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

Институт цифрового развития
Кафедра инфокоммуникаций

«Работа с функциями в языке Python»

ОТЧЕТ
по лабораторной работе №11
дисциплины
«Основы программной инженерии»

Выполнил:
Борсуков Владислав Олегович
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия», направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил:

_____
(подпись)

Отчет защищен с оценкой_____Дата защиты_____

Ставрополь, 2022 г.

Проработка примера из лабораторной работы:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import sys
from datetime import date


def get_worker():
    """
    Запросить данные о работнике.
    """
    name = input("Фамилия и инициалы? ")
    post = input("Должность? ")
    year = int(input("Год поступления? "))

    # Создать словарь.
    return {
        'name': name,
        'post': post,
        'year': year,
    }


def display_workers(staff):
    """
    Отобразить список работников.
    """
    # Проверить, что список работников не пуст.
    if staff:

        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "No",
                "Ф.И.О.",
```

Рисунок 1 – Код примера

```
                "Должность",
                "Год"
            )
        )
        print(line)

        # Вывести данные о всех сотрудниках.
        for idx, worker in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('name', ''),
                    worker.get('post', ''),
                    worker.get('year', 0)
                )
            )
        print(line)
    else:
        print("Список работников пуст.")


def select_workers(staff, period):
    """
    Выбрать работников с заданным стажем.
    """
    # Получить текущую дату.
    today = date.today()

    # Сформировать список работников.
    result = []
    for employee in staff:
        if today.year - employee.get('year', today.year) >= period:
            result.append(employee)

    # Возвратить список выбранных работников.
    return result


def main():
    """
    Главная функция программы.
```

Рисунок 2 – Код примера

```
    """
    # Список работников.

    workers = []

    # Организовать бесконечный цикл запроса команд.
    while True:

        # Запросить команду из терминала.
        command = input(">>> ").lower()

        # Выполнить действие в соответствие с командой.
        if command == 'exit':
            break
        elif command == 'add':

            # Запросить данные о работнике.
            worker = get_worker()

            # Добавить словарь в список.
            workers.append(worker)

            # Отсортировать список в случае необходимости.
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))

        elif command == 'list':
            # Отобразить всех работников.
            display_workers(workers)
        elif command.startswith('select '):

            # Разбить команду на части для выделения стажа.
            parts = command.split(' ', maxsplit=1)

            # Получить требуемый стаж.
            period = int(parts[1])

            # Выбрать работников с заданным стажем.
            selected = select_workers(workers, period)

```

Рисунок 3 – Код примера

```
123             # Отобразить выбранных работников.
124             display_workers(selected)
125
126         elif command == 'help':
127
128             # Вывести справку о работе с программой.
129             print("Список команд:\n")
130             print("add - добавить работника;")
131             print("list - вывести список работников;")
132             print("select <стаж> - запросить работников со стажем;")
133             print("help - отобразить справку;")
134             print("exit - завершить работу с программой.")
135         else:
136             print(f"Неизвестная команда {command}", file=sys.stderr)
137
138
139 if __name__ == '__main__':
140     main()
141
```

Рисунок 4 – Код примера

```
C:\Users\Borsukov\Desktop\LR11\PyCharm\venv\Scripts\python.exe C:\Users\Borsu
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Фамилия и инициалы? Иванов И.И.
Должность? Директор
Год поступления? 1980
>>> add
Фамилия и инициалы? Петров П.П.
Должность? Зам. директора
Год поступления? 1999
>>> add
Фамилия и инициалы? Мизин Г.Е.
Должность? Кринжик
Год поступления? 2022
>>> list
+------+----------------------------------+----------------------+----------+
|  No  |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Иванов И.И.                      | Директор             |     1980 |
|    2 | Мизин Г.Е.                       | Кринжик              |     2022 |
|    3 | Петров П.П.                      | Зам. директора       |     1999 |
+------+----------------------------------+----------------------+----------+
>>> select 42
+------+----------------------------------+----------------------+----------+
|  No  |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Иванов И.И.                      | Директор             |     1980 |
+------+----------------------------------+----------------------+----------+
>>> exit

Process finished with exit code 0
```

Рисунок 5 – Результат работы примера

Задание №1: решить следующую задачу: основная ветка программы, не считая заголовков функций, состоит из двух строки кода. Это вызов функции test() и инструкции if__name__== '__main__' . В ней запрашивается на ввод целое число. Если оно положительное, то вызывается функция positive(), тело которой содержит команду вывода на экран слова "Положительное". Если число отрицательное, то вызывается функция negative(), ее тело содержит выражение вывода на экран слова "Отрицательное". Понятно, что вызов test() должен следовать после определения функций. Однако имеет ли значение порядок определения самих функций? То есть должны ли определения positive() и negative() предшествовать test() или могут следовать после него? Проверьте вашу гипотезу, поменяв объявления функций местами. Попробуйте объяснить результат.

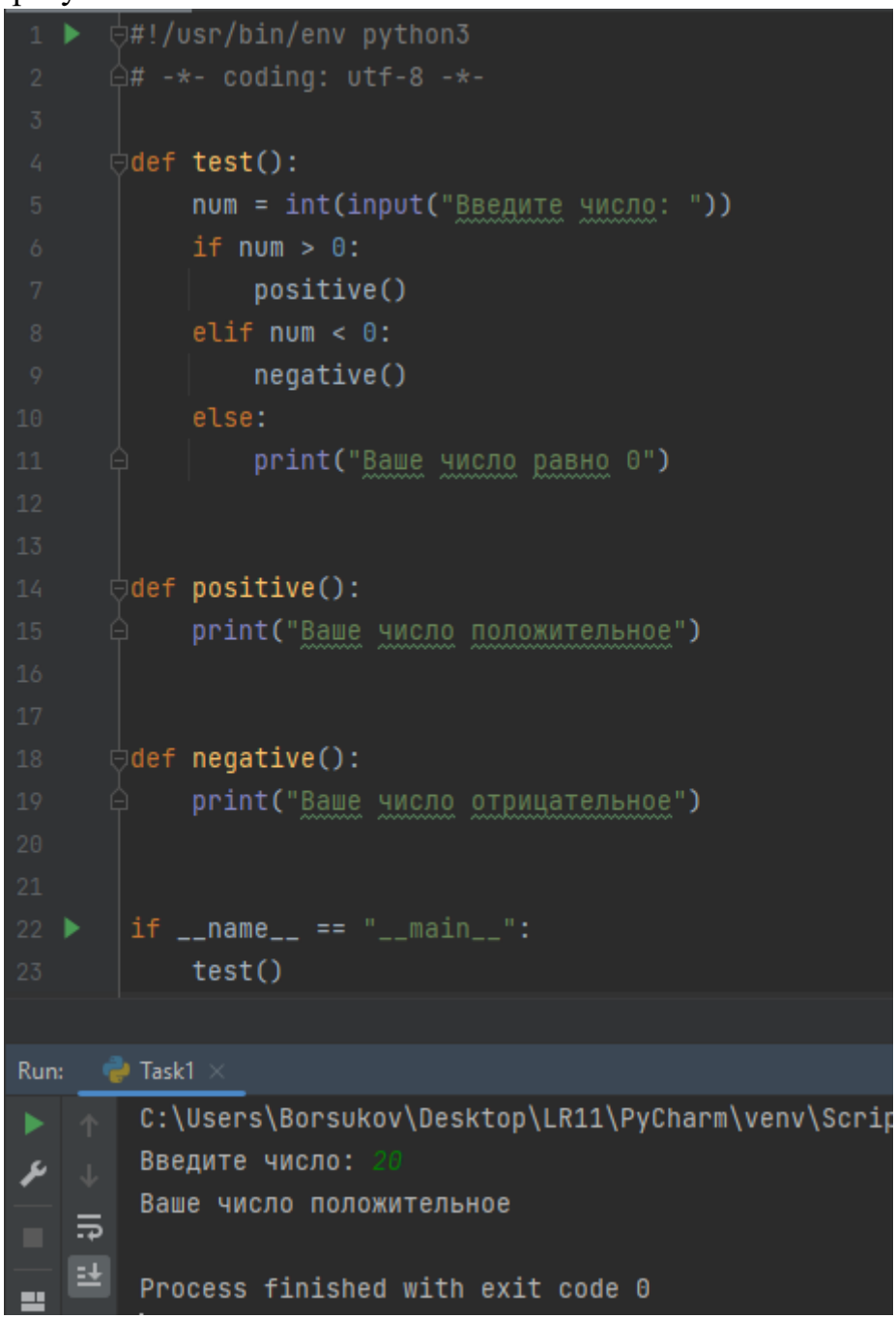

Рисунок 6 – Код и результат работы программы задания №1 с вызовом функций positive() и negative() после test()

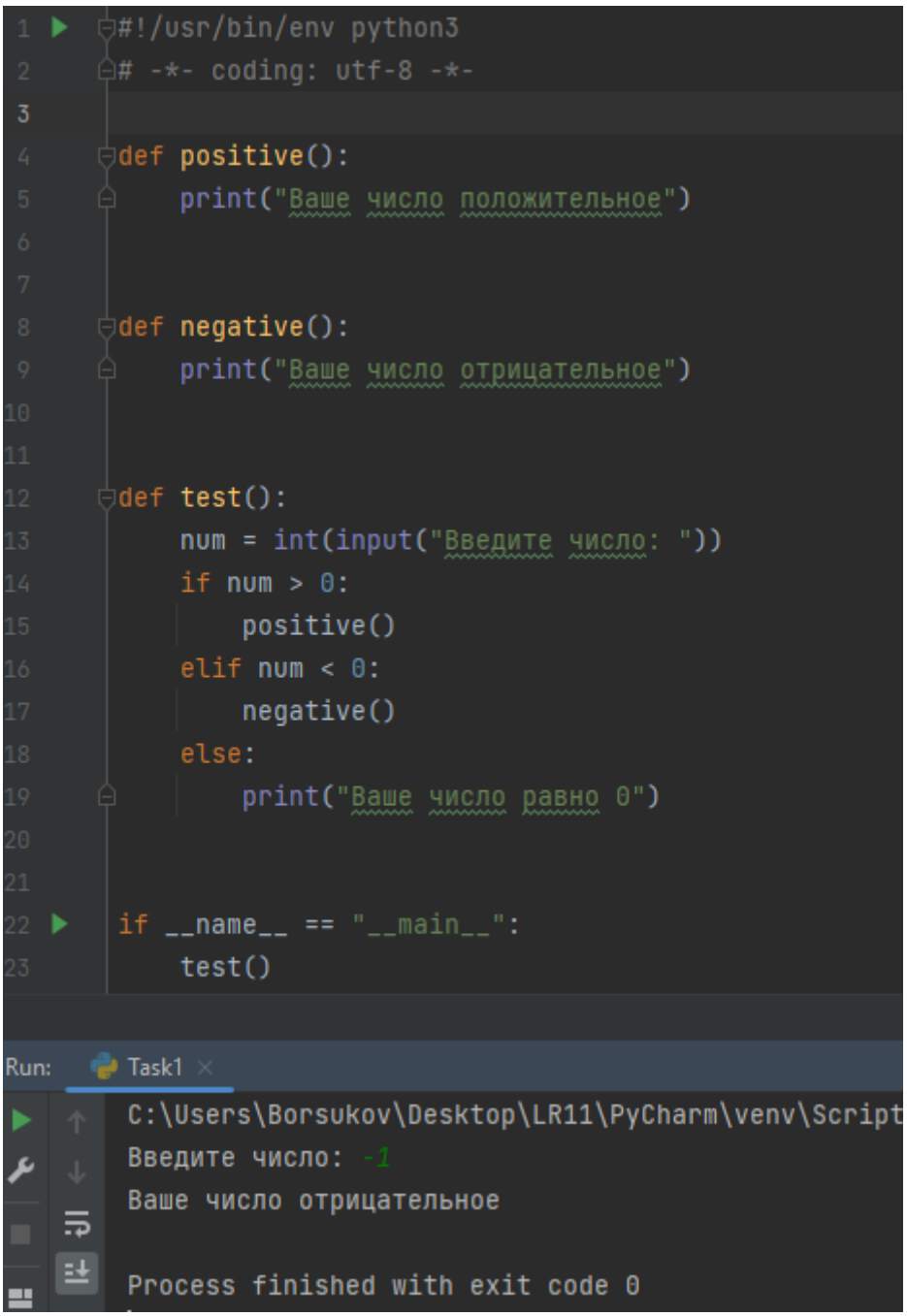

Рисунок 7 – Код и результат работы программы задания №1 с вызовом функций positive() и negative() перед test()

Задание №2: Решите следующую задачу: в основной ветке программы вызывается функция cylinder(), которая вычисляет площадь цилиндра. В теле cylinder() определена функция circle(), вычисляющая площадь круга по формуле . В теле cylinder() у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле , или полную площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции circle().

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

import math


def cylinder():
    r = float(input("Введите радиус цилиндра: "))
    h = float(input("Введите высоту цилиндра: "))
    s = 2 * math.pi * r * h

    def circle():
        s_circle = math.pi * r ** 2
        return s_circle

    check = int(input("1 - боковая площадь; 2 - полная площадь: "))
    if check == 1:
        print(f"Боковая площадь цилиндра: {s}")
    else:
        full_s = s + circle() * 2
        print(f"Полная площадь цилиндра: {full_s}")


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

```
Run:  Task2
C:\Users\Borsukov\Desktop\LR11\PyCharm\venv\Scripts\python.exe C:\Use
Введите радиус цилиндра: 10
Введите высоту цилиндра: 5
1 - боковая площадь; 2 - полная площадь: 2
Полная площадь цилиндра: 942.4777960769379

Process finished with exit code 0
```

Рисунок 8 – Код и результат работы программы задания №2

Задание №3: решите следующую задачу: напишите функцию, которая считывает с клавиатуры числа и перемножает их до тех пор, пока не будет введен 0. Функция должна возвращать полученное произведение. Вызовите функцию и выведите на экран результат ее работы.

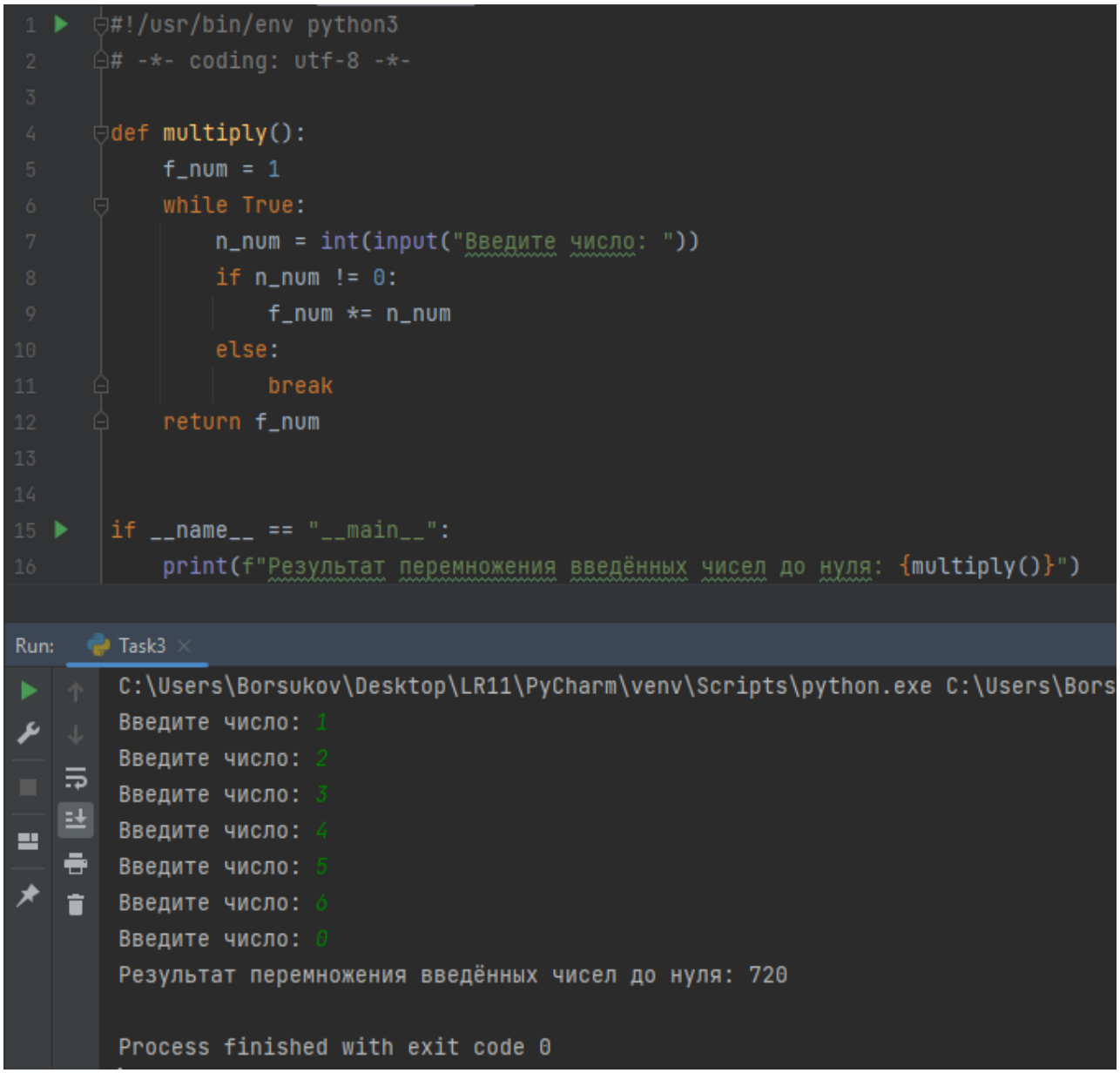

Рисунок 9 – Код и результат работы программы задания №3

Задание №4: решите следующую задачу: напишите программу, в которой определены следующие четыре функции:

1. Функция get_input() не имеет параметров, запрашивает ввод с клавиатуры и возвращает в основную программу полученную строку.

2. Функция test_input() имеет один параметр. В теле она проверяет, можно ли переданное ей значение преобразовать к целому числу. Если можно, возвращает логическое True. Если нельзя – False.

3. Функция str_to_int() имеет один параметр. В теле преобразовывает переданное значение к целочисленному типу. Возвращает полученное число.

4. Функция print_int() имеет один параметр. Она выводит переданное значение на экран и ничего не возвращает.

В основной ветке программы вызовите первую функцию. То, что она вернула, передайте во вторую функцию. Если вторая функция вернула True, то те же данные (из первой функции) передайте в третью функцию, а возвращенное третьей функцией значение – в четвертую.

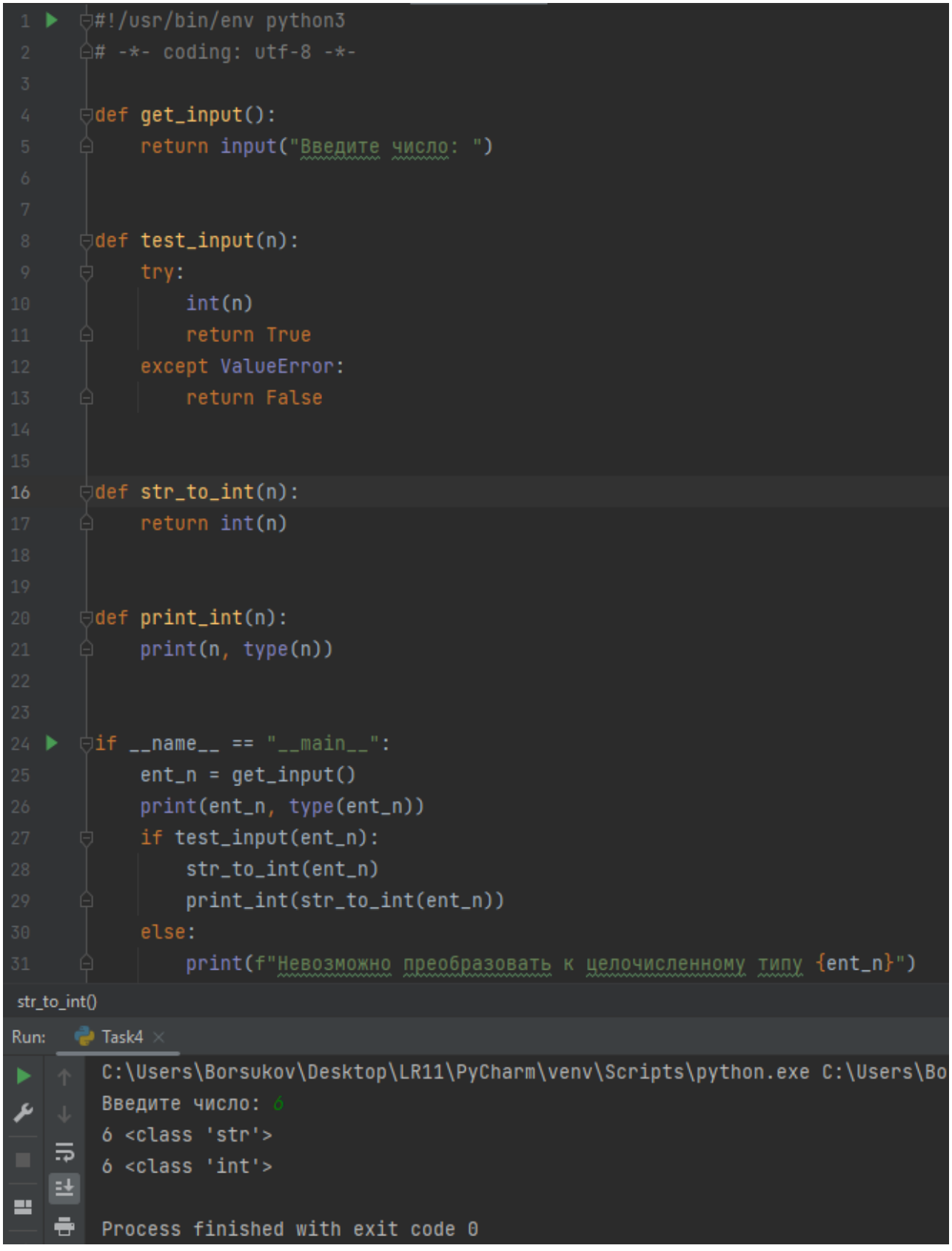

Рисунок 10 – Код и результат работы программы задания №

Индивидуальное задание: решить индивидуальное задание лабораторной работы 2.6, оформив каждую команду в виде отдельной функции.

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

import sys


def help_info():
    print("Список команд:")
    print("add - добавить студента")
    print("list - вывести список студентов")
    print("filter list - список студентов со средним баллом больше 4")
    print("exit - завершить работу с программой")


def add_student():
    name = input("Фамилия и инициалы студента: ")
    group = int(input("Номер группы: "))
    marks = list(map(int, input("Пять оценок студента: ").split()))

    if len(marks) != 5:
        print("Неверное количество оценок", file=sys.stderr)
        return

    student = {
        'name': name,
        'group': group,
        'marks': marks,
    }

    if sum(marks) / 5 > 4:
        filter_students.append(student)

    students.append(student)


def out_students():
    line = '+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 30,
```

Рисунок 11 – Код программы для индивидуального задания

```
            '-' * 14,
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^14} |'.format(
                "№",
                "Ф.И.О.",
                "Номер группы",
            )
        )
        print(line)

        for idx, student in enumerate(students, 1):
            print(
                '| {:>4} | {:<30} | {:<14} |'.format(
                    idx,
                    student.get('name', ''),
                    student.get('group', ''),
                )
            )
        print(line)


def out_students_finter():
    line = '+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 30,
        '-' * 14,
    )
    print(line)
    print(
        '| {:^4} | {:^30} | {:^14} |'.format(
            "№",
            "Ф.И.О.",
            "Номер группы",
        )
    )
    print(line)

```

Рисунок 12 – Код программы для индивидуального задания

```
    for idx, student in enumerate(filter_students, 1):
        print(
            '| {:>4} | {:<30} | {:<14} |'.format(
                idx,
                student.get('name', ''),
                student.get('group', ''),
            )
        )
    print(line)


def main():
    while True:
        command = input(">>> ").lower()

        if command == 'exit':
            break

        elif command == 'help':
            help_info()

        elif command == 'add':
            add_student()

            if len(students) > 1:
                students.sort(key=lambda item: item.get('group', ''))

            if len(filter_students) > 1:
                filter_students.sort(key=lambda item: item.get('group', ''))

        elif command == 'list':
            if len(students) > 0:
                out_students()
            else:
                print("Список студентов пустой.")

        elif command == "filter list":
            if len(students) > 0:
                out_students_finter()
```

Рисунок 13 – Код программы для индивидуального задания

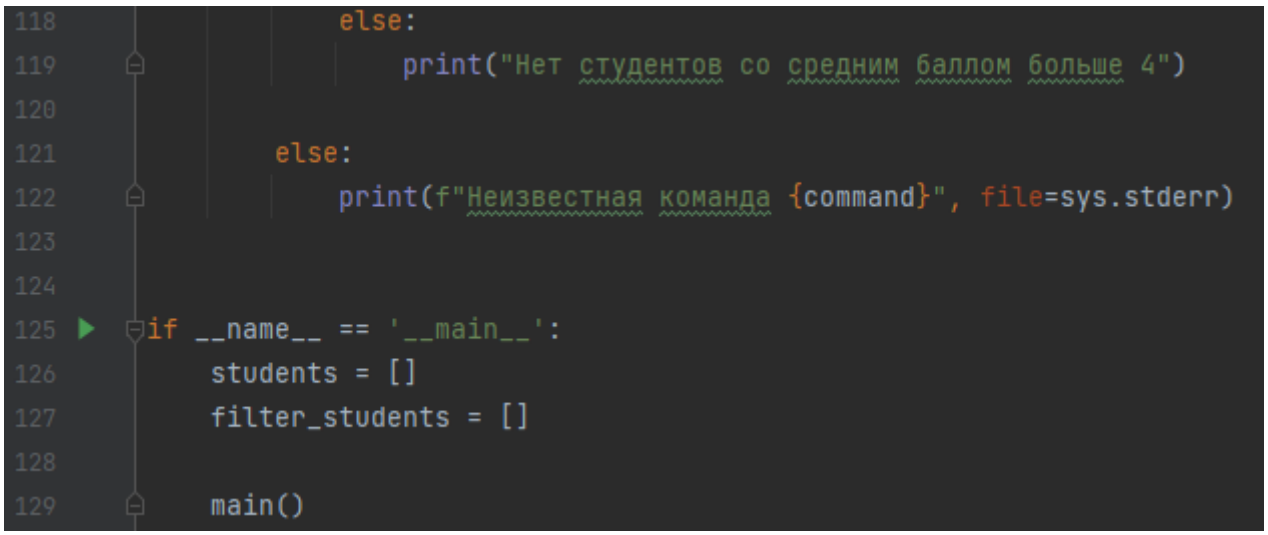

Рисунок 14 – Код программы для индивидуального задания

```
C:\Users\Borsukov\Desktop\LR11\PyCharm\venv\Scripts\python.exe C:\
>>> help
Список команд:
add - добавить студента
list - вывести список студентов
filter list - список студентов со средним баллом больше 4
exit - завершить работу с программой
>>> add
Фамилия и инициалы студента: Петросян А.Ю.
Номер группы: 25
Пять оценок студента: 4 5 4 5 4
>>> ADD
Фамилия и инициалы студента: Мясников С.И.
Номер группы: 10
Пять оценок студента: 4 4 4 4 5
>>> add
Фамилия и инициалы студента: Иванов И.И.
Номер группы: 3
Пять оценок студента: 3 3 3 3 3
>>> list
+------+----------------------------------+----------------+
|  №   |             Ф.И.О.               |  Номер группы  |
+------+----------------------------------+----------------+
|    1 | Иванов И.И.                      | 3              |
|    2 | Мясников С.И.                    | 10             |
|    3 | Петросян А.Ю.                    | 25             |
+------+----------------------------------+----------------+
>>> filter list
+------+----------------------------------+----------------+
|  №   |             Ф.И.О.               |  Номер группы  |
+------+----------------------------------+----------------+
|    1 | Мясников С.И.                    | 10             |
|    2 | Петросян А.Ю.                    | 25             |
+------+----------------------------------+----------------+
>>> git
>>> Неизвестная команда git
exit

Process finished with exit code 0
```

Рисунок 15 – Результат работы программы для индивидуального задания

Контрольные вопросы

1. Каково назначение функций в языке программирования Python?

Функция в программировании представляет собой обособленный участок кода, который можно вызывать, обратившись к нему по имени, которым он был назван. При вызове происходит выполнение команд тела функции.

2. Каково назначение операторов def и return ?

В языке программирования Python функции определяются с помощью оператора def

Функции могут передавать какие-либо данные из своих тел в основную ветку программы. Говорят, что функция возвращает значение. В большинстве языков программирования, в том числе Python, выход из функции и передача данных в то место, откуда она была вызвана, выполняется оператором return.

3. Каково назначение локальных и глобальных переменных при написании функций в Python?

В программировании особое внимание уделяется концепции о локальных и глобальных переменных, а также связанное с ними представление об областях видимости. Соответственно, локальные переменные видны только в локальной области видимости, которой может выступать отдельно взятая функция. Глобальные переменные видны во всей программе. "Видны" – значит, известны, доступны. К ним можно обратиться по имени и получить связанное с ними значение.

4. Как вернуть несколько значений из функции Python?

```
    return side, full


scyl, fcyl = cylinder()
```

5. Какие существуют способы передачи значений в функцию?

Через параметры, и через ввод, запрашиваемый самой функцией

6. Как задать значение аргументов функции по умолчанию?

```
def cylinder(h, r=1):
```

7. Каково назначение lambda-выражений в языке Python?

Python поддерживает интересный синтаксис, позволяющий определять небольшие однострочные функции на лету. Позаимствованные из Lisp, так называемые lambda-функции могут быть использованы везде, где требуется функция

8. Как осуществляется документирование кода согласно PEP257?

Документирование кода в python - достаточно важный аспект, ведь от нее порой зависит читаемость и быстрота понимания вашего кода, как другими людьми, так и вами через полгода. PEP 257 описывает соглашения, связанные со строками документации python, рассказывает о том, как нужно документировать python код. Цель этого PEP - стандартизировать структуру строк документации: что они должны в себя включать, и как это написать (не касаясь вопроса синтаксиса строк документации). Этот PEP описывает соглашения, а не правила или синтаксис.

9. В чем особенность однострочных и многострочных форм строк документации?

Для согласованности, всегда используйте """triple double quotes""" для строк документации. Используйте r"""raw triple double quotes""" , если вы будете использовать обратную косую черту в строке документации. Существует две формы строк документации: однострочная и многострочная.