

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

**Кафедра  
инфокоммуникаций  
Институт цифрового  
развития**

**ОТЧЁТ  
по лабораторной работе №2.8  
Дисциплина: «Основы программной инженерии»  
Тема: «Работа с функциями в Python»**

Выполнила:  
студентка 2 курса  
группы Пиж-б-о-21-1  
Джолдошова Мээрим  
Бекболотовна

Ставрополь 2022

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

1. Был создан репозиторий в Github в который были добавлены правила gitignore для работы IDE PyCharm, была выбрана лицензия MIT, сам репозиторий был клонирован на локальный сервер и был организован в соответствии с моделью ветвления git-flow.

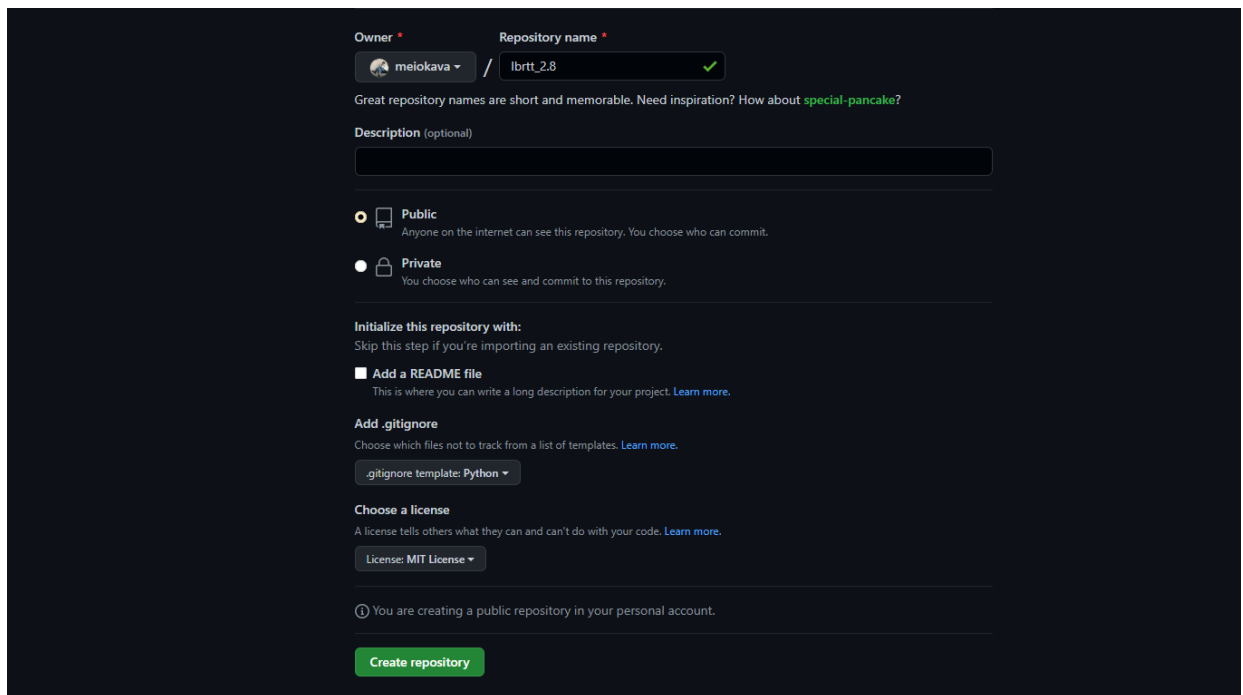


Рисунок 1 – Создание репозитория

```
c:\Users\мвидео>cd/d c:\lbrtt_2.8
c:\lbrtt_2.8>git clone https://github.com/meiokava/lbrtt_2.8.git
Cloning into 'lbrtt_2.8'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
c:\lbrtt_2.8>cd/d c:\lbrtt_2.8\lbrtt_2.8
c:\lbrtt_2.8\lbrtt_2.8>git flow init_
```

Рисунок 2 – Клонирование репозитория

```

c:\lbrtt_2.8\lbrtt_2.8>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:/lbrtt_2.8/lbrtt_2.8/.git/hooks]
c:\lbrtt_2.8\lbrtt_2.8>_

```

Рисунок 3 – Организация репозитория в соответствии с моделью ветвления git-flow

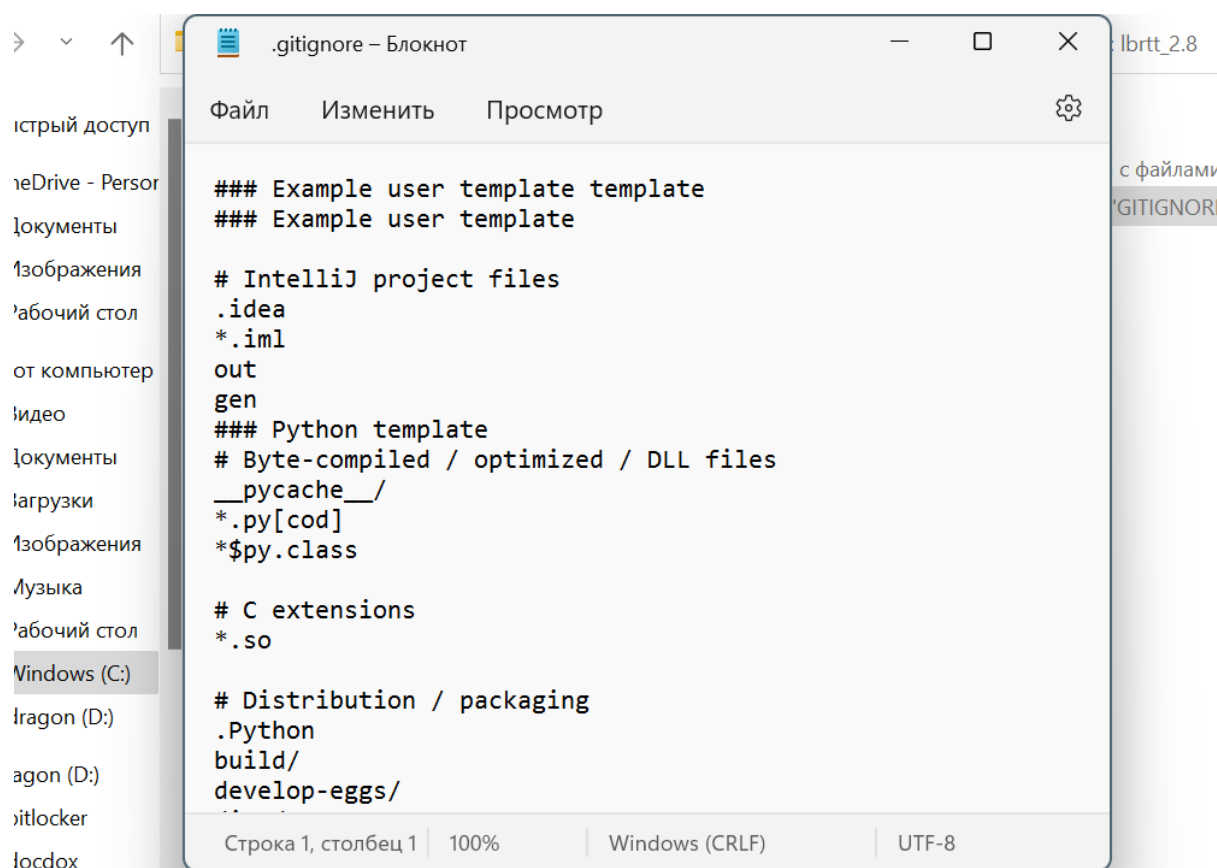


Рисунок 4 – Изменение gitignore

2. Была создана папка PyCharm в которой хранятся примеры из лабораторной работы.

Имя	Дата изменения	Тип
.git	03.12.2022 10:36	Папка с файлами
PyCharm	03.12.2022 10:45	Папка с файлами
.gitignore	02.12.2022 23:43	Файл "GITIGNORE"
LICENSE	03.12.2022 10:34	Файл

Рисунок 5 – Папка для хранения примеров

```

C:\Users\mvideo\PycharmProjects\pythonProject53\venv\Scripts\python.exe C:\Users\mvideo\PycharmProjects\pythonProject53\main.py
>>> add
Фамилия и инициалы: gosh n
Должность: boss
Год поступления: 2003
>>> list
+-----+-----+-----+-----+
| № |      Ф.И.О.      |      Должность      |      Год      |
+-----+-----+-----+-----+
| 1 | gosh n          | boss                | 2003          |
+-----+-----+-----+-----+
>>>

```

Рисунок 6 – Результат работы программы

## Задача 1

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.

Код программы:

```

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

```

```
import sys

def test():
    """
    inputing a digit and checking the meaning
    """
    dig = int(input('enter positive or negative digit: '))
    if dig >= 0:
        positive()
    else:
        negative()

def positive():
    """
    output to the screen
    """
    print('positive')

def negative():
    """
    output to the screen
    """
    print('negative')

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

```
C:\Users\мвидео\PycharmProjects\pythonProject54\venv\Scripts\python.exe
enter positive or negative digit: 6
positive

Process finished with exit code 0
```

Рисунок 7 – Результат работы программы при положительном числе

```
main x
C:\Users\мвидео\PycharmProjects\pythonProject54\venv\Scripts\python.exe
enter positive or negative digit: -1
negative

Process finished with exit code 0
```

Рисунок 8 – Результат работы программы при отрицательном числе

## Задача 2

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 программы:

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

import math

def cylinder():
    """
    calculation of the full or side square cylinder
    """
    while True:
        option = int(input("<<<What kind of square do you want to calculate?>>>\n"))
        if option == 1:
            print("1 - Side surface\n")
        elif option == 2:
            print("2 - Full surface\n")
        else:
            print(">>> ")
        if (option != 1) and (option != 2):
            print('unknown command')
            break
        r = int(input("Enter the radius: "))
        h = int(input("Enter the height of the cylinder: "))
        if option == 1:
            s = 2 * math.pi * r * h
            print("S(side.) = ", '{:.3f}'.format(s))
            break
        else:
            s = 2 * math.pi * r * h + 2 * circle(r)
            print("S(full) = ", '{:.3f}'.format(s))
            break
```

```

        else:
            s = (2 * math.pi * r * h) + (circle(r) * 2)
            print("S(full.) = ", '{:.3f}'.format(s))
            break

def circle(r):
    """
    Calculation of the square of the circle by a given radius
    """
    return math.pi * (r ** 2)

if __name__ == '__main__':
    cylinder()

```

```

C:\Users\мвидео\PycharmProjects\pythonProject55\venv\Scripts\python.exe C:\
<<<What kind of square do you want to calculate?>>>
1 - Side surface
2 - Full surface
>>> 1
Enter the radius: 3
Enter the height of the cylinder: 5
S(side.) = 94.248

Process finished with exit code 0

```

Рисунок 9 – Результат работы программы

### Задача 3:

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 программы:

```

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

def multiply():
    """
    multiplying digits while it is not 0
    """
    print("<<<Enter digits that will be multiplied>>>")
    res = 1
    while True:
        a = int(input(">>> "))

```

```

        if a == 0:
            break
        else:
            res *= a
    return res

if __name__ == '__main__':
    print('result is', multiply())

```

```

Run: main x
C:\Users\мвидео\PycharmProjects\pythonProject56\venv\Scripts\py
<<<Enter digits that will be multiplied>>>
>>> 1
>>> 3
>>> 5
>>> 0
result is 15

Process finished with exit code 0

```

Рисунок 10 – Результат работы программы

#### Задание 4

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 программы

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

import sys

def get_input():
    """
    requests keyboard input and returns the given string
    """
    get_str = input("enter digit: ")
    return get_str

def test_input(a):
    """
    checks if the given string can be transformed
    into a number. If yes, it returns yes.
    """
    if type(a) == int or type(a) == float:
        return True
    elif a.isnumeric():
        return True
    else:
        return False

def str_to_int(b):
    """
    converts transmitted meaning into int type
    """
    c = int(b)
    return c

def print_int(c):
    """
    Displays the transmitted value on the screen
    """
    print(c)

if __name__ == '__main__':
    s = get_input()
    bol = test_input(s)
    if bol:
        nmb = str_to_int(s)
        print_int(nmb)
    else:
        print(f"the entered value is not a number!", file=sys.stderr)
```

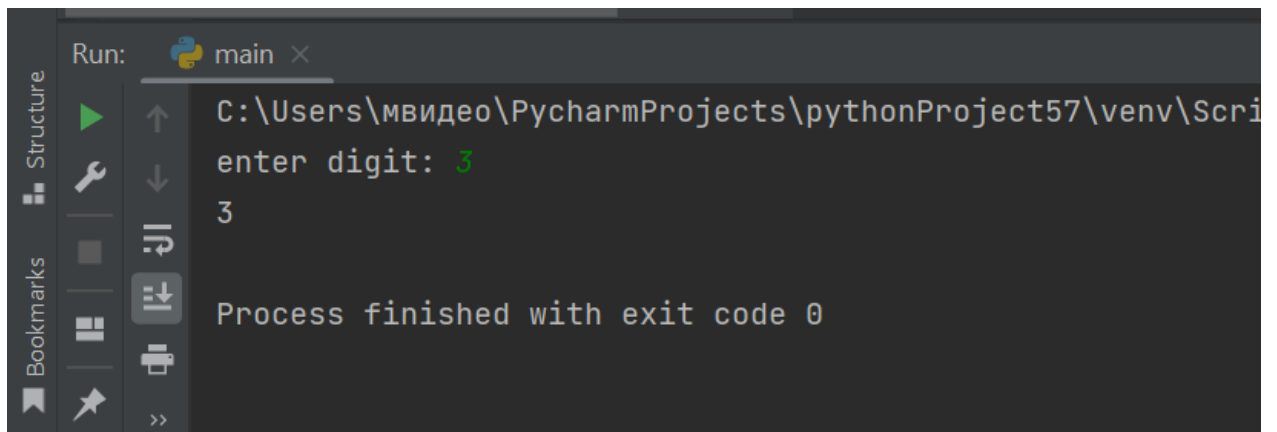


Рисунок 11 – Результат работы программы

### Индивидуальное задание

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

Код программы:

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

import sys

def get_flight():
    """
    requesting information about the flight
    """
    dst = input("What destination do you need? ")
    nmb = int(input("Which number of the flight do you need? "))
    tpe = input("Which type of plane do you need? ")

    #creation of dictionary
    return {
        'destination': dst,
        'number_flight': nmb,
        'type_plane': tpe,
    }

def display_flights(flights):
    """
    displaying the given information
    """
    if flights:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 18
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^18} |'.format(
                "№",

```

```

        "Destination",
        "Number of the flight",
        "Type of the plane"
    )
)
print(line)
for idx, flight in enumerate(flights, 1):
    print(
        '| {:>4} | {:<30} | {:<20} | {:>18} |'.format(
            idx,
            flight.get('destination', ''),
            flight.get('number_flight', ''),
            flight.get('type_plane', 0)
        )
    )
print(line)
else:
    print('list is empty')

def select_flights(flights):
    """
    selecting flights that are appropriate
    """
    t = input("choose type of the plane: ")
    count = 0
    for flight in flights:
        if flight.get('type_plane') == t:
            count += 1
            print(
                '{:>4}: {} {}'.format(
                    count,
                    flight.get('destination', ''),
                    flight.get("number_flight")
                )
            )
    if count == 0:
        print("We couldn't find this type of plane")

def main():
    """
    main function of the program
    """
    #list of the flights
    flights = []

    #organization of an endless loop
    while True:
        command = input(">>> ").lower()
        if command == 'exit':
            break
        elif command == 'add':
            flight = get_flight()
            flights.append(flight)
            if len(flights) > 1:
                flights.sort(key=lambda item: item.get('destination', ''))
        elif command == 'list':
            display_flights(flights)
        elif command.startswith('select'):
            select_flights(flights)
        elif command == 'help':
            print("command list:\n")
            print("add - add information about a flight;")

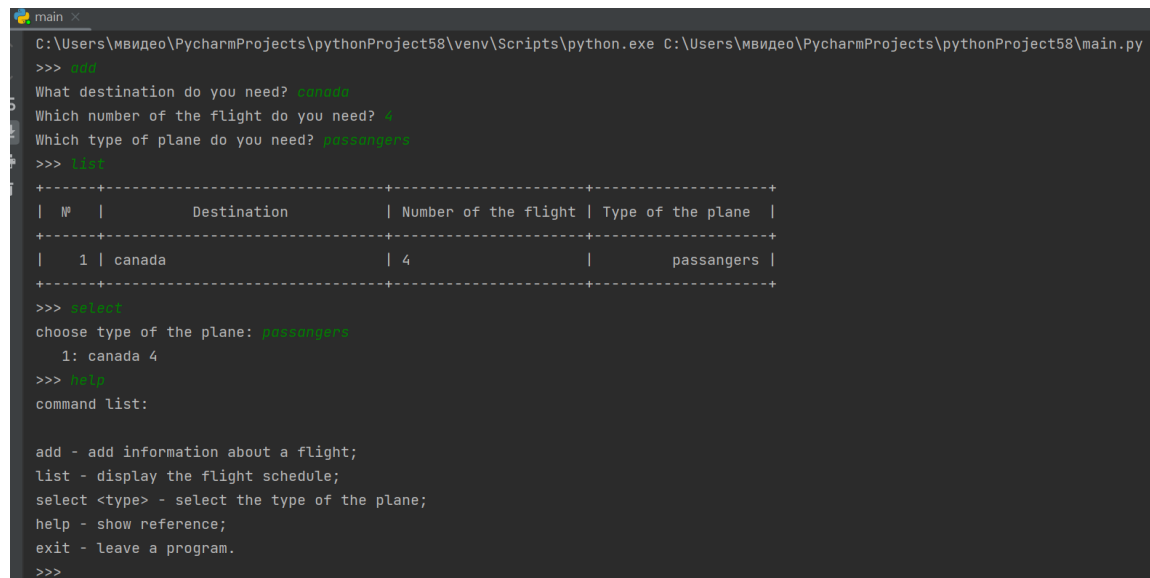
```

```

        print("list - display the flight schedule;")
        print("select <type> - select the type of the plane;")
        print("help - show reference;")
        print("exit - leave a program.")
    else:
        print(f"unknown command {command}", file=sys.stderr)

if __name__ == '__main__':
    main()

```



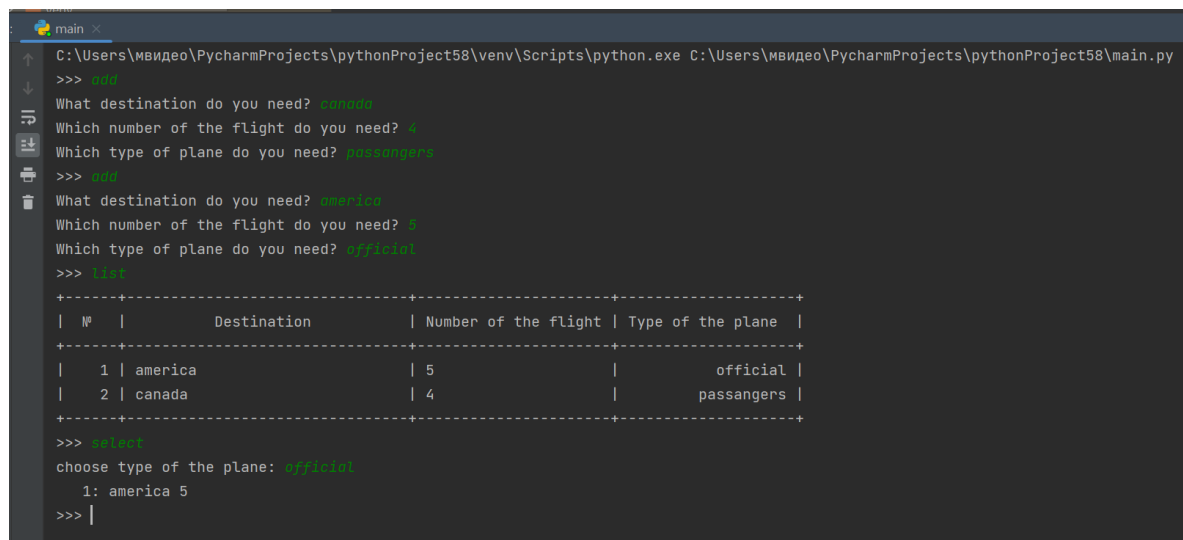
```

C:\Users\мвидео\PycharmProjects\pythonProject58\venv\Scripts\python.exe C:\Users\мвидео\PycharmProjects\pythonProject58\main.py
>>> add
What destination do you need? canada
Which number of the flight do you need? 4
Which type of plane do you need? passangers
>>> list
+-----+-----+-----+-----+
| № | Destination | Number of the flight | Type of the plane |
+-----+-----+-----+-----+
| 1 | canada | 4 | passangers |
+-----+-----+-----+-----+
>>> select
choose type of the plane: passangers
1: canada 4
>>> help
command list:

add - add information about a flight;
list - display the flight schedule;
select <type> - select the type of the plane;
help - show reference;
exit - leave a program.
>>>

```

Рисунок 12 – Результат работы программы



```

C:\Users\мвидео\PycharmProjects\pythonProject58\venv\Scripts\python.exe C:\Users\мвидео\PycharmProjects\pythonProject58\main.py
>>> add
What destination do you need? america
Which number of the flight do you need? 5
Which type of plane do you need? official
>>> add
What destination do you need? canada
Which number of the flight do you need? 4
Which type of plane do you need? passangers
>>> list
+-----+-----+-----+-----+
| № | Destination | Number of the flight | Type of the plane |
+-----+-----+-----+-----+
| 1 | america | 5 | official |
| 2 | canada | 4 | passangers |
+-----+-----+-----+-----+
>>> select
choose type of the plane: official
1: america 5
>>>

```

Рисунок 13 – Результат работы программы с двумя рейсами

```

c:\lbrtt_2.8\lbrtt_2.8>git add .
c:\lbrtt_2.8\lbrtt_2.8>git commit -m "completed tasks"
[develop 26c6fc5] completed tasks
7 files changed, 401 insertions(+), 2 deletions(-)
create mode 100644 Individ/idniv1.py
create mode 100644 PyCharm/examp1.py
create mode 100644 Tasks/task1.py
create mode 100644 Tasks/task2.py
create mode 100644 Tasks/task3.py
create mode 100644 Tasks/task4.py
c:\lbrtt_2.8\lbrtt_2.8>git push
fatal: The current branch develop has no upstream branch.

```

Рисунок 14 – Коммит и пуш изменений

```

c:\lbrtt_2.8\lbrtt_2.8>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
c:\lbrtt_2.8\lbrtt_2.8>git merge develop
Updating 6634198..26c6fc5
Fast-forward
 .gitignore      | 24 ++++++
 Individ/idniv1.py | 110 ++++++
 PyCharm/examp1.py | 123 ++++++
 Tasks/task1.py   | 36 ++++++
 Tasks/task2.py   | 39 ++++++
 Tasks/task3.py   | 21 ++++++
 Tasks/task4.py   | 50 ++++++
 7 files changed, 401 insertions(+), 2 deletions(-)
 create mode 100644 Individ/idniv1.py
 create mode 100644 PyCharm/examp1.py
 create mode 100644 Tasks/task1.py
 create mode 100644 Tasks/task2.py
 create mode 100644 Tasks/task3.py
 create mode 100644 Tasks/task4.py
c:\lbrtt_2.8\lbrtt_2.8>

```

Рисунок 15 – Слияние веток main и develop

```

c:\lbrtt_2.8\lbrtt_2.8>git push
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (12/12), 4.39 KiB | 1.46 MiB/s, done.
Total 12 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/meiokava/lbrtt_2.8.git
 6634198..26c6fc5  main -> main
c:\lbrtt_2.8\lbrtt_2.8>

```

Рисунок 16 – Пуш изменений

Вывод: в результате лабораторной работы были приобретены навыки по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.

## Контрольные вопросы и ответы на них:

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

Главной задачей функций в Python, как и в других языках программирования, является сокращение объёма кода и его структуризация. В функции, как правило, выносятся те части кода, которые выполняются в программе многократно.

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

Оператор def необходим для определения функции. После него идёт название самой функции, передаваемые в функцию параметры и само тело функции. Оператор return служит для возвращения результата выполнения функции в основную программу, где эта функция была вызвана.

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

Локальные переменные существуют только внутри функции. В другой части программы как-либо вызывать или изменить их невозможно.

Глобальные напротив – существуют во всей программе.

### 4. Как вернуть несколько значений из функции Python?

После оператора return необходимо записать все возвращаемые переменные через запятую, а при вызове функции нужно задать необходимое количество переменных. Куда будут возвращены параметры.

### 5. Какие существуют способы передачи значений в функцию?

По ссылке и по значению.

### 6. Как задать значение аргументов функции по умолчанию?

Нужно в скобках передаваемых параметров присвоить им значение.

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

Lambda-выражения – это небольшие функции, которые вызываются в программе один раз.

### 8. Как осуществляется документирование кода согласно PEP257?

Если пояснение функции содержит одну строку, то достаточно двух кавычек с каждой стороны строки. Пример: """ Пояснение """.

многострочное пояснение, то необходимо три кавычки с каждой стороны.

Пояснение находится в теле функции, сразу после её объявления