

**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»**

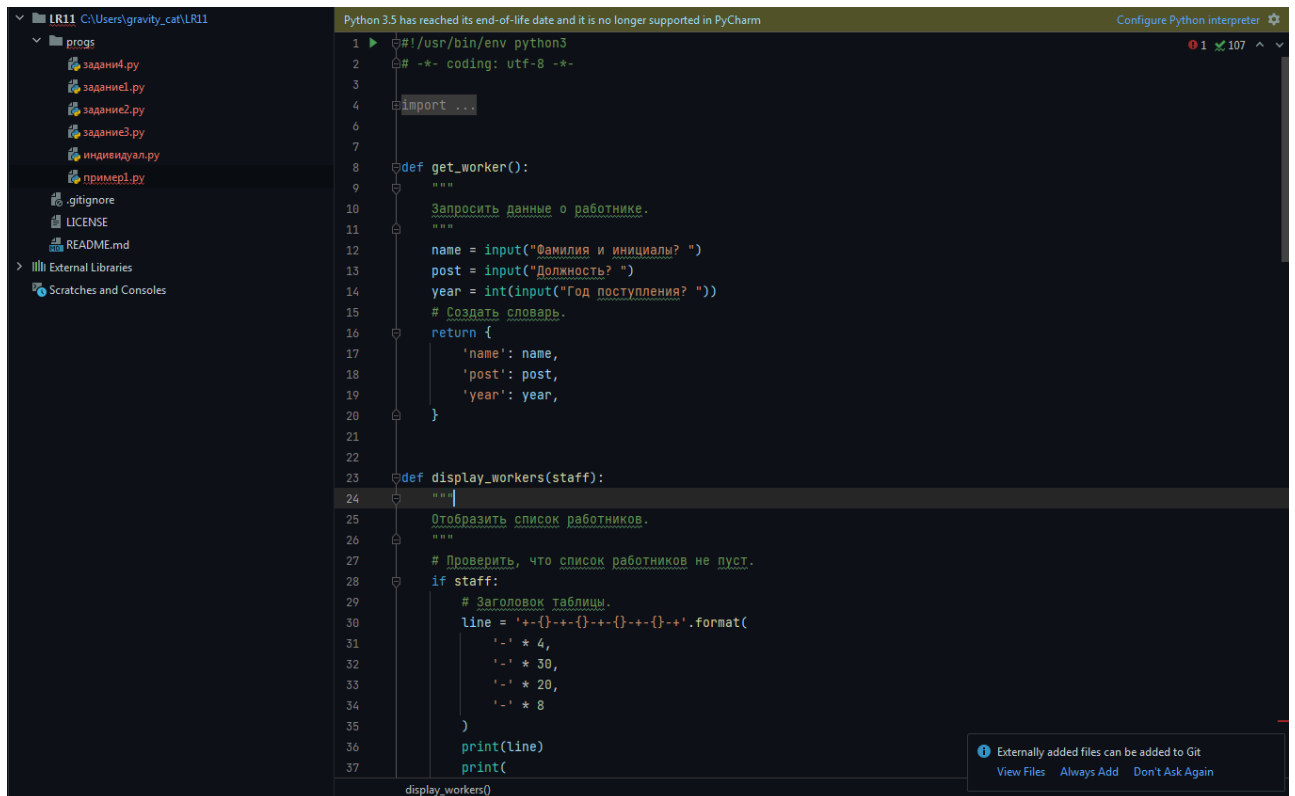
Выполнил:  
Гребен Владислав  
Александрович,  
2 курс, группа ПИЖ-б-о-20-1,

Проверил:  
Доцент кафедры  
прикладной математики и  
компьютерной безопасности,  
Воронкин Р.А.

Отчет защищен с оценкой\_\_\_\_\_Дата защиты\_\_\_\_\_

Ставрополь, 2021 г.

## ВЫПОЛНЕНИЕ:



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import ...
5
6
7
8 def get_worker():
9     """
10     Запросить данные о работнике.
11     """
12     name = input("Фамилия и инициалы? ")
13     post = input("Должность? ")
14     year = int(input("Год поступления? "))
15     # Создать словарь.
16     return {
17         'name': name,
18         'post': post,
19         'year': year,
20     }
21
22
23 def display_workers(staff):
24     """
25     Отобразить список работников.
26     """
27     # Проверить, что список работников не пуст.
28     if staff:
29         # Заголовок таблицы.
30         line = '+-{}-+-{}-+-{}-+-{}-+'.format(
31             '-' * 4,
32             '-' * 30,
33             '-' * 20,
34             '-' * 8
35         )
36         print(line)
37         print(line)
```

Рисунок 11.1 – код первого примера



```
C:\Users\gravity_cat\AppData\Local\Programs\Python\Python35\python.exe C:/Users/gravity_cat/LR11/progs/пример1.py
>>> 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? Shayrman O.V
Должность? Cook
Год поступления? 2001
>>> add
Фамилия и инициалы? Kireev V.e
Должность? ment
Год поступления? 2020
>>> add
Фамилия и инициалы? Tvoroshniy V.E
Должность? Driver
Год поступления? 2010
>>> list
+-----+
| No |      Ф.И.О.      |      Должность      |      Год      |
+-----+
| 1 | Kireev V.e        | ment                | 2020 |
+-----+
| 2 | Shayrman O.V      | Cook                | 2001 |
+-----+
| 3 | Tvoroshniy V.E    | Driver              | 2010 |
+-----+
>>> |
```

Рисунок 11.2 – результат первого примера

```
Python 3.5 has reached its end-of-life date and it is no longer supported in PyCharm
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 def test():
5     a = int(input("Enter the number: "))
6     if a > 0:
7         positive()
8     elif a < 0:
9         negative()
10
11
12 def positive():
13     print("This number is positive")
14
15
16 def negative():
17     print("This number is negative")
18
19
```

Run: **пример1** × **задание1** ×

C:\Users\gravity\_cat\AppData\Local\Programs\Python\Python35\python.exe C:/Users/gravity\_cat/LR11/progs/задание1.py

Enter the number: 20  
This number is positive

Рисунок 11.3 – Задание №1

```
Python 3.5 has reached its end-of-life date and it is no longer supported in PyCharm
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5
6
7 def cylinder():
8
9     def circle(r):
10         circle_square = math.pi * r ** 2
11         return circle_square
12
13     radius = float(input("Enter the radius of the cylinder: "))
14     height = float(input("Enter the height if the cylinder: "))
15     print("Do you want to compute the full cylinder square? - 'yes' or 'no'")
16     command = input().lower()
17     if command == 'yes':
18         print(2 * math.pi * radius * height + 2 * circle(radius))
19     if command == 'no':
20
```

Run: **пример1** × **задание2** ×

C:\Users\gravity\_cat\AppData\Local\Programs\Python\Python35\python.exe C:/Users/gravity\_cat/LR11/progs/задание2.py

Enter the radius of the cylinder: 20  
Enter the height if the cylinder: 40  
Do you want to compute the full cylinder square? - 'yes' or 'no'  
yes  
7539.822368615503

Рисунок 11.4 – Задание №2

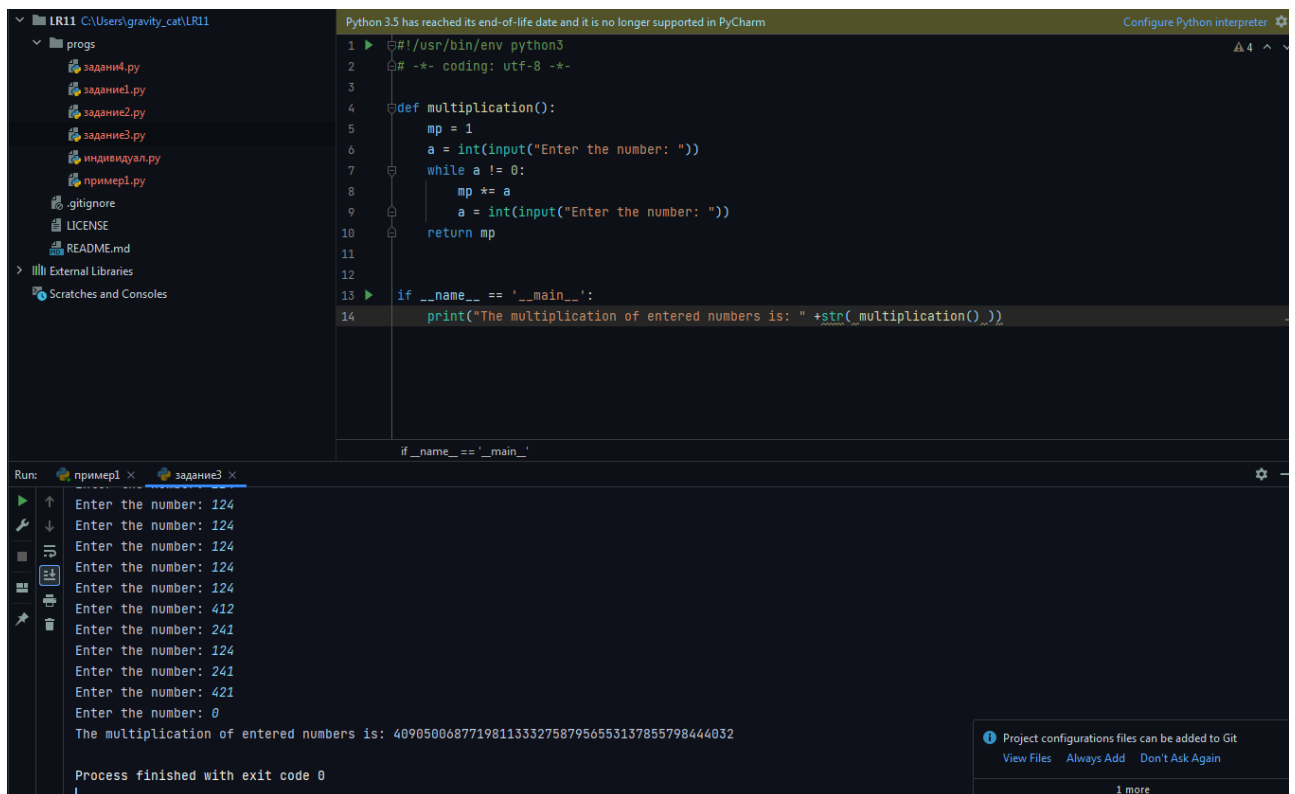


Рисунок 11.5 – Задание №3

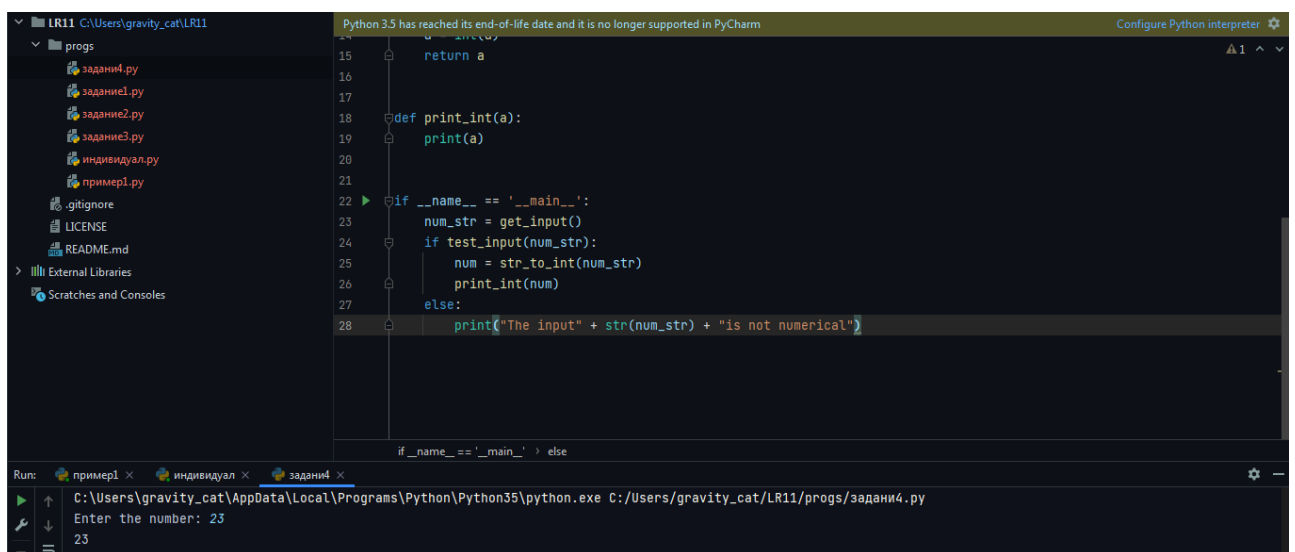


Рисунок 11.6 – Задание №4

## ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

### Вариант 5

```
7  def get_flight():
8      """
9      Запросить данные о полёте
10     """
11     flight_destination = input("Введите название пункта назначения ")
12     flight_number = input("Введите номер рейса ")
13     airplane_type = input("Введите тип самолета ")
14     return {
15         'flight_destination': flight_destination,
16         'flight_number': flight_number,
17         'airplane_type': airplane_type,
18     }
19
20
21 def display_flights(flights):
22     """
23     Отобразить список рейсов
24     """
25     if flights:
26         line = '+--{}--{}--{}--{}--+'.format(
27             '-' * 4,
28             '-' * 30,
29             '-' * 20,
30             '-' * 15
31         )
32         print(line)
33         print(
34             '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
35                 "No",
36                 "Пункт назначения",
37                 "Номер рейса",
38                 "Тип самолета"
39             )
40         )
41         print(line)
```

Рисунок 11.7 – Код индивидуального задания

```

C:\Users\gravity_cat\AppData\Local\Programs\Python\Python35\python.exe C:/Users/gravity_cat/LR11/progs/индивидуал.py
>>> help
Список команд:

add - добавить рейс;
list - вывести список всех рейсов;
select <тип самолета> - запросить рейсы указанного типа самолета;
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Введите название пункта назначения Москва
Введите номер рейса 435
Введите тип самолета FlexAir
>>> add
Введите название пункта назначения Уганда
Введите номер рейса 228
Введите тип самолета Люфтвафля
>>> add
Введите название пункта назначения Нью-Йорк
Введите номер рейса 911
Введите тип самолета
>>> select 911
Для типа самолета : 911
рейсы не найдены
Список рейсов пуст
>>> list
+-----+-----+-----+-----+
| No |      Пункт назначения      |      Номер рейса      |      Тип самолета      |
+-----+-----+-----+-----+
|  1 | Москва                    |      435              |      FlexAir           |
|  2 | Нью-Йорк                  |      911              |      Husein            |
|  3 | Уганда                    |      228              |      Люфтвафля          |
+-----+-----+-----+-----+
>>>

```

Рисунок 11.8 – Вывод ИДЗ

## ОТВЕТЫ НА ВОПРОСЫ

1. Функция в программировании представляет собой обособленный участок кода, который можно вызывать, обратившись к нему по имени, которым он был назван. При вызове происходит выполнение команд тела функции. Функции можно сравнить с небольшими программками, которые сами по себе, т. е. автономно, не исполняются, а встраиваются в обычную программу. Нередко их так и называют – подпрограммы. Других ключевых отличий функций от программ нет. Функции также при необходимости могут получать и возвращать данные. Только обычно они их получают не с ввода (клавиатуры, файла и др.), а из вызывающей программы. Сюда же они возвращают результат своей работы. Внедрение функций позволяет решить проблему дублирования кода в разных местах программы. Благодаря им можно исполнять один и тот же участок кода не сразу, а только тогда, когда он понадобится.
2. В языке программирования Python функции определяются с помощью оператора `def`. Ключевое слово `def` сообщает интерпретатору, что перед ним определение функции. За `def` следует имя функции. Оно может быть любым, так же, как и всякий идентификатор, например, переменная.
3. Функции могут передавать какие-либо данные из своих тел в основную ветку программы. Говорят, что функция возвращает значение. В большинстве языков программирования, в том числе Python, выход из функции и передача данных в то место, откуда она была вызвана, выполняется оператором `return`. Если интерпретатор Питона, выполняя тело функции, встречает `return`, то он "забирает" значение, указанное после этой команды, и "уходит" из функции.
4. 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. 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. При выходе из нее, локальные переменные исчезают. Компьютерная память, которая под них отводилась, освобождается. Когда функция будет снова вызвана, локальные переменные будут созданы заново.
5. В Питоне позволительно возвращать из функции несколько объектов, перечислив их через запятую после команды `return`.
6. В строке объявления функции указать в скобках значение параметра по умолчанию.
7. `lambda` функции позволяют 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. Есть и еще одно интересное применение - хранение списка обработчиков данных в списке/словаре.

8. Строки документации - строковые литералы, которые являются первым оператором в модуле, функции, классе или определении метода. Такая строка документации становится специальным атрибутом `__doc__` этого объекта. Все модули должны иметь строки документации, и все функции и классы, экспортируемые модулем также должны иметь строки документации. Публичные методы (в том числе `__init__` ) также должны иметь строки документации. Пакет модулей может быть документирован в `__init__.py`. Для согласованности, всегда используйте `"""triple double quotes"""` для строк документации. Используйте `r"""raw triple double quotes"""`, если вы будете использовать обратную косую черту в строке документации. Существует две формы строк документации: однострочная и многострочная.

9. Одиночные строки документации предназначены для действительно очевидных случаев. Они должны уместиться на одной строке. Используйте тройные кавычки, даже если документация уместается на одной строке. Потом будет проще её дополнить. Однострочная строка документации не должна быть "подписью" параметров функции / метода (которые могут быть получены с помощью 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. Вставляйте пустую строку до и после всех строк документации (однострочных или многострочных), которые документируют класс - вообще говоря, методы класса разделены друг от друга одной пустой строкой, а строка документации должна быть смещена от первого метода пустой строкой; для симметрии, поставьте пустую строку между заголовком класса и строкой документации. Строки документации функций и методов, как правило, не имеют этого требования