

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**
**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

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

Выполнил студент группы
ПИЖ-б-о-21-1

Трушева В. О. .« » 2022г.

Подпись студента _____

Работа защищена «
» _____ 20__ г.

Проверила Воронкин Р.А.

(подпись)

Ставрополь 2022

Методика и порядок выполнения 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 соответствие с моделью ветвления `git-flow`.
 6. Создайте проект PyCharm в папке репозитория.
 7. Проработайте пример лабораторной работы. Создайте для него отдельный модуль языка Python. Зафиксируйте изменения в репозитории.
- Условие. Для примера 1 лабораторной работы 2.6, оформить каждую команду в виде вызова отдельной функции.

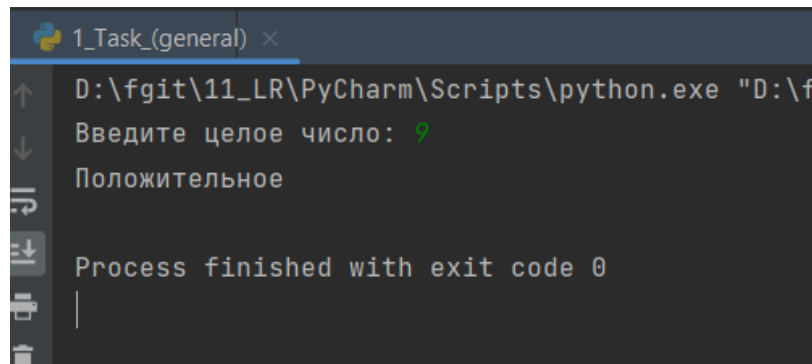
```
1_Task_(primer) x
D:\fgit\11_LR\PyCharm\Scripts\python.exe "D:\fgit\11_LR\Tasks\1_Task_(primer).py"
>>> help
Список команд:
add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - вывести список команд;
exit - завершить работу с программой.
>>> add
Фамилия и инициалы: Трушева В0
Должность: Фотограф
Год поступления: 2020
>>> list
+-----+-----+-----+-----+
| № |           Ф.И.О.           |      Должность      |   Год   |
+-----+-----+-----+-----+
|  1 | Трушева В0                 | Фотограф            |  2020   |
+-----+-----+-----+-----+
>>> select 1
+-----+-----+-----+-----+
| № |           Ф.И.О.           |      Должность      |   Год   |
+-----+-----+-----+-----+
|  1 | Трушева В0                 | Фотограф            |  2020   |
+-----+-----+-----+-----+
>>> exit

Process finished with exit code 0
```

Рисунок 1 – Результат выполнения программы

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()`

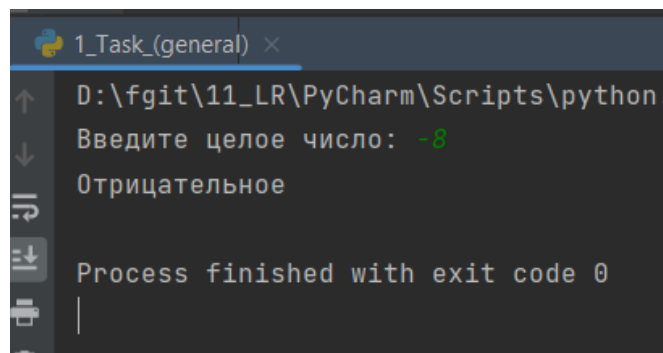
предшествовать test() или могут следовать после него? Проверьте вашу гипотезу, поменяв объявления функций местами. Попробуйте объяснить результат.



```
1_Task_(general) x
D:\fgit\11_LR\PyCharm\Scripts\python.exe "D:\f
Введите целое число: 9
Положительное

Process finished with exit code 0
```

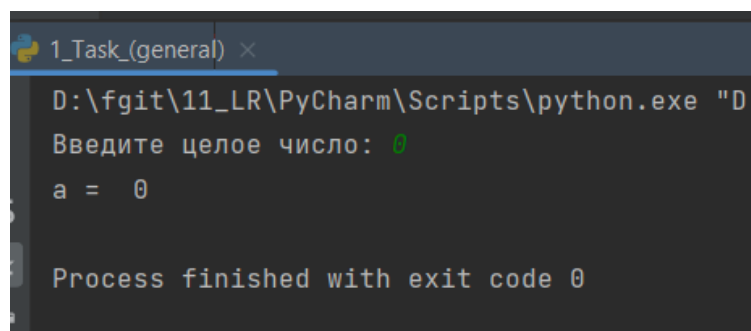
Рисунок 2 – Результат выполнения программы



```
1_Task_(general) x
D:\fgit\11_LR\PyCharm\Scripts\python.
Введите целое число: -8
Отрицательное

Process finished with exit code 0
```

Рисунок 3 – Результат выполнения программы



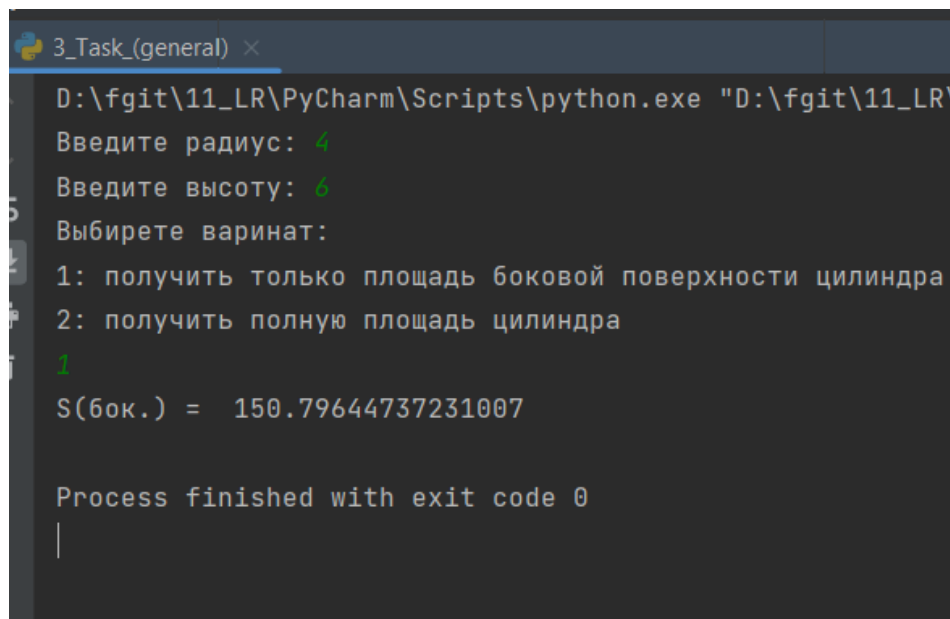
```
1_Task_(general) x
D:\fgit\11_LR\PyCharm\Scripts\python.exe "D:
Введите целое число: 0
a = 0

Process finished with exit code 0
```

Рисунок 4 – Результат выполнения программы

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



```
3_Task_(general) x
D:\fgit\11_LR\PyCharm\Scripts\python.exe "D:\fgit\11_LR\
Введите радиус: 4
Введите высоту: 6
Выберите вариант:
1: получить только площадь боковой поверхности цилиндра
2: получить полную площадь цилиндра
1
S(бок.) = 150.79644737231007

Process finished with exit code 0
|
```

Рисунок 5 – Результат выполнения программы

```
3_Task_(general) x
D:\fgit\11_LR\PyCharm\Scripts\python.exe "D:\fgit\11_LR\T
Введите радиус: 5
Введите высоту: 2
Выберите варинат:
1: получить только площадь боковой поверхности цилиндра
2: получить полную площадь цилиндра
2
S(полн.) = 219.9114857512855

Process finished with exit code 0
```

Рисунок 6 – Результат выполнения программы

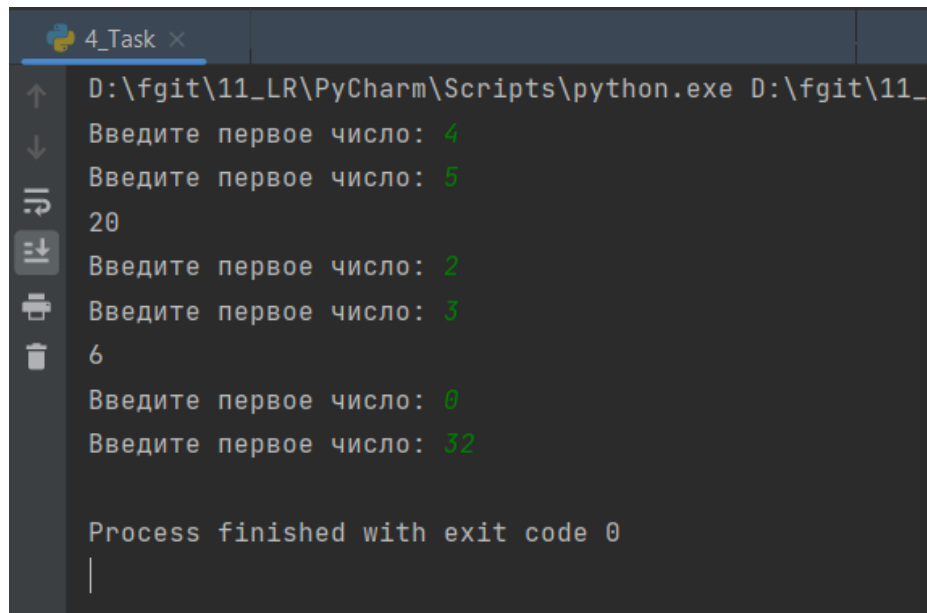
```
3_Task_(general) x
D:\fgit\11_LR\PyCharm\Scripts\python.exe "D:\fgit\11_LR\Ta
Введите радиус: 4
Введите высоту: 6
Выберите варинат:
1: получить только площадь боковой поверхности цилиндра
2: получить полную площадь цилиндра
4
Неизвестная команда 4

Process finished with exit code 0
```

Рисунок 7 – Результат выполнения программы

11. Зафиксируйте 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.



```
D:\fgit\11_LR\PyCharm\Scripts\python.exe D:\fgit\11_
Введите первое число: 4
Введите первое число: 5
20
Введите первое число: 2
Введите первое число: 3
6
Введите первое число: 0
Введите первое число: 32

Process finished with exit code 0
```

Рисунок 8 – Результат выполнения программы

13. Зафиксируйте 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.

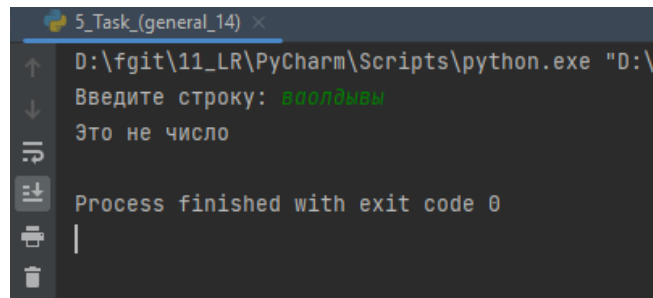
2. Функция `test_input()` имеет один параметр. В теле она проверяет, можно ли переданное ей значение преобразовать к целому числу. Если можно, возвращает логическое `True`. Если нельзя – `False`.

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 и ничего не возвращает.

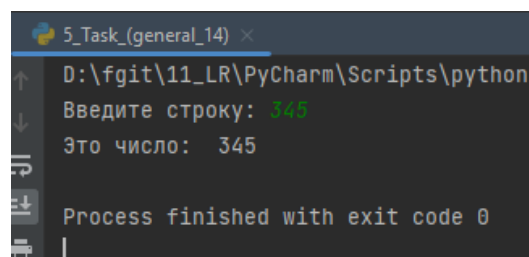
В основной ветке программы вызовите первую функцию. То, что она вернула, передайте во вторую функцию. Если вторая функция вернула

True, то те же данные (из первой функции) передайте в третью функцию, а возвращенное третьей функцией значение – в четвертую.



```
5_Task_(general_14) x
D:\fgit\11_LR\PyCharm\Scripts\python.exe "D:\f
Введите строку: вапдым
Это не число
Process finished with exit code 0
```

Рисунок 9 – Результат выполнения программы



```
5_Task_(general_14) x
D:\fgit\11_LR\PyCharm\Scripts\python.
Введите строку: 345
Это число: 345
Process finished with exit code 0
```

Рисунок 10 – Результат выполнения программы

Индивидуальное задание. Вариант – 11

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

Условие. Использовать словарь, содержащий следующие ключи: фамилия, имя; номер телефона; дата рождения (список из трех чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по датам рождения; вывод на экран информации о человеке, номер телефона которого введен с клавиатуры; если такого нет, выдать на дисплей соответствующее сообщение.


```
1_Task x
Введите команду >>> help
Список команд:
add - добавить человека;
list - вывести список людей;
find - найти человека по фамилии;
help - отобразить справку;
exit - завершить работу с программой.
Введите команду >>> add
Фамилия Имя: Вероника Трушева
Номер телефона: 123
Дата рождения: 12.12.2000
Введите команду >>> add
Фамилия Имя: Иван Иванов
Номер телефона: 2143
Дата рождения: 10.10.1998
Введите команду >>> find
Введите номер телефона: 123
+-----+-----+-----+-----+
| № | Фамилия Имя | Номер телефона | Дата рождения |
+-----+-----+-----+-----+
| 1 | Вероника Трушева | 123 | 2000-12-12 |
+-----+-----+-----+-----+
Введите команду >>> list
+-----+-----+-----+-----+
| № | Фамилия Имя | Номер телефона | Дата рождения |
+-----+-----+-----+-----+
| 1 | Вероника Трушева | 123 | 2000-12-12 |
| 2 | Иван Иванов | 2143 | 1998-10-10 |
+-----+-----+-----+-----+
Введите команду >>> exit
```

Рисунок 11 – Результат выполнения программы

15. Зафиксируйте изменения в репозитории.
16. Приведите в отчете скриншоты результатов выполнения примера при различных исходных данных вводимых с клавиатуры.
17. Приведите в отчете скриншоты работы программ решения индивидуального задания.
18. Зафиксируйте сделанные изменения в репозитории.

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 – 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 PEP 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.