

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 множествами в языке Python»

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

Выполнила:

Кувшин Ирина Анатольевна
2 курс, группа ПИЖ-б-о-21-1,
011.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

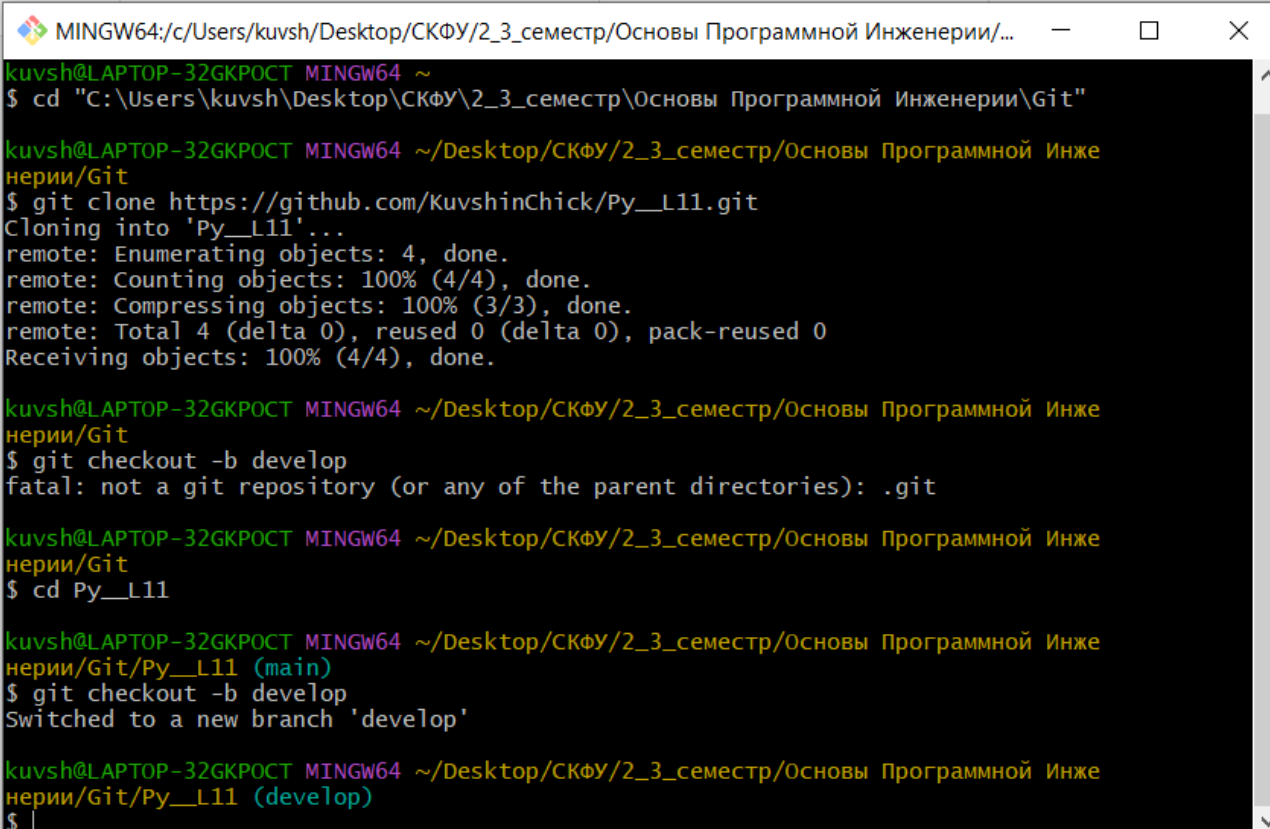
Ставрополь, 2022 г.

Цель работы: приобретение навыков по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.

Ссылка на репозиторий: https://github.com/KuvshinChick/Py__L11.git

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.



```
MINGW64:/c:/Users/kuvsh/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/...
kuvsh@LAPTOP-32GKPOCT MINGW64 ~
$ cd "C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git"
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ git clone https://github.com/KuvshinChick/Py__L11.git
Cloning into 'Py__L11'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ git checkout -b develop
fatal: not a git repository (or any of the parent directories): .git
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ cd Py__L11
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L11 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L11 (develop)
$
```

Рисунок 11.1 – Клонирование репозитория и создание ветки develop

```
MINGW64/c/Users/kuvsh/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py_L11
Switched to a new branch 'develop'

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py_L11 (develop)
$ git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

no changes added to commit (use "git add" and/or "git commit -a")

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py_L11 (develop)
$ git add .

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py_L11 (develop)
$ git commit -m "modified .gitignore & readme"
[develop f62ae3d] modified .gitignore & readme
 2 files changed, 133 insertions(+), 1 deletion(-)
 create mode 100644 .gitignore

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py_L11 (develop)
$ git push origin develop
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 1.52 KiB | 1.52 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/KuvshinChick/Py_L11/pull/new/develop
remote:
To https://github.com/KuvshinChick/Py_L11.git
 * [new branch]      develop -> develop

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py_L11 (develop)
$ |
```

Рисунок 11.2 – Обновление .gitignore и readme

6. Создайте проект PyCharm в папке репозитория.

7. Проработайте примеры лабораторной работы. Зафиксируйте изменения.

Пример 1. Для примера 1 лабораторной работы 2.6, оформить каждую команду в виде вызова отдельной функции.

```
#!/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.",
                "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.
    """
    # 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)

        # Отобразить выбранных работников.
        display_workers(selected)

    elif command == 'help':

        # Вывести справку о работе с программой.
        print("Список команд:\n")
        print("add - добавить работника;")
        print("list - вывести список работников;")
        print("select <стаж> - запросить работников со стажем;")
        print("help - отобразить справку;")
        print("exit - завершить работу с программой.")
    else:
        print(f"Неизвестная команда {command}", file=sys.stderr)

if __name__ == '__main__':
    main()

```

Рисунок 11.3 – Код программы-примера

```
ex_1
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git\Py_L11\PyCharm\venv\Scripts\python.exe" "C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git\Py_L11\PyCharm\venv\Scripts\python.exe"
>>> 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.А.
Должность? Программист
Год поступления? 2004
>>> add
Фамилия и инициалы? Петров В.Д.
Должность? Администратор
Год поступления? 1997
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Иванов И.А.              |      Программист     |      2004     |
|  2 | Петров В.Д.              |      Администратор   |      1997     |
+-----+-----+-----+-----+
>>> select 1
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Иванов И.А.              |      Программист     |      2004     |
|  2 | Петров В.Д.              |      Администратор   |      1997     |
+-----+-----+-----+-----+
>>> select 2
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Иванов И.А.              |      Программист     |      2004     |
|  2 | Петров В.Д.              |      Администратор   |      1997     |
+-----+-----+-----+-----+
>>> select 10
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Иванов И.А.              |      Программист     |      2004     |
|  2 | Петров В.Д.              |      Администратор   |      1997     |
+-----+-----+-----+-----+
>>> select 20
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Петров В.Д.              |      Администратор   |      1997     |
+-----+-----+-----+-----+
>>>
```

Рисунок 11.4 – Результат работы программы – примера

8. 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.

9. Зафиксируйте изменения в репозитории.

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

def test():
    number = int(input("Введите число: "))
    if number > 0:
        positive()
    elif number < 0:
        negative()
    else:
        print("Ноль")

def positive():
    print("Положительное")

def negative():
    print("Отрицательное")

if __name__ == '__main__':
    test()
```

Рисунок 11.5 – Код программы

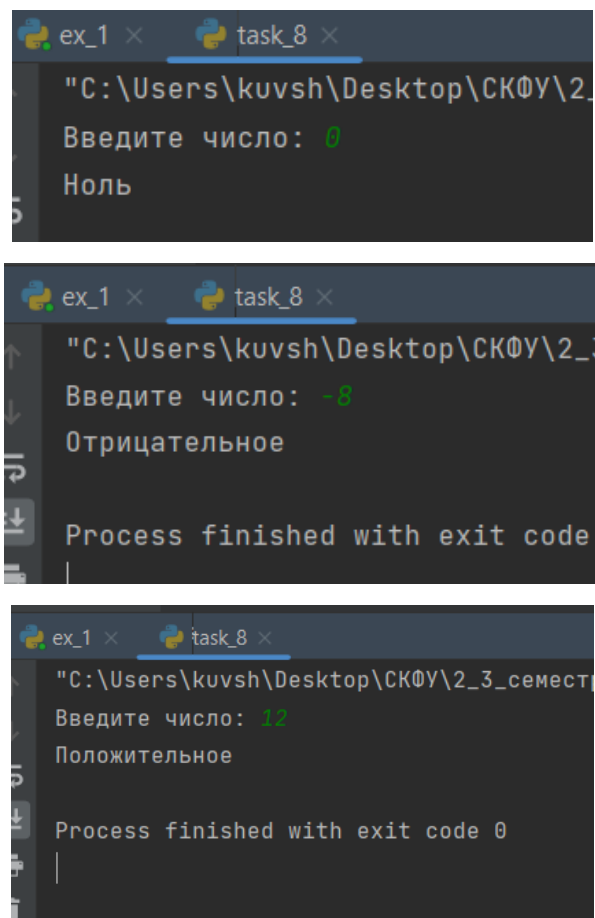


Рисунок 11.6 – Результат работы программы

10. 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 $S = \pi r^2$. В теле `cylinder()` у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле $S_{бок} = 2\pi rh$, или полную площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции `circle()`.

11. Зафиксируйте изменения в репозитории.

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

import math

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

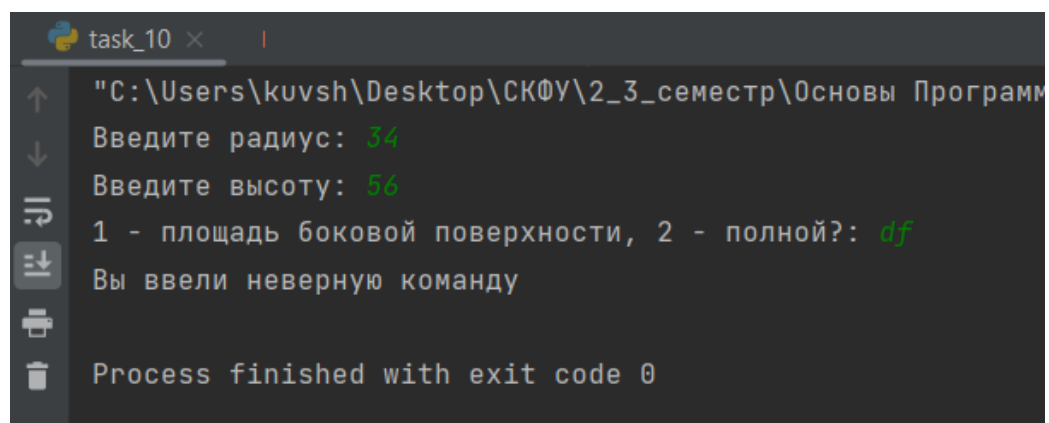
    try:
        r = float(input("Введите радиус: "))
        h = float(input("Введите высоту: "))
    except ValueError:
        return

    side = 2 * math.pi * r * h

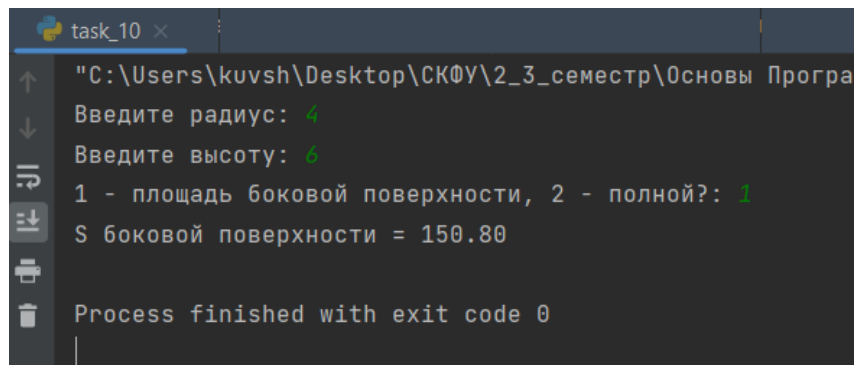
    answer = input("1 - площадь боковой поверхности, 2 - полной?: ")
    if answer == '1':
        print(f"S боковой поверхности = {side:.2f}")
    elif answer == '2':
        print(f"S полной поверхности = {side + circle() * 2:.2f}")
    else:
        print("Вы ввели неверную команду")

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

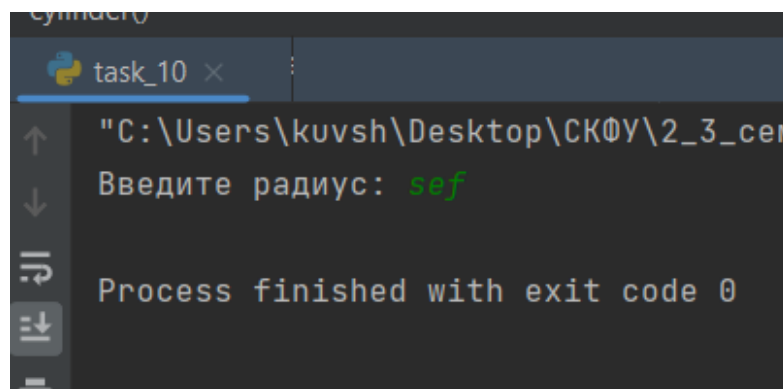
Рисунок 11.7 – Код программы



```
task_10 x |
C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программ
Введите радиус: 34
Введите высоту: 56
1 - площадь боковой поверхности, 2 - полной?: df
Вы ввели неверную команду
Process finished with exit code 0
```

```
task_10 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Програ
Введите радиус: 4
Введите высоту: 6
1 - площадь боковой поверхности, 2 - полной?: 1
S боковой поверхности = 150.80
Process finished with exit code 0
```

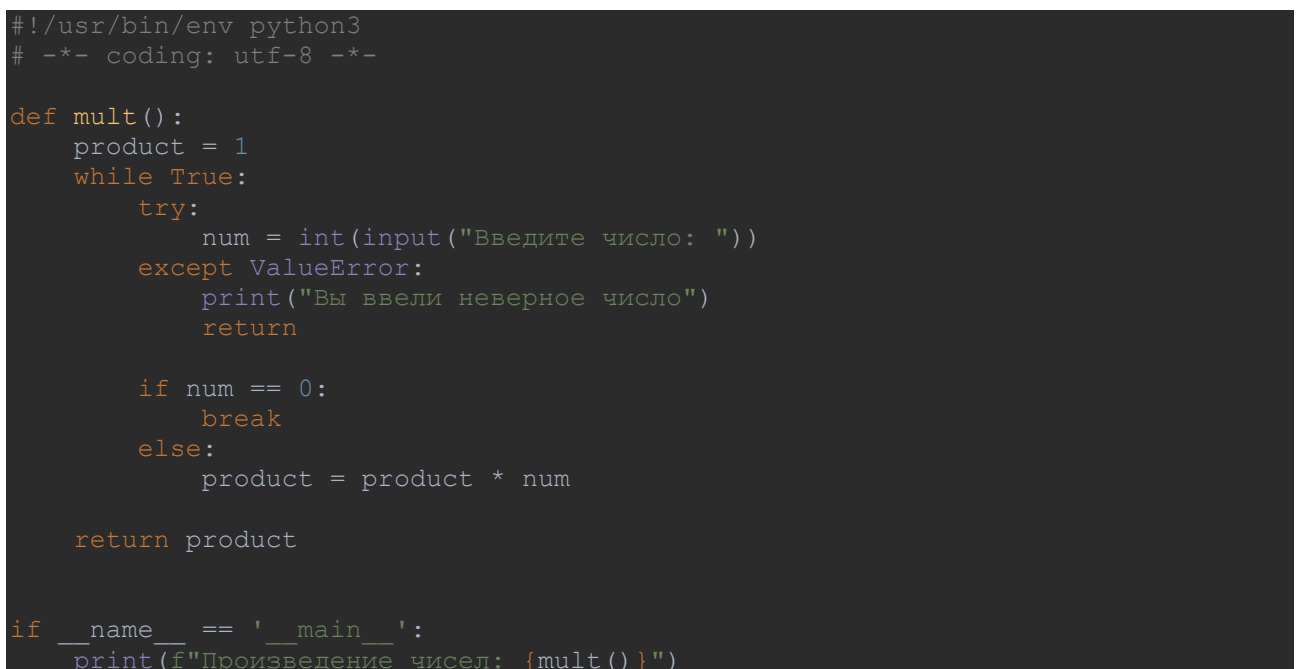


```
task_10 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_сем
Введите радиус: sef
Process finished with exit code 0
```

Рисунок 11.8 – Результат работы программы

12. 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.

13. Зафиксируйте изменения в репозитории.



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

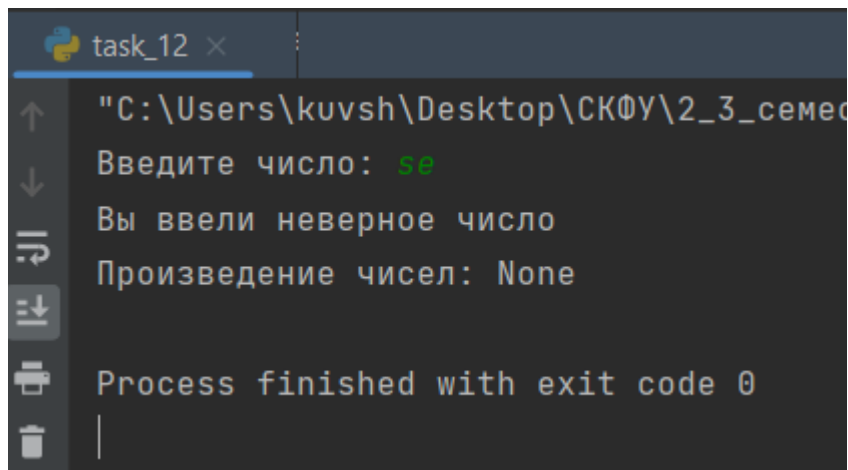
def mult():
    product = 1
    while True:
        try:
            num = int(input("Введите число: "))
        except ValueError:
            print("Вы ввели неверное число")
            return

        if num == 0:
            break
        else:
            product = product * num

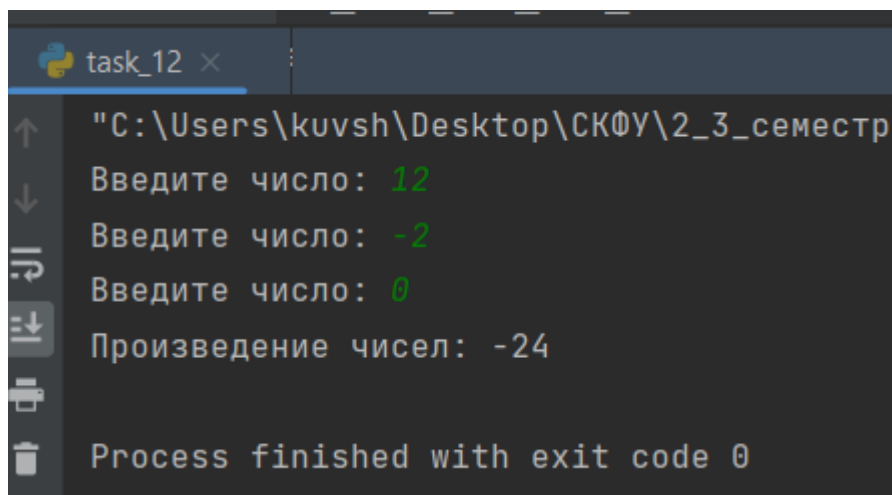
    return product

if __name__ == '__main__':
    print(f"Произведение чисел: {mult()}")
```

Рисунок 11.9 – Код программы



```
task_12 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр
Введите число: se
Вы ввели неверное число
Произведение чисел: None
Process finished with exit code 0
```



```
task_12 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр
Введите число: 12
Введите число: -2
Введите число: 0
Произведение чисел: -24
Process finished with exit code 0
```

Рисунок 11.10 – Результат работы программы

14. 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.

15. Зафиксируйте изменения в репозитории.

16. Приведите в отчете скриншоты результатов выполнения примера при различных исходных данных вводимых с клавиатуры.

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

def get_input():
    return input("Введите строку: ")

def test_input(string):
    try:
        int(string)
        return True
    except ValueError:
        return False

def str_to_int(string):
    return int(string)

def print_int(string):
    print(string)

if __name__ == '__main__':
    s = get_input()
    test_input(s)
    if test_input(s):
        str_to_int(s)
        print_int(str_to_int(s))
    else:
        print("Не число")
```

Рисунок 11.11 – Код программы

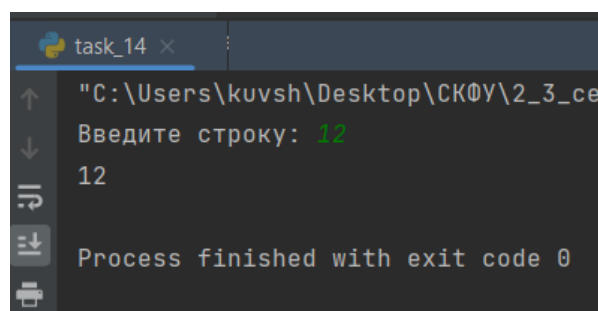
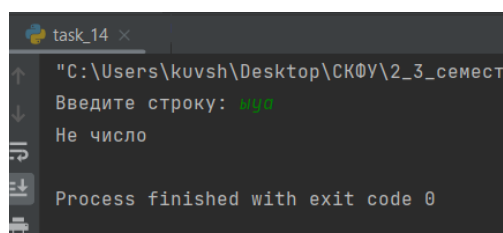


Рисунок 11.12 – Результат работы программы

17. Приведите в отчете скриншоты работы программ решения индивидуального задания.

18. Зафиксируйте сделанные изменения в репозитории.

Решить индивидуальное задание лабораторной работы 2.6, оформив каждую команду в виде отдельной функции.

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

import datetime

def help_inf():
    # Вывести справку о работе с программой.
    print("\033[32mСписок команд:\n")
    print("add - добавить человека;")
    print("select <ЗЗ> - запросить людей с определенным ЗЗ;")
    print("show - показать список людей;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")

def add_human():
    # Кортеж ЗЗ
    zodiac_signs = (
        03.21,
        "овен",
        04.21,
        "телец",
        05.22,
        "близнецы",
        06.22,
        "рак",
        07.23,
        "лев",
        08.23,
        "дева",
        09.24,
        "весы",
        10.24,
        "скорпион",
        11.23,
        "стрелец",
        12.22,
        "козерог",
        01.21,
        "водолей",
        02.19,
        "рыбы",
        03.21
    )

    zs_ny = (
        12.22,
        "козерог",
        01.21,
        "водолей",
        02.19,
        "рыбы",
        03.21
    )

    # Запросить данные человека
```

```

name = input("Enter name: ")
# Проверка ЗЗ
while True:
    zodiac_sign = input("\033[0mEnter zodiac sign: ").lower()
    if zodiac_sign not in zodiac_signs \
        and zodiac_sign not in zs_ny:
        print("\033[31mIncorrect zodiac sign")
    else:
        break

# Проверка Даты Рождения
while True:
    birth = input("\033[0mEnter date of birth (yyyy.mm.dd): ")
    # Обработка исключения (проверка правильности даты)
    try:
        birth_data = datetime.datetime.strptime(birth, '%Y.%m.%d')
    except Exception:
        print("\033[31mIncorrect data format")
    # Если ДР введен корректно относительно формата,
    # проверяем дату согласно ЗЗ
    else:
        # Получаем из кортежа начальную дату введенного ЗЗ
        # и преобразуем к нужному формату даты
        if zodiac_sign in zs_ny:
            beg_zs_data = datetime.datetime.strptime(
                str(birth_data.year-1) + '.' +
                str(zs_ny[zs_ny.index(zodiac_sign) - 1]),
                '%Y.%m.%d'
            )
            # Получаем из кортежа конечную дату введенного ЗЗ
            # и преобразуем к нужному формату даты
            end_zs_data = datetime.datetime.strptime(
                str(birth_data.year) + '.' +
                str(zs_ny[zs_ny.index(zodiac_sign) + 1]),
                '%Y.%m.%d'
            )
            # Сравнение введенной даты с промежутком дат ЗЗ
            if beg_zs_data <= birth_data < end_zs_data:
                break
            else:
                print("\033[31mEnter CORRECR date of birth "
                    "(yyyy.mm.dd): ")
        else:
            beg_zs_data = datetime.datetime.strptime(
                str(birth_data.year) + '.' +
                str(zodiac_signs[zodiac_signs.index(zodiac_sign) - 1]),
                '%Y.%m.%d'
            )
            # Получаем из кортежа конечную дату введенного ЗЗ
            # и преобразуем к нужному формату даты
            end_zs_data = datetime.datetime.strptime(
                str(birth_data.year) + '.' +
                str(zodiac_signs[zodiac_signs.index(zodiac_sign) + 1]),
                '%Y.%m.%d'
            )
            # Сравнение введенной даты с промежутком дат ЗЗ
            if beg_zs_data <= birth_data < end_zs_data:
                break
            else:
                print("\033[31mEnter CORRECR date of birth "
                    "(yyyy.mm.dd): ")

# Вернуть словарь
return {
    'name': name,
    'zodiac_sign': zodiac_sign,
    'birth': birth,
}

```

```

def display_people(people_list):
    # Заголовок таблицы.
    line = (f'{"+" + "-" * 15 + "+" + "-" * 12 + "+"}'
            f'{"-" * 15 + "+"}')
    print(line)
    print(f"|{'Name' :^15}|{'Birth ' :^12}|{'Zodiac_sign ' :^15}|")
    print(line)

    # Вывести данные о всех людях.
    for idx, man in enumerate(people_list):
        print(
            f'|{man.get("name", "") :^15}|'
            f'|{man.get("birth", "") :^12}|'
            f'|{man.get("zodiac_sign", "") :^15}|'
        )
        print(line)

def select_zz(com, people_list):
    # Разбить команду на части для выделения номера года.
    # Параметр maxsplit - int, сколько раз делить строку.
    while True:
        parts = com.split(' ', maxsplit=1)
        if len(parts) == 1:
            com = input("\033[31mEnter command "
                        "select and Zodiac_sign: ").lower()
            # Получить требуемый ЗЗ.
        else:
            break

    # Инициализировать счетчик.
    count = 0
    zz = parts[1]
    # Заголовок таблицы.
    line = (f'\033[0m{"+" + "-" * 12 + "+" + "-" * 12 + "+"}'
            f'{"-" * 15 + "+"}')
    print(line)
    print(
        f"|{'Name' :^12}|{'Birth ' :^12}"
        f"|{'Zodiac_sign ' :^15}|"
    )
    print(line)
    # Таблица с людьми
    for p in people_list:
        if zz == p.get('zodiac_sign'):
            count += 1
            print(
                f'|{p.get("name", "") :^12}|'
                f'|{p.get("birth", "") :^12}|'
                f'|{p.get("zodiac_sign", "") :^15}|'
            )
            print(line)

    # Если счетчик равен 0, то люди не найдены.
    if count == 0:
        print("Люди с заданным ЗЗ не найдены")

if __name__ == '__main__':
    # Список людей
    people = []

    # zodiac_signs = {
    #     'name': "овен",
    #     'data_beg': datetime.datetime.strptime("11.11", "%d.%m"),
    #     'data_end': "birth"
    # }

```

```

# print(zodiac_signs.get("data_beg"))

# Организовать бесконечный цикл запроса команд.
while True:
    # Запросить команду из терминала.
    # На Python с помощью ANSI-код можно делать цвет, фон и т.д.
    # \033[0m - сброс форматирования
    command = input("\033[0mEnter command: ").lower()

    # Выполнить действие в соответствии с командой.
    if command == 'exit':
        break

    # Добавить нового человека
    elif command == 'add':
        human = add_human()
        # Добавить словарь в список.
        people.append(human)

    # Отсортировать список в случае необходимости.
    # Сортировка по дате рождения (от старшего к младшим)
    if len(people) > 1:
        people.sort(key=lambda item: item.get('birth', ''))

    # Показать список людей
    elif command == 'show':
        display_people(people)

    # .startswith - Поиск строк с заданным началом строки
    elif command.startswith('select'):
        select_zz(command, people)

    # Список команд с описанием
    elif command == 'help':
        help_inf()

    # Неопознанная команда
    else:
        print("\033[31mUnknown command")

```

Рисунок 11.13 – Код программы

```

ind_1 x
"C:\Users\kuvsh\Desktop\СКОУ\2_3_семестр\Основы Программной Инженерии\Git\Py_L11\PyCharm\venv\Scripts\python.exe" "C:\Users\kuvsh\Desktop\СКОУ\2_3
Enter command: help
Список команд:

add - добавить человека;
select <33> - запросить людей с определенным 33;
show - показать список людей;
help - отобразить справку;
exit - завершить работу с программой.
Enter command: add
Enter name: Иванов И.И.
Enter zodiac sign: рак
Enter date of birth (yyyy.mm.dd): 2007.06.25
Enter CORRECR date of birth (yyyy.mm.dd):
Enter date of birth (yyyy.mm.dd): 2007.06.25
Enter command: add
Enter name: Петров И.П.
Enter zodiac sign: рыбы
Enter date of birth (yyyy.mm.dd): 2001.02.23
Incorrect data format
Enter date of birth (yyyy.mm.dd): 2001.02.23
Enter command: show
+-----+-----+-----+
| Name   | Birth  | Zodiac_sign |
+-----+-----+-----+
| Петров И.П. | 2001.02.23 | рыбы |
+-----+-----+-----+
| Иванов И.И. | 2007.06.25 | рак |
+-----+-----+-----+

```

```
Enter command: select
Enter command select and Zodiac_sign: select row
+-----+-----+
| Name | Birth | Zodiac_sign |
+-----+-----+
|Иванов И.И. | 2007.06.25 | рак |
+-----+-----+
Enter command: select row
+-----+-----+
| Name | Birth | Zodiac_sign |
+-----+-----+
|Петров И.П. | 2001.02.23 | рыбы |
+-----+-----+
Enter command: exit

Process finished with exit code 0
```

Рисунок 11.14 – Результат работы программы

19. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.
20. Выполните слияние ветки для разработки с веткой master/main.
21. Отправьте сделанные изменения на сервер GitHub.
22. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Вопросы для защиты работы:

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