

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ  
ФЕДЕРАЦИИ**

**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций  
«Работа с функциями в языке Python»**

**Отчет по лабораторной работе № 2.8  
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1

Халимендик Я. Д. « » 2022г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2022

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

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---

**Owner \*** **Repository name \***

Yana-Kh / Lab-11-OPJ

Great repository names are short and memorable. Need inspiration? How about [shiny-garbanzo?](#)

**Description** (optional)

---

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

---

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

**.gitignore template:** Python

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

**License:** MIT License

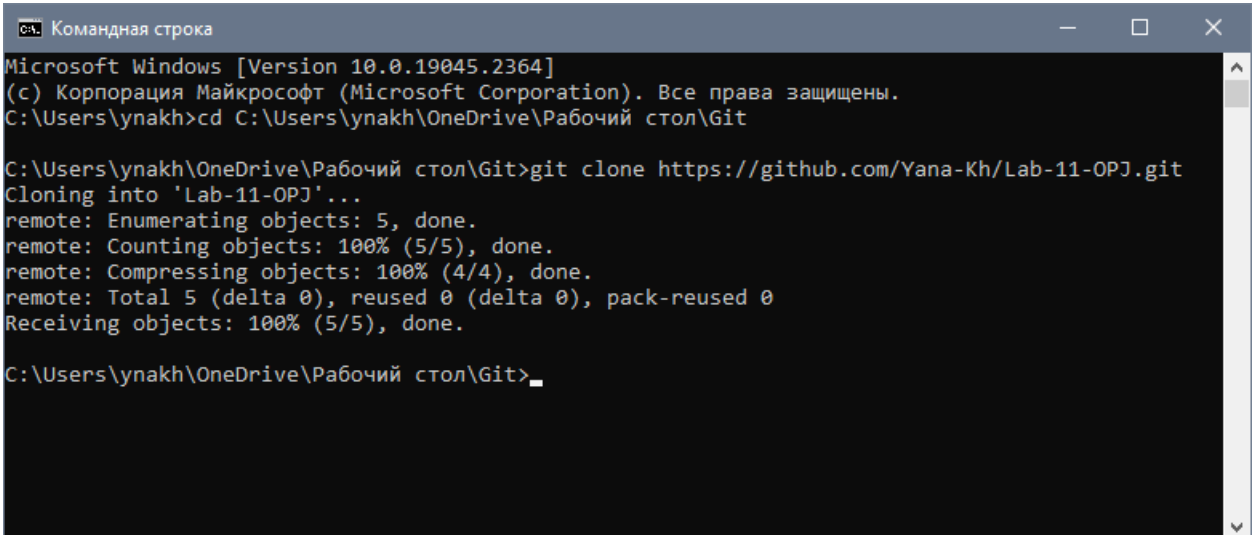
This will set **main** as the default branch. Change the default name in your [settings](#).

You are creating a public repository in your personal account.

**Create repository**

Рисунок 1 – Создание репозитория

### 3. Выполните клонирование созданного репозитория.



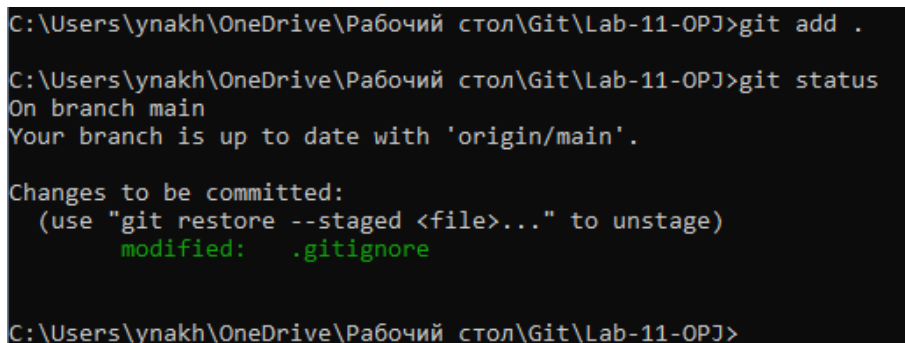
```
cmd. Командная строка
Microsoft Windows [Version 10.0.19045.2364]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
C:\Users\ynakh>cd C:\Users\ynakh\OneDrive\Рабочий стол\Git

C:\Users\ynakh\OneDrive\Рабочий стол\Git>git clone https://github.com/Yana-Kh/Lab-11-OPJ.git
Cloning into 'Lab-11-OPJ'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

C:\Users\ynakh\OneDrive\Рабочий стол\Git>
```

Рисунок 2 – Клонирование репозитория

### 4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.



```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-11-OPJ>git add .

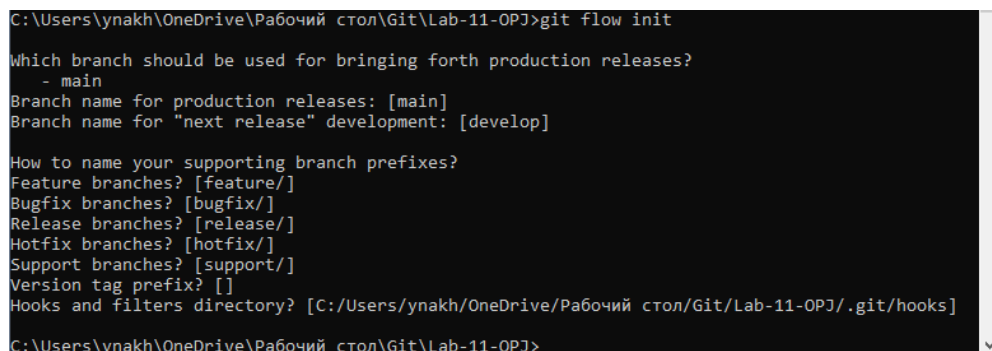
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-11-OPJ>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   .gitignore

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-11-OPJ>
```

Рисунок 3 – Дополнение файла .gitignore

### 5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.



```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-11-OPJ>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:/Users/ynakh/OneDrive/Рабочий стол/Git/Lab-11-OPJ/.git/hooks]

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-11-OPJ>
```

Рисунок 4 – Организация репозитория в соответствии с моделью git-flow

## 6. Создайте проект PyCharm в папке репозитория.

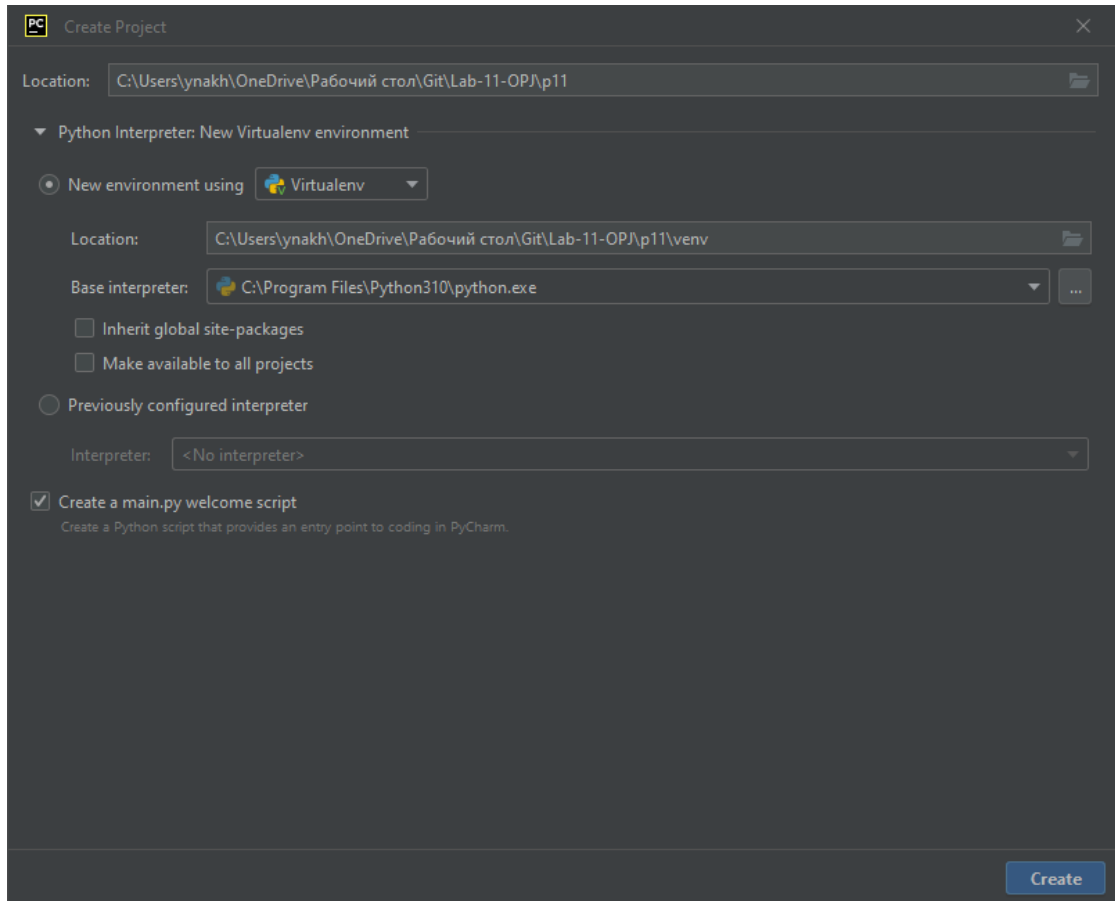


Рисунок 5 – Создание проекта PyCharm в папке репозитория

7. Проработайте примеры лабораторной работы. Зафиксируйте изменения в репозитории.

Пример 1. Определить результат выполнения операций над множествами. Считать элементы множества строками.

Код:

```
#!/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 = []
    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()

```

```

>>> 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 ИИ
Должность? программист
Год поступления? 2019
>>> add
Фамилия и инициалы? Петров ПР
Должность?
Год поступления? 1995
>>> add
Фамилия и инициалы? Романов ФС
Должность? менеджер
Год поступления? 2008
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Иванов ИИ | программист | 2019 |
+-----+-----+-----+-----+
| 2 | Петров ПР | | 1995 |
+-----+-----+-----+-----+
| 3 | Романов ФС | менеджер | 2008 |
+-----+-----+-----+-----+
>>> select 5
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Петров ПР | | 1995 |
+-----+-----+-----+-----+
| 2 | Романов ФС | менеджер | 2008 |
+-----+-----+-----+-----+
>>> _

```

Рисунок 6 – Результат работы программы

8. Решите 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".

Код:

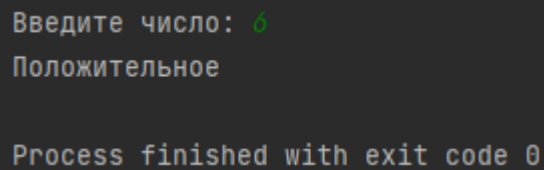
```
# !/usr/bin/env python3
# -*- coding: utf-8 -*-

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

def positive(a):
    print("Положительное")

def negative(a):
    print("Отрицательное")

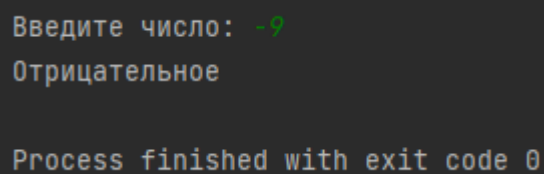
if __name__ == '__main__':
    test()
```



```
Введите число: 6
Положительное

Process finished with exit code 0
```

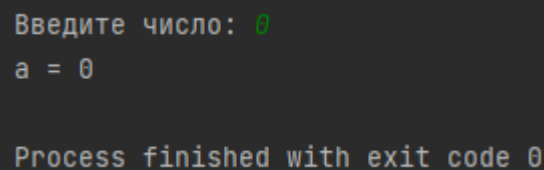
Рисунок 7 – Результат работы программы



```
Введите число: -9
Отрицательное

Process finished with exit code 0
```

Рисунок 8 – Результат работы программы



```
Введите число: 0
a = 0

Process finished with exit code 0
```

Рисунок 9 – Результат работы программы



9. Зафиксируйте сделанные изменения в репозитории.

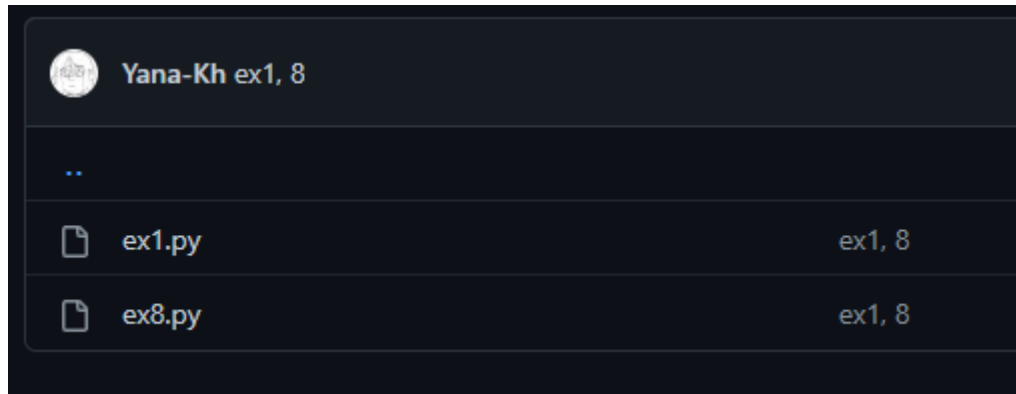


Рисунок 10 – Фиксация изменений в репозитории

10. Решите 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  $S_{\text{бок}} = 2\pi r h$ , или полную площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции `circle()`.

Код:

```
Введите радиус: 4
Введите высоту: 3
S боковой поверхности (1), S цилиндра (2): 1
75.39822368615503

Process finished with exit code 0
```

Рисунок 12 – Результат работы программы

```
Введите радиус: 4
Введите высоту: 3
S боковой поверхности (1), S цилиндра (2): 2
175.92918860102841

Process finished with exit code 0
```

Рисунок 13 – Результат работы программы

11. Зафиксируйте сделанные изменения в репозитории.

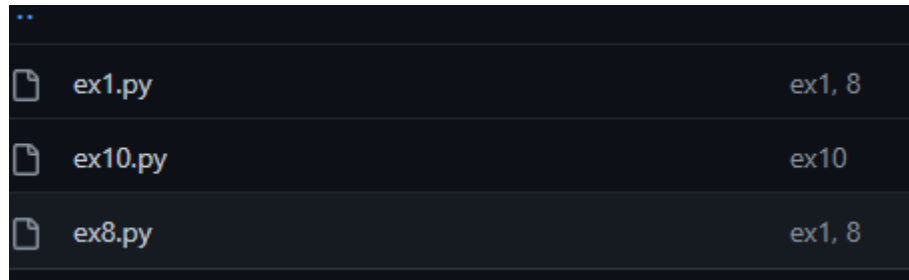


Рисунок 14 – Фиксация изменений в репозитории

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.

Код:

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

def composition():
    n = 1
    print("Признак окончания: '0'")
    while True:
        a = int(input("Введите число: "))
        if a == 0:
            return print(f"Произведение чисел: {n}")
        else:
            n *= a

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

```
Признак окончания: '0'
Введите число: 4
Введите число: 5
Введите число: -3
Введите число: 0
Произведение чисел: -60

Process finished with exit code 0
```

Рисунок 15 – Результат работы программы

```
Признак окончания: '0'  
Введите число: 4  
Введите число: 9  
Введите число: -3  
Введите число: -6  
Введите число: 0  
Произведение чисел: 648  
  
Process finished with exit code 0
```

Рисунок 16 – Результат работы программы

13. Зафиксируйте изменения в репозитории.





 ex1.py	ex1, 8
 ex10.py	ex10
 ex12.py	ex12
 ex8.py	ex1, 8

Рисунок 17 – Фиксация изменений 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.

Код:

```
Введите строку: 123
123

Process finished with exit code 0
```

Рисунок 18 – Результат работы программы

```
Введите строку: 76.0
Строка не является целым числом

Process finished with exit code 0
```

Рисунок 18 – Результат работы программы

```
Введите строку: привет
Строка не является целым числом

Process finished with exit code 0
```

Рисунок 18 – Результат работы программы

15. Зафиксируйте изменения в репозитории.





 ex10.py	ex10
 ex12.py	ex12
 ex14.py	ex14
 ex8.py	ex1, 8

Рисунок 17 – Фиксация изменений в репозитории

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

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

Вариант 32

Код:

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

import sys
import datetime

def get_human():
    """
    Запросить данные о работнике.
    """
    name = input("Фамилия и имя: ")
    phone = int(input("Номер телефона: +7"))
    bday = list(map(int, input("Дата рождения: ").split('.')))
    d_bday = datetime.date(bday[2], bday[1], bday[0])

    # Вернуть словарь.
    return {
        'name': name,
        'phone': phone,
        'birthday': d_bday
    }

def display_human(staff):
    """
    Отобразить список работников.
    """
    # Проверить, что список работников не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 15,
            '-' * 15
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^15} | {:^15} |'.format(
                "№",
                "Ф и И.",
                "Телефон",
                "День рождения"
            )
        )
        print(line)

    # Вывести данные о всех сотрудниках.
```

```

        for idx, human in enumerate(staff, 1):
            print(
                f'| {idx:>4} | '
                f'| {human.get("name", ""):<30} | '
                f'| {human.get("phone", 0):<15} | '
                f'| {human.get("birthday")} | '
            )
            print(line)

    else:
        print("Список пуст.")

def find_human(staff, fname):
    """
    Выбрать работников с заданным стажем.
    """
    # Сформировать список людей.
    result = []
    for h in staff:
        if fname in str(h.values()):
            result.append(h)

    # Проверка на наличие записей
    if len(result) == 0:
        return print("Запись не найдена")

    # Возвратить список выбранных работников.
    return result

def main():
    """
    Главная функция программы.
    """
    # Список работников.
    people = []

    # 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.
            human = get_human()

            # Добавить словарь в список.
            people.append(human)
            # Отсортировать список в случае необходимости.
            if len(people) > 1:
                people.sort(key=lambda item: item.get('phone', 0))

        elif command == 'list':
            # Отобразить всех работников.
            display_human(people)

        elif command == 'find':
            f = input('Введите фамилию: ')

```

```

# Выбрать людей с заданной фамилией.
selected = find_human(people, f)
# Отобразить выбранных работников.
display_human(selected)

elif command == 'help':
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("add - добавить человека;")
    print("list - вывести список людей;")
    print("find - найти человека по фамилии;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")

else:
    print(f"Неизвестная команда {command}", file=sys.stderr)

if __name__ == '__main__':
    main()

```

```

>>> add
Фамилия и имя: Иванов Иван
Номер телефона: +79184351212
Дата рождения: 09.08.2003
>>> add
Фамилия и имя: Петров Василий
Номер телефона: +79098005611
Дата рождения: 12.12.2000
>>> add
Фамилия и имя: Смирнов Егор
Номер телефона: +79612340090
Дата рождения: 30.08.1995
>>> дшые
Неизвестная команда дшые
>>> list
+-----+-----+-----+-----+
| № |           Ф и И.           | Телефон | День рождения |
+-----+-----+-----+-----+
|  1 | Петров Василий           | 9098005611 | 2000-12-12 |
+-----+-----+-----+-----+
|  2 | Иванов Иван              | 9184351212 | 2003-08-09 |
+-----+-----+-----+-----+
|  3 | Смирнов Егор             | 9612340090 | 1995-08-30 |
+-----+-----+-----+-----+
>>> find
Неизвестная команда find
>>> find
Введите фамилию: Иванов
+-----+-----+-----+-----+
| № |           Ф и И.           | Телефон | День рождения |
+-----+-----+-----+-----+
|  1 | Иванов Иван              | 9184351212 | 2003-08-09 |
+-----+-----+-----+-----+
>>> find
Введите фамилию: Халимендик
Запись не найдена

```

Рисунок 18 – Результат работы программы

17. Зафиксируйте сделанные изменения в репозитории.







 ex1.py	ex1
 ex10.py	ex10
 ex12.py	ex12
 ex14.py	ex14
 ex8.py	ex1, 8
 id.py	id

Рисунок 10 – Фиксация изменений в репозитории



## Вопросы для защиты работы

### 1. Каково назначение функций в языке программирования Python?

Функция – это средство (способ) группирования фрагментов программного кода таким образом, что этот программный код может вызываться многократно с помощью использования имени функции. Использование функций в программах на Python даёт следующие взаимосвязанные преимущества: избежание повторения одинаковых фрагментов кода в разных частях программы; уменьшение избыточности исходного кода программы. Как следствие, уменьшение логических ошибок программирования; улучшенное восприятие исходного кода программы в случаях, где вместо блоков многочисленных инструкций (операторов) вызываются имена готовых протестированных функций. Это, в свою очередь, также уменьшает количество ошибок; упрощение внесения изменений в повторяемых блоках кода, организованных в виде функций. Достаточно внести изменения только в тело функции, тогда во всех вызовах данной функции эти изменения будут учтены; с помощью функций удобно разбивать сложную систему на более простые части. Значит, функции – удобный способ структурирования программы; уменьшение трудозатрат на программирование, а, значит, повышение производительности работы программиста.

### 2. Каково назначение операторов `def` и `return`?

Оператор `def`, выполняемый внутри определения функции, определяет локальную функцию, которая может быть возвращена или передана. Свободные переменные, используемые во вложенной функции, могут обращаться к локальным переменным функции, содержащей `def`.

Оператор `return [выражение]` возвращает результат из функции. Оператор `return` без аргументов аналогичен `return None`

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

Области видимости определяют, в какой части программы мы можем работать с той или иной переменной, а от каких переменная «скрыта».

Так глобальные переменные доступны в любой точке программы, а локальные переменные, только в функциях, где они объявлены.

4. Как вернуть несколько значений из функции Python?

С помощью оператора `return`. Чтобы вернуть несколько значений, нужно написать их через запятую.

5. Какие существуют способы передачи значений в функцию?

Существует два способа передачи параметров в функцию: по значению и по адресу. При передаче по значению на месте формальных параметров записываются имена фактических параметров. При вычислении функции в стек заносятся копии значений фактических параметров, и операторы функции работают с этими копиями.

6. Как задать значение аргументов функции по умолчанию?

В Python аргументам функции можно присваивать значения по умолчанию, используя оператор присваивания `=`.

7. Каково назначение `lambda`-выражений в языке Python?

Лямбда-выражения на Python - конструкторы простых безымянных однострочных функций. Могут быть использованы везде, где требуется.

8. Как осуществляется документирование кода согласно PEP257?

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 PER описывает соглашения, а не правила или синтаксис.

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

Одиночные строки документации предназначены для действительно очевидных случаев.

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.