МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ 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:

Мамонтов Д.В.,

2 курс, группа ПИЖ-б-о-20-1,

Проверил:

Доцент кафедры инфокоммуникаций,

Воронкин Р.А.

Ставрополь, 2022 г

**ХОД РАБОТЫ**

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – код программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – код программы (продолжение)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – код программы (продолжение)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – код программы (конец)

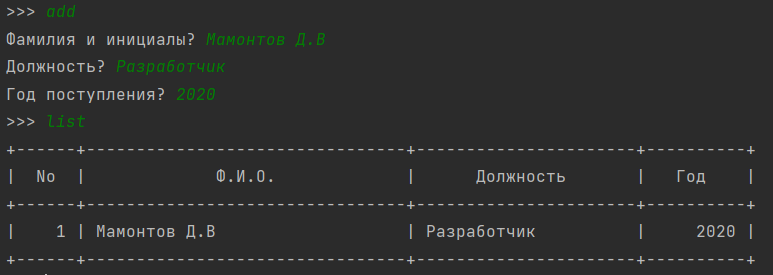


Рисунок 5 – результат работы программы



Рисунок 6 – код программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – результат работы программы при number = 3

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – результат работы программы при number = -5

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – результат работы программы при number = 0

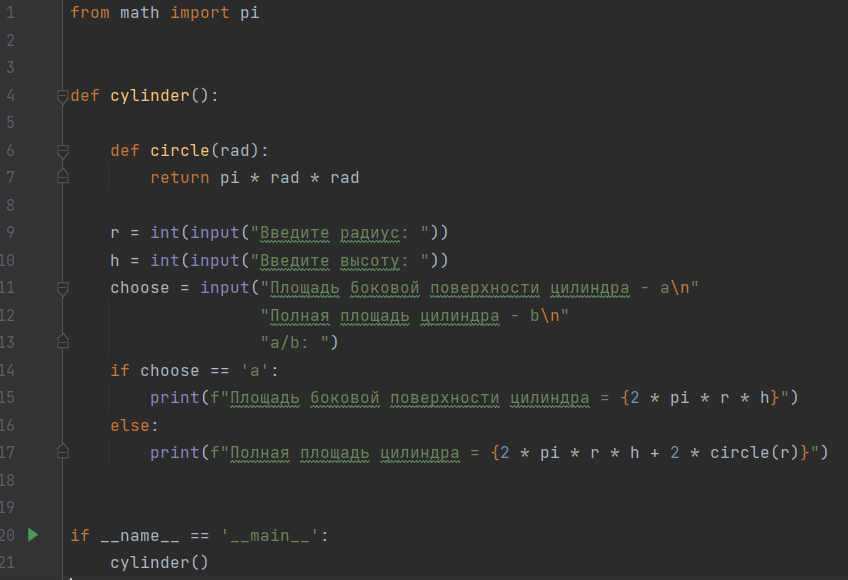


Рисунок 10 – код программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11 – результат работы программы с выбором а

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12 – результат работы программы с выбором b

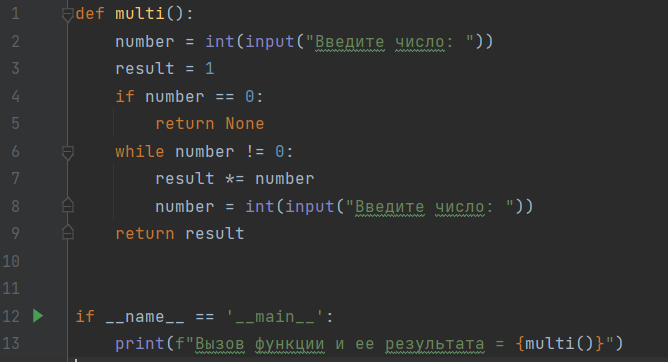


Рисунок 13 – код программы

Изображение выглядит как текст

Автоматически созданное описание  
рисунок 14 – результат работы программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 15 – результат работы программы

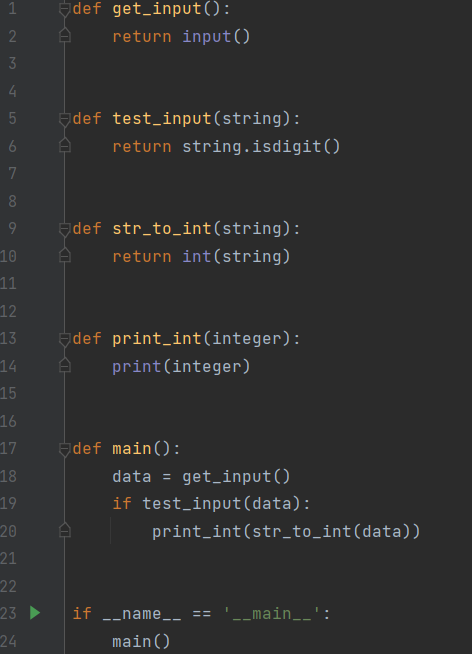


Рисунок 16 – код программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 17 – результат работы программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 18 – результат работы программы

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

Задание:

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

Код:

import sys  
  
  
def get\_student():  
 # Запросить данные о человеке.  
 name = input("Фамилия и имя? ")  
 phone = input("Номер телефона? ")  
 birth = input("Дата рождения? ")  
 # Создать словарь.  
 return {  
 'name': name,  
 'phone': phone,  
 'birth': birth,  
 }  
  
  
def display\_students(staff):  
 if staff:  
 # Заголовок таблицы.  
 line = '+-{}-+-{}-+-{}-+-{}-+'.format(  
 '-' \* 4,  
 '-' \* 30,  
 '-' \* 20,  
 '-' \* 13  
 )  
 print(line)  
 print(  
 '| {:^4} | {:^30} | {:^20} | {:^13} |'.format(  
 "No",  
 "Фамилия и имя",  
 "Номер телефона",  
 "Дата рождения"  
 )  
 )  
 print(line)  
 # Вывести данные о всех людях.  
 for idx, student in enumerate(staff, 1):  
 print(  
 '| {:>4} | {:<30} | {:<20} | {:>13} |'.format(  
 idx,  
 student.get('name', ''),  
 student.get('phone', ''),  
 student.get('birth', '')  
 )  
 )  
 print(line)  
 else:  
 print("Список людей пуст")  
  
  
def select\_student(staff, phone):  
 # Проверить сведения людей из списка.  
 result = ""  
 found = False  
 for one in staff:  
 if one.get('phone', '') == phone:  
 result = one.get('name', '')  
 found = True  
 if not found:  
 return "Нет человека с таким номером."  
 else:  
 return result  
  
  
def date\_key(birth):  
 data = birth.split(".")  
 return data[2], data[1], data[0]  
  
  
def main():  
 # Список людей.  
 students = []  
 # 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 человеке  
 student = get\_student()  
 # Добавить словарь в список  
 students.append(student)  
 # Отсортировать список в случае необходимости  
 if len(students) > 1:  
 students.sort(key=lambda item: date\_key(item.get('birth', '')))  
 elif command == 'list':  
 # Отобразить всех людей  
 display\_students(students)  
 elif command.startswith('phone '):  
 # Разбить команду на части для выделения номера телефона.  
 parts = command.split(' ', maxsplit=1)  
 # Получить требуемый номер.  
 phone = parts[1]  
 # Отобразить выбранного человека  
 print(select\_student(students, phone))  
 elif command == 'help':  
 # Вывести справку о работе с программой.  
 print("Список команд:\n")  
 print("add - добавить человека;")  
 print("list - вывести список людей;")  
 print("phone <номер> - запросить человека по номеру;")  
 print("help - отобразить справку;")  
 print("exit - завершить работу с программой.")  
 else:  
 print(f"Неизвестная команда {command}", file=sys.stderr)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

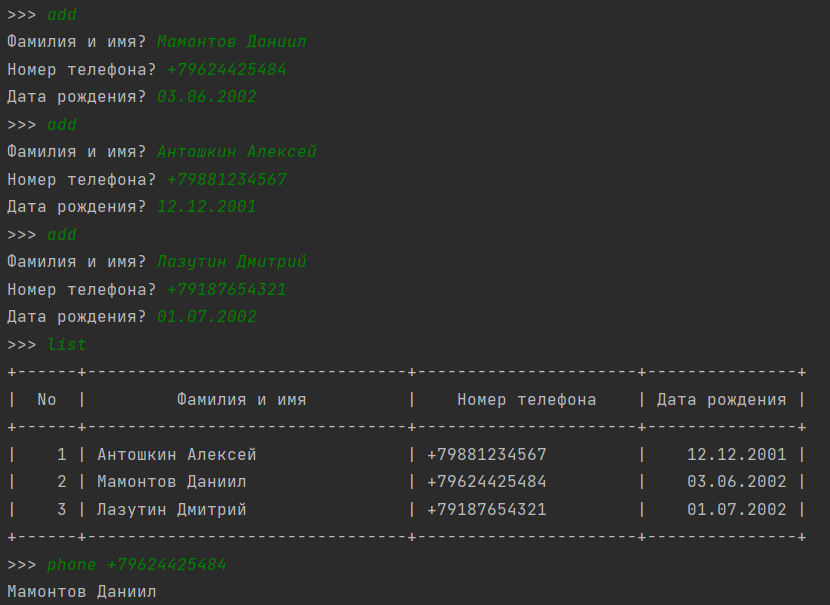


Рисунок 1 – результат работы программы

**КОНТРОЛЬНЫЕ 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.

1. 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.

1. 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.

1. Как вернуть несколько значений из функции Python?

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

1. Какие существуют способы передачи значений в функцию?

C 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.

1. Как задать значение аргументов функции по умолчанию?

def do\_smth(a, b=2) # Значение по умолчанию b = 2

1. 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, например.

1. 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.

1. 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.