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 ЛАБОРАТОРНОЙ РАБОТЕ №11
дисциплины «Программирование на python»

Выполнил:
Кожуховский Виктор Андреевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств вычислительной
техники и автоматизированных систем
», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Работа с функциями в языке Python

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

Методика и порядок выполнения работы

1. Изучил теоретический материал работы.

2.Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

3. Выполнил клонирование созданного репозитория.

4. Дополнил файл .gitignore необходимыми правилами для работы с IDE PyCharm.

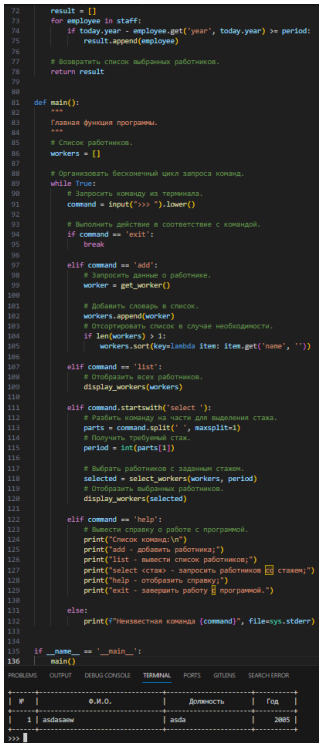
5. Организовал свой репозиторий в соответствие с моделью ветвления git-flow.

6. Создал проект в папке репозитория.

7. Проработал пример лабораторной работы. Зафиксировал изменения в репозитории.

```
#!/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(
                "№",
                "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 = []
```

Рисунок 1. Код примера



```
    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()
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  SEARCH ERROR

+----+--------------------------------+----------------------+----------+
| №  |             Ф.И.О.             |       Должность      |   Год    |
+----+--------------------------------+----------------------+----------+
|  1 | asdasasw                       | asda                 |     2005 |
+----+--------------------------------+----------------------+----------+
>>>
```

Рисунок 2. Вторая часть кода примера и вывод

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.

Порядок объявления функций до вызова неважен, т.к. в процессе интерпретации кода, интерпретатор Python последовательно проходит по коду и выполняет его.

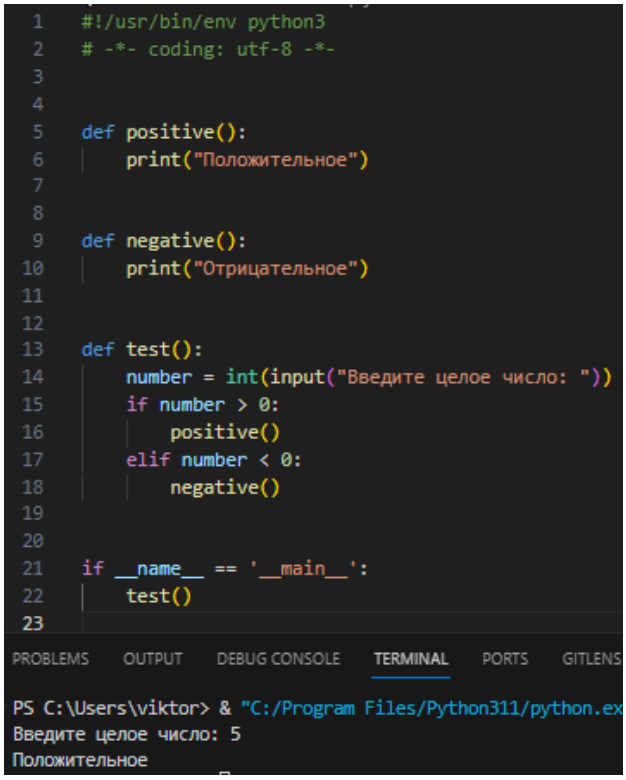


Рисунок 3. Код решения задачи с одним расположением функций до вызова

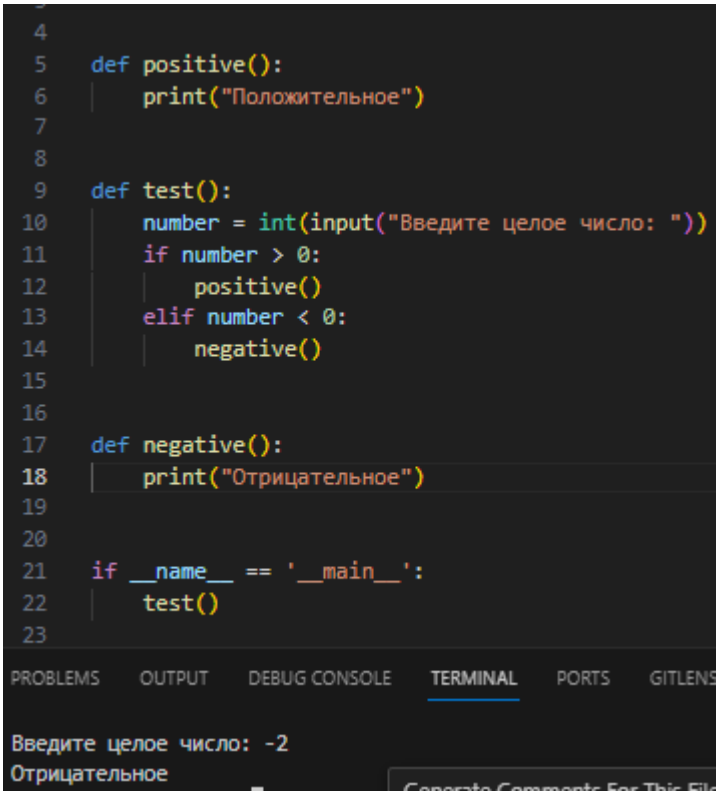


Рисунок 4. Код решения задачи с другим расположением функций до вызова

9. Зафиксировал сделанные изменения в репозитории.

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 . 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().

Рисунок 5. Код решения задачи 2

11. Зафиксировал сделанные изменения в репозитории.

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.

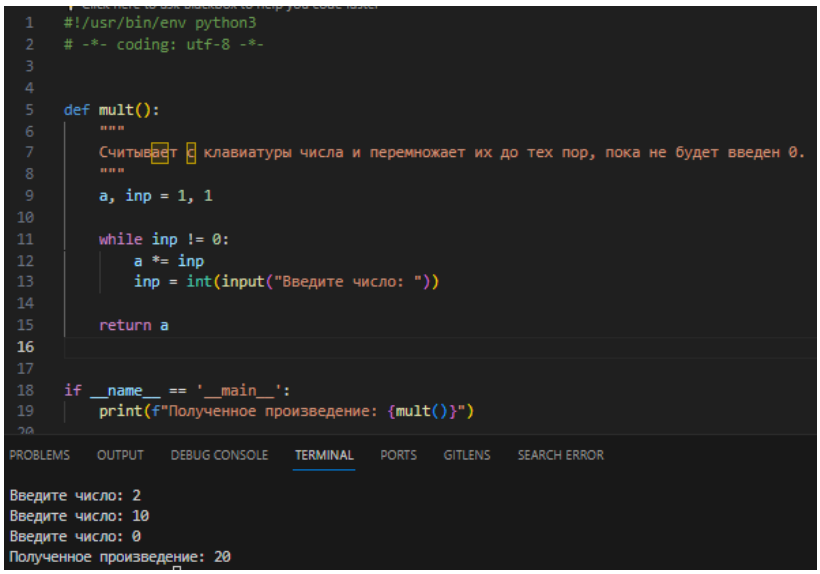


Рисунок 6. Код решения задачи 3

13. Зафиксировал сделанные изменения в репозитории.

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.

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


def get_input():
    """
    Запрашивает ввод с клавиатуры и возвращает в основную программу полученную строку.
    """
    return input("Введите данные: ")


def test_input(value):
    """
    Проверяет, можно ли переданное значение преобразовать к целому числу.
    """
    try:
        int(value)
        return True
    except ValueError:
        return False


def str_to_int(value):
    """
    Преобразовывает переданное значение к целочисленному типу.
    """
    return int(value)


def print_int(value):
    """
    Не выводит переданное значение на экран и ничего не возвращает.
    """
    print(value)


if __name__ == '__main__':
    user_input = get_input()

    if test_input(user_input):
        integer = str_to_int(user_input)
        print_int(integer)
    else:
        print("Введенные данные не могут быть преобразованы в целое число.")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  SEARCH ERROR

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" "c:/Users/viktor/Desktop/скфу/python/
Введите данные: 987
987
PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" "c:/Users/viktor/Desktop/скфу/python/
Введите данные: ы
Введенные данные не могут быть преобразованы в целое число.
```

Рисунок 7. Код решения задачи 4

15. Зафиксировал сделанные изменения в репозитории.

16. Привел в отчете скриншоты работы программ решения индивидуального задания.

Решить **индивидуальное** задание лабораторной работы 9, оформив каждую команду в виде отдельной функции.

Использовать словарь, содержащий следующие ключи: фамилия, имя; знак Зодиака; дата рождения (массив из трех чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по знакам Зодиака; вывод на экран информации о людях, родившихся в месяц, значение которого введено с клавиатуры; если таких нет, выдать на дисплей соответствующее сообщение.

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

import sys
from datetime import datetime


def add_person(people):
    """
    Добавление нового человека в список.
    Список сортируется по знаку зодиака после добавления нового элемента.
    """
    name = input("Фамилия: ")
    surname = input("Имя: ")
    date_of_birth = datetime.strptime(
        input("Введите дату рождения (в формате ДД.ММ.ГГГГ через точку): "), '%d.%m.%Y')
    zodiac_sign = input("Знак зодиака: ")

    person = {
        'name': name,
        'surname': surname,
        'date_of_birth': date_of_birth,
        'zodiac_sign': zodiac_sign
    }

    people.append(person)
    people.sort(key=lambda item: item.get('zodiac_sign', ''))


def list_people(people):
    """
    Вывод таблицы людей.
    """
    line = '+-{}-+-{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 20,
        '-' * 20,
        '-' * 15,
        '-' * 13
    )
    print(line)
    print(
        '| {:^4} | {:^20} | {:^20} | {:^15} | {:^12} |'.format(
            "№",
            "Имя",
            "Фамилия",
            "Знак Зодиака",
            "Дата рождения"
        )
    )
    print(line)

    for idx, person in enumerate(people, 1):
        birth_date_str = person.get('date_of_birth').strftime('%d.%m.%Y')
        print(
            '| {:^4} | {:<20} | {:<20} | {:<15} | {:<13} |'.format(
                idx,
                person.get('name', ''),
                person.get('surname', ''),
                person.get('zodiac_sign', ''),
                birth_date_str
            )
        )

    print(line)


def select_people(people, month):
    """
    Вывести список людей, родившихся в заданном месяце.
    """
    count = 0
```

Рисунок 8. Код индивидуального задания

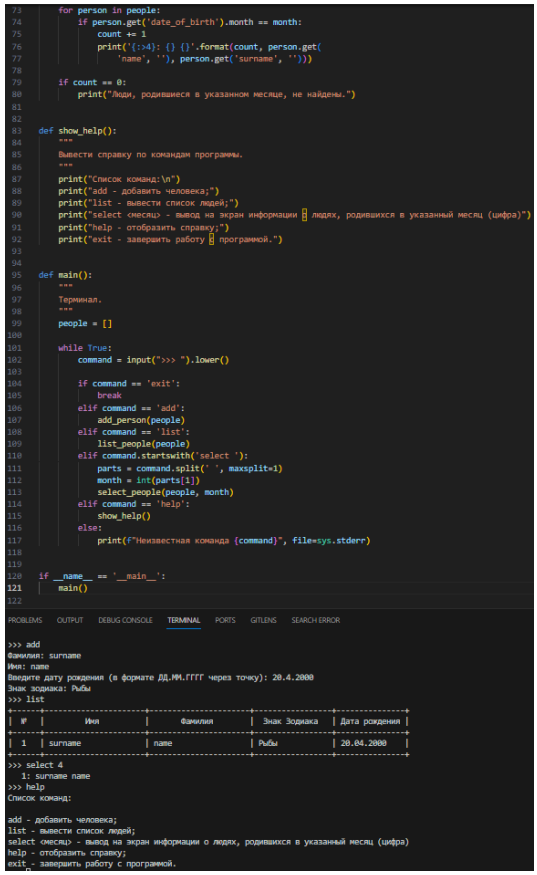


Рисунок 9. Код индивидуального задания и его вывод

17. Зафиксировал сделанные изменения в репозитории.

18. Добавил отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксировал изменения.

19. Выполнил слияние ветки для разработки с веткой main / master.

20. Отправил сделанные изменения на сервер GitHub.

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.

2. Каково назначение операторов def и return?

В языке программирования Python функции определяются с помощью оператора def .

def countFood():

```
a = int(input())
b = int(input())
print("Всего", a+b, "шт.")
```

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?

Перечислением через запятую.

5. Какие существуют способы передачи значений в функцию?

- По позиции: Передача аргументов в том порядке, в котором они определены в функции.

```
def func(a, b):
    pass
func(1, 2)
```

- По ключу: Указание имени аргумента при вызове функции, что позволяет изменять порядок передачи.

```
def func(a, b):
    pass
func(b=2, a=1)
```

- Значения по умолчанию: Функции могут иметь аргументы со значениями по умолчанию, которые используются, если аргумент не был передан.

```
def func(a, b=2):
    pass
func(1)
```

- Переменное число аргументов: Использование `*args` для передачи неопределенного количества позиционных аргументов, и `kwargs` для передачи неопределенного количества аргументов по ключу.

```
def func(*args, kwargs):
    pass
func(1, 2, three=3, four=4)
```

6. Как задать значение аргументов функции по умолчанию?

Однако в Python у функций бывают параметры, которым уже присвоено значение по-умолчанию. В таком случае, при вызове можно не передавать соответствующие этим параметрам аргументы. Хотя можно и передать. Тогда значение по умолчанию заменится на переданное.

```
def cylinder(h, r=1):
side = 2 * math.pi * r * h
circle = math.pi * r2
full = side + 2 * circle
return full
```

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?

-Однострочные docstrings должны быть в тройных кавычках и занимать одну строку.

```
def func():
    """Краткое описание функции."""
```

-Многострочные docstrings начинаются с однострочного описания, за ним пустая строка, затем подробное описание. Закрывающие кавычки на отдельной строке.

```
def func():
    """
    Краткое описание.

    Детальное описание функции здесь.
    """
```

-Каждый модуль, класс и функция должны иметь docstrings.

-Содержание docstrings должно быть описательным и сформулировано так, чтобы описывать действие или результат, а не саму реализацию.

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

Одиночные строки документации предназначены для действительно очевидных случаев. Они должны умещаться на одной строке. Однострочная строка документации не должна быть "подписью" параметров функции / метода (которые могут быть получены с помощью интроспекции).

Этот тип строк документации подходит только для C функций (таких, как встроенные модули), где интроспекция не представляется возможной. Тем не менее, возвращаемое значение не может быть определено путем интроспекции.

Многострочные строки документации состоят из однострочной строки документации с последующей пустой строкой, а затем более подробным описанием. Первая строка может быть использована автоматическими

средствами индексации, поэтому важно, чтобы она находилась на одной строке и была отделена от остальной документации пустой строкой. Первая строка может быть на той же строке, где и открывающие кавычки, или на следующей строке. Вся документация должна иметь такой же отступ, как кавычки на первой строке