МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО
ОБРАЗОВАНИЯРОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №11 по дисциплине «Основы программной инженерии»

Выполнил студент
2 курса, группы ПИЖ-б-о-20-1
Тотубалина С.С.
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г

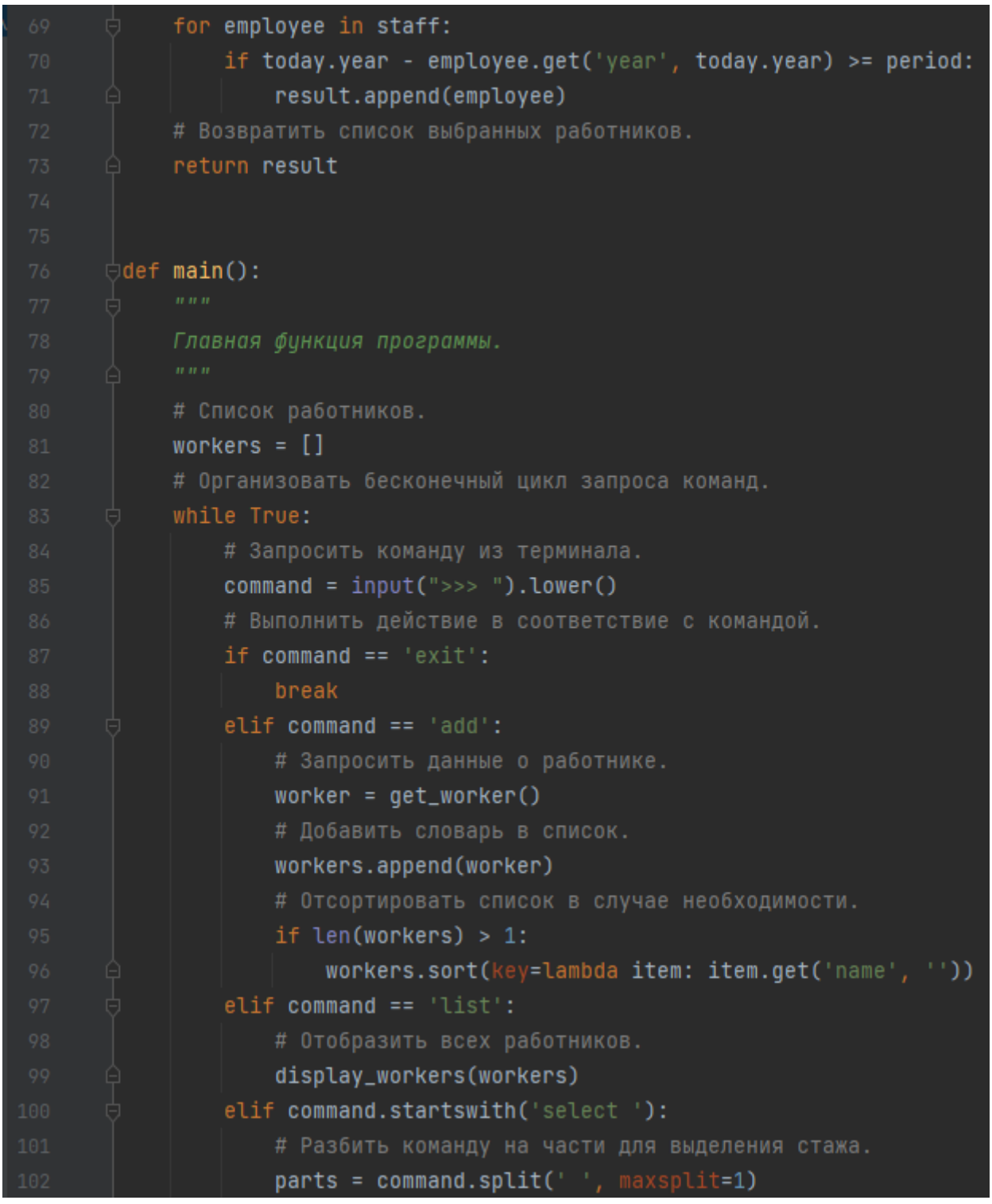
Ход работы

```
#!/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
```

Рис. 1 – код программы lab.11_ex.1.py

```
        )
        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 = []
```

Рис. 2 – код программы lab.11_ex.1.py



```
    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)
```

Рис. 3 – код программы lab.11_ex.1.py

```
            # 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()
```

Рис. 4 – код программы lab.11_ex.1.py

```
>>> add
Фамилия и инициалы? Тотубалина С. С.
Должность? Программист
Год поступления? 2020
>>> list
+------+--------------------------------+----------------------+----------+
|  No  |             Ф.И.О.             |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Тотубалина С. С.               | Программист          |     2020 |
+------+--------------------------------+----------------------+----------+
>>> select 2
+------+--------------------------------+----------------------+----------+
|  No  |             Ф.И.О.             |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Тотубалина С. С.               | Программист          |     2020 |
+------+--------------------------------+----------------------+----------+
>>> selest 10
>>> Неизвестная команда selest 10
select 10
Список работников пуст.
>>> exit

Process finished with exit code 0
```

Рис. 5 – результат работы программы программы lab.11_ex.1.py

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


def test():
    number = int(input("Введите целое число: "))
    if number > 0:
        positive()
    elif number < 0:
        negative()
    else:
        print("Число равно нулю.")


def positive():
    print("Число положительное.")


def negative():
    print("Число отрицательное.")


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

Рис. 6 – код программы lab.11_ex.2.py

```
Введите целое число: 3
Число положительное.

Process finished with exit code 0
```

Рис. 7 – пример работы программы lab.11_ex.2.py при number = 3

```
Введите целое число: -5
Число отрицательное.

Process finished with exit code 0
```

Рис. 8 – пример работы программы lab.11_ex.2.py при number = -5

```
Введите целое число: 0
Число равно нулю.

Process finished with exit code 0
```

Рис. 9 – пример работы программы lab.11_ex.2.py при number = 0

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

from math import pi


def cylinder():

    def circle(rad):
        return pi * rad * rad

    r = int(input("Введите радиус: "))
    h = int(input("Введите высоту: "))
    choose = input("Площадь боковой поверхности цилиндра - a\n"
                   "Полная площадь цилиндра - b\n"
                   "a/b: ")
    if choose == 'a':
        print(f"Площадь боковой поверхности цилиндра = {2 * pi * r * h}")
    else:
        print(f"Полная площадь цилиндра = {2 * pi * r * h + 2 * circle(r)}")


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

Рис. 10 – код программы lab.11_ex.3.py

```
Введите радиус: 2
Введите высоту: 7
Площадь боковой поверхности цилиндра - a
Полная площадь цилиндра - b
a/b: a
Площадь боковой поверхности цилиндра = 87.96459430051421

Process finished with exit code 0
```

Рис. 11 – результат работы программы lab.11_ex.3.py с выбором a

```
Введите радиус: 2
Введите высоту: 7
Площадь боковой поверхности цилиндра - a
Полная площадь цилиндра - b
a/b: b
Полная площадь цилиндра = 113.09733552923255

Process finished with exit code 0
```

Рис. 12 – результат работы программы lab.11_ex.3.py с выбором b

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

def multi():
    number = int(input("Введите число: "))
    result = 1
    if number == 0:
        return None
    while number != 0:
        result *= number
        number = int(input("Введите число: "))
    return result


if __name__ == '__main__':
    print(f"Вызов функции и ее результата = {multi()}")
```

Рис. 13 – код программы lab.11_ex.4.py

```
Введите число: 0
Вызов функции и ее результата = None

Process finished with exit code 0
```

Рис. 14 – результат работы программы lab.11_ex.4.py

```
Введите число: 1
Введите число: 2
Введите число: 3
Введите число: 0
Вызов функции и ее результата = 6

Process finished with exit code 0
```

Рис. 15 – результат работы программы lab.11_ex.4.py

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

def get_input():
    return input()


def test_input(string):
    return string.isdigit()


def str_to_int(string):
    return int(string)


def print_int(integer):
    print(integer)


def main():
    data = get_input()
    if test_input(data):
        print_int(str_to_int(data))


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

Рис. 16 – код программы lab.11_ex.5.py

```
3
3

Process finished with exit code 0
```

Рис. 17 – результат работы программы программы lab.11_ex.5.py

```
qwertyuio

Process finished with exit code 0
```

Рис. 18 – результат работы программы программы lab.11_ex.5.py

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

import sys


def get_flight():
    """
    Запросить данные о полёте
    """
    flight_destination = input("Введите название пункта назначения ")
    flight_number = input("Введите номер рейса ")
    airplane_type = input("Введите тип самолета ")
    return {
        'flight_destination': flight_destination,
        'flight_number': flight_number,
        'airplane_type': airplane_type,
    }


def display_flights(flights):
    """
     Отобразить список рейсов
    """
    if flights:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 15
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                "No",
                "Пункт назначения",
                "Номер рейса",
                "Тип самолета"
            )
        )
        print(line)
```

Рис. 19 – код программы individual_11.py (Вариант №22)

```
        for idx, flight in enumerate(flights, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:<15} |'.format(
                    idx,
                    flight.get('flight_destination', ''),
                    flight.get('flight_number', ''),
                    flight.get('airplane_type', 0)
                )
            )
        print(line)

    else:
        print("Список рейсов пуст")


def select_flights(flights, airplane_type):
    """
    Выбрать рейсы самолётов заданного типа
    """
    count = 0
    res = []
    for flight in flights:
        if flight.get('airplane_type') == airplane_type:
            count += 1
            res.append(flight)
    if count == 0:
        print("рейсы не найдены")

    return res


def main():
    """
    Главная функция программы
    """
    flights = []
    while True:
        command = input(">>> ").lower()
        if command == 'exit':
            break

        elif command == 'add':
```

Рис. 20 – код программы individual_11.py (Вариант №22)

```
            flight = get_flight()
            flights.append(flight)
            if len(flights) > 1:
                flights.sort(
                    key=lambda item:
                    item.get('flight_destination', ''))

        elif command == 'list':
            display_flights(flights)

        elif command.startswith('select '):
            parts = command.split(' ', maxsplit=1)
            airplane_type = (parts[1].capitalize())
            print(f"Для типа самолета {airplane_type}:")
            selected = select_flights(flights, airplane_type)
            display_flights(selected)

        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить рейс;")
            print("list - вывести список всех рейсов;")
            print("select <тип самолета> - запросить рейсы указанного типа "
                  "самолета;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")
        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)


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

Рис. 21 – код программы individual_11.py (Вариант №22)

```
>>> add
Введите название пункта назначения Армавир
Введите номер рейса 5
Введите тип самолета Пассажирский
>>> list
+------+--------------------------------+----------------------+-----------------+
|  No  |        Пункт назначения        |     Номер рейса      |  Тип самолета   |
+------+--------------------------------+----------------------+-----------------+
|    1 | Армавир                        | 5                    | Пассажирский    |
+------+--------------------------------+----------------------+-----------------+
>>>
```

Рис. 22 – код программы individual_11.py (Вариант №22)

Ответы на вопросы:

1. Каково назначение функций в языке программирования Python?

Функция 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. 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?

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. К глобальной переменной можно обратиться из локальной области видимости. К локальной переменной нельзя обратиться из глобальной области видимости, потому что локальная переменная существует только в момент выполнения тела функции.

4. Как вернуть несколько значений из функции Python?

В Питоне позволительно возвращать из функции несколько объектов, перечислив их через запятую после команды return.

5. Какие существуют способы передачи значений в функцию?

С помощью так называемых параметров, которые указываются в скобках в заголовке функции. Количество параметров может быть любым.

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.

6. Как задать значение аргументов функции по умолчанию?

def do_smth(a, b=2) # Значение по умолчанию b = 2

7. Каково назначение lambda-выражений в языке Python?

Интересный синтаксис, позволяющий определять небольшие однострочные функции на лету. lambda – это выражение, а не инструкция. По этой причине ключевое слово lambda может появляться там, где синтаксис языка Python не позволяет использовать инструкцию def, – внутри литералов или в вызовах функций.

8. Как осуществляется документирование кода согласно PEP257?

- Тройные кавычки используются даже если строка помещается на одной линии. Это облегчает последующее расширение документации.

- Закрывающие кавычки находятся на той же строке, что и открывающие. Для однострочных docstring это выглядит лучше.
- Ни до, ни после документации не пропускаются строки. Код пишется сразу же на следующей линии
- Документационная строка — это «фраза», заканчивающаяся точкой. Она описывает эффект функции или метода в командном тоне
- Однострочная документация НЕ должна быть простой «подписью», повторяющей параметры функции/метода

Многострочные:

- Многострочные документации состоят из сводной строки (summary line) имеющей такую же структуру, как и однострочный docstring, после которой следует пустая линия, а затем более сложное описание.
- Оставляйте пустую строку после всех документаций (однострочных или многострочных), которые используются в классе;
- Документация скрипта (автономной программы) представляет из себя сообщение «о правильном использовании» и возможно будет напечатано, когда скрипт вызовется с неверными или отсутствующими аргументами
- Документация модуля должна обычно содержать список классов, исключений и функций (и любых других важных объектов), которые экспортируются при помощи библиотеки, а также однострочное пояснение для каждого из них.
- Документация функции или метода должна описывать их поведение, аргументы, возвращаемые значения, побочные эффекты, возникающие исключения и ограничения на то, когда они могут быть вызваны.
- Документация класса должна обобщать его поведение и перечислять открытые методы, а также переменные экземпляра.
- Если класс является потомком и его поведение в основном наследуется от основного класса, в его документации необходимо упомянуть об этом и описать возможные различия.

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

Одиночные строки документации предназначены для действительно очевидных случаев. Они должны умещаться на одной строке.

Многострочные строки документации состоят из однострочной строки документации с последующей пустой строкой, а затем более подробным описанием.