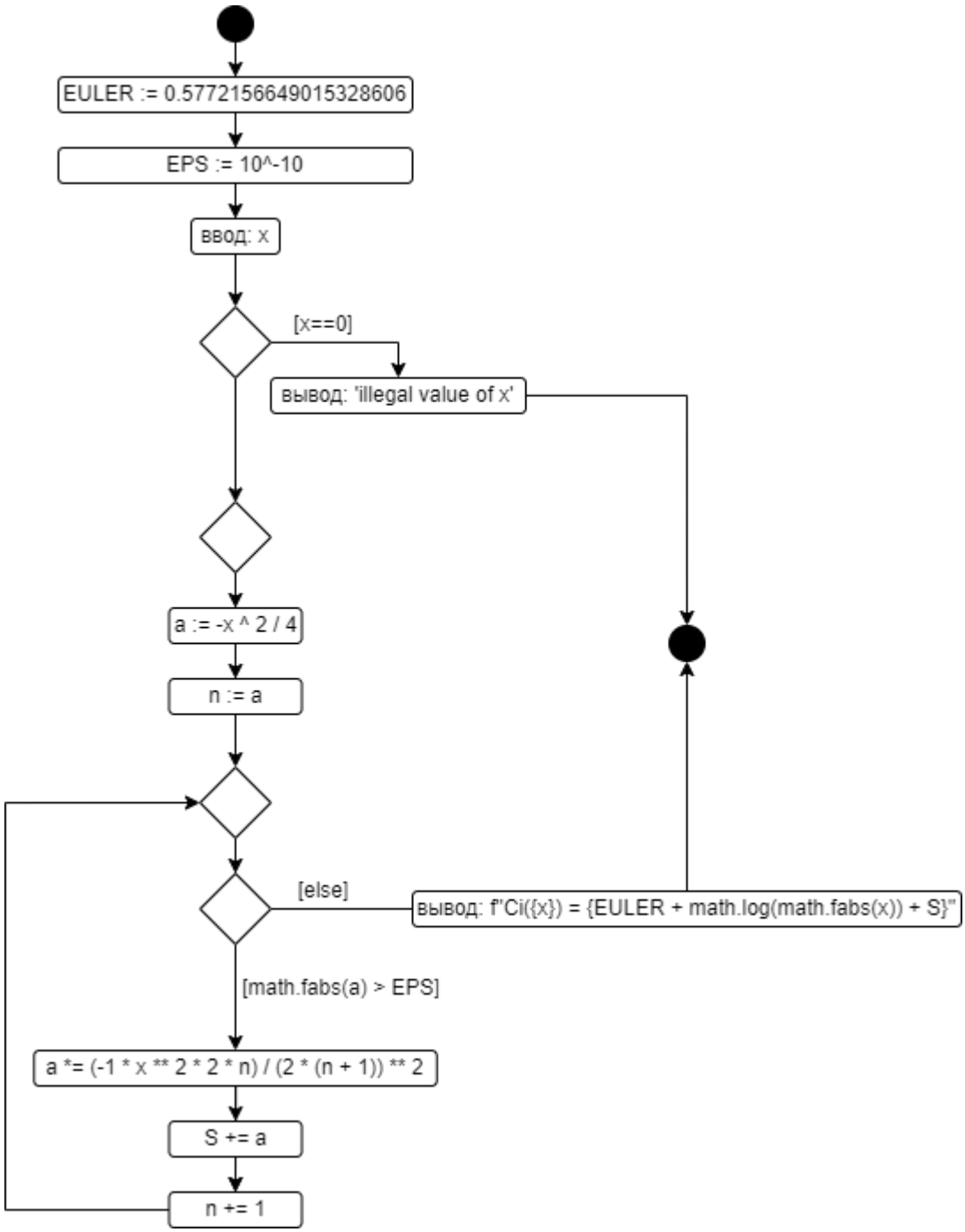


Рисунок 3.8 UML – диаграмма деятельности программы для усложненного задания

4. Сделал коммит, выполнил слияние с веткой main, и запушил изменения в уд. репозиторий.

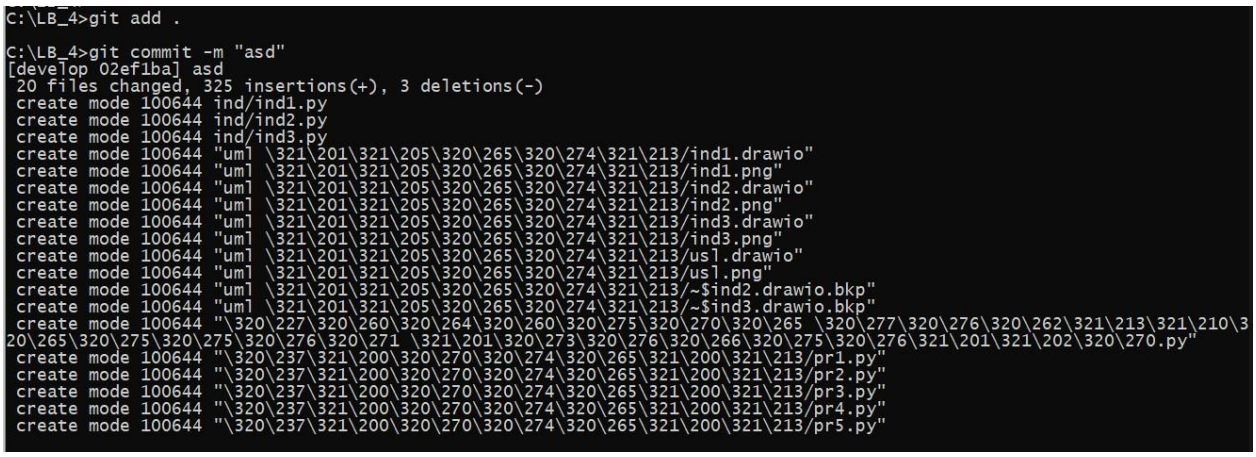


Рисунок 4.1 Фиксация и коммит файлов

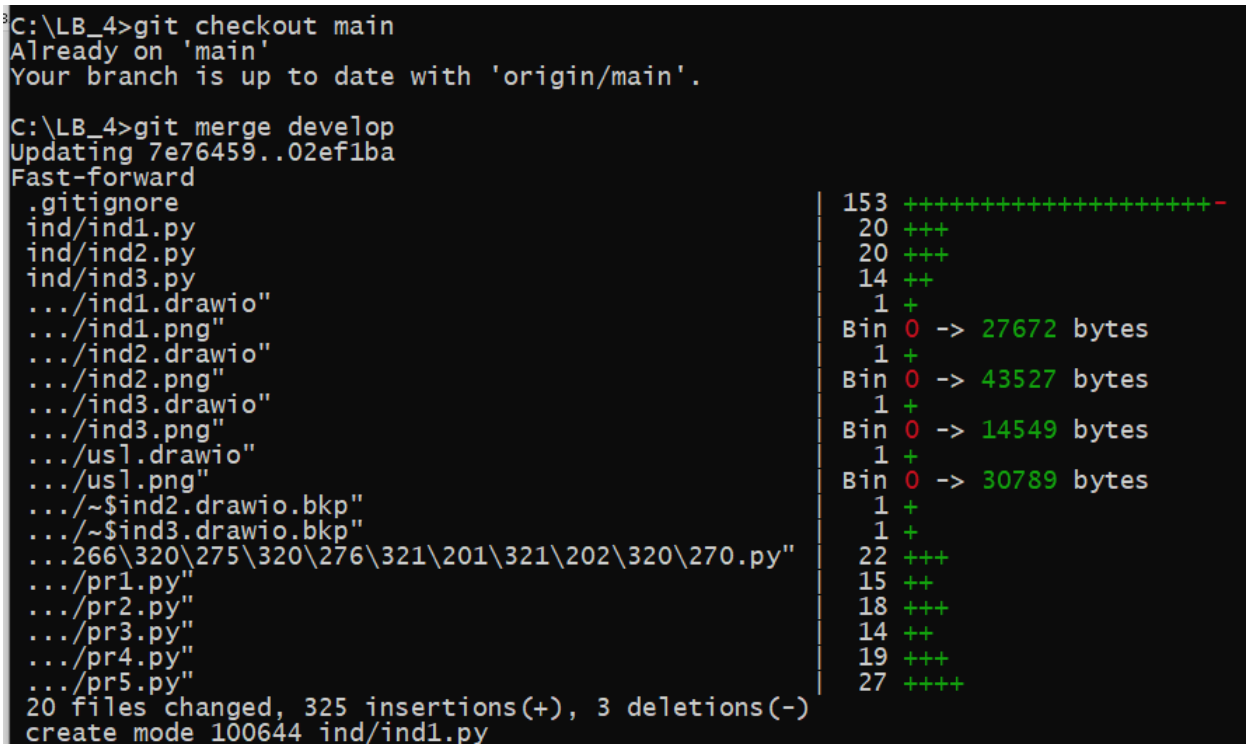


Рисунок 4.2 Слияние ветки develop с main

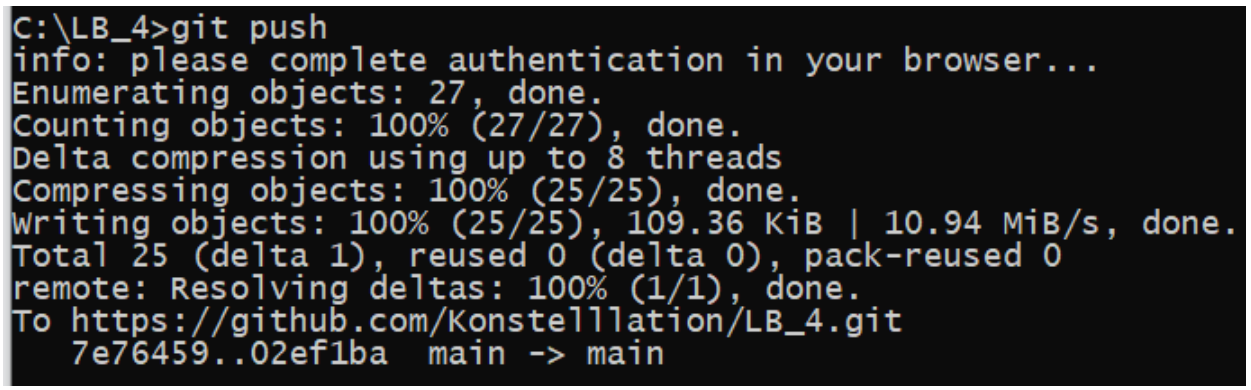


Рисунок 4.3 Пуш коммитов

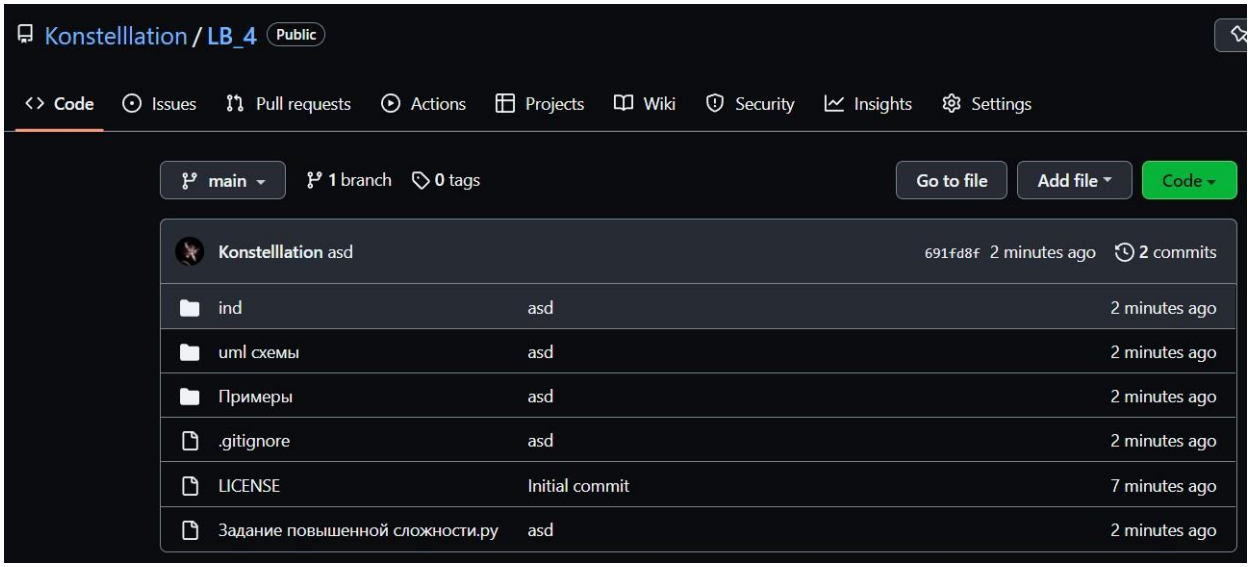


Рисунок 4.4 Изменения на уд. сервере

**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 объединенных 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. Могул 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.