

**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-22-1

Душин Александр Владимирович.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

Тема: Работа с функциями в языке Python.

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

Ход выполнения работы:


1. Создать общедоступный репозиторий на GitHub с использованием лицензии MIT и язык программирования Python:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *

 MrPlatynum ▾

Repository name *

/ ProgrammEngineering11

✓ ProgrammEngineering11 is available.

Great repository names are short and memorable. Need inspiration? How about [special-waddle](#) ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)



You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание общедоступного репозитория на GitHub с заданными настройками

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents
$ git clone https://github.com/MrPlatynum/ProgrammEngineering11.git
Cloning into 'ProgrammEngineering11'...
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.
```

Рисунок 2 – Клонирование созданного репозитория на локальный компьютер

```
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
# JetBrains specific template is maintained in a separate JetBrains.gitignore that can
# be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
# and can be added to the global gitignore or merged into this file. For a more nuclear
# option (not recommended) you can uncomment the following to ignore the entire idea folder.
.idea/
```

Рисунок 3 – файл .gitignore

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering11 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering11 (develop)
$
```

Рисунок 4 – организация репозитория в соответствии с моделью ветвления git flow

2. Проработать примеры лабораторной работы, оформляя код согласно PEP-8:

Листинг примера:

```
#!/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
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "№",
                "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 = []
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)
            # 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()

```

```

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/3 семестр/Основы программной инженерии/ЛР11_ДушинАВ/example1.py"
>>> add
Фамилия и инициалы? Иванов И.И.
Должность? Главный
Год поступления? 2006
>>> add
Фамилия и инициалы? Петров П.П.
Должность? Секретарь
Год поступления? 2007
>>> list
+-----+-----+-----+-----+
| № |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Иванов И.И.              |      Главный        |      2006      |
|  2 | Петров П.П.              |      Секретарь      |      2007      |
+-----+-----+-----+-----+
>>> select 15
+-----+-----+-----+-----+
| № |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Иванов И.И.              |      Главный        |      2006      |
|  2 | Петров П.П.              |      Секретарь      |      2007      |
+-----+-----+-----+-----+
>>> exit

```

Рисунок 5 – Вывод программы

3. 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".

```

task1.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      def test():
5          num = int(input("Введите целое число: "))
6          if num > 0:
7              positive()
8          elif num < 0:
9              negative()
10
11
12      def positive():
13          print("Положительное")
14
15
16      def negative():
17          print("Отрицательное")
18
19
20  ▶  if __name__ == '__main__':
21      test()
22

```

Рисунок 6 – Задание №1

```

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/3 семестр/Основы программной инженерии/ЛР11_ДушинАВ/task1.py"
Введите целое число: -5
Отрицательное

```

Рисунок 7 – Вывод программы

4. 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. В теле `cylinder()` у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле, или полную площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции `circle()`.

```

1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      import math
5
6
7      MrPlatinum
8      def circle(radius):
9          """
10             Вычисляет площадь круга по заданному радиусу.
11             """
12             return math.pi * radius ** 2
13
14      MrPlatinum *
15      def cylinder():
16          """
17             Вычисляет площадь боковой поверхности цилиндра или полную площадь цилиндра
18             в зависимости от выбора пользователя.
19             """
20             radius = float(input("Введите радиус цилиндра: "))
21             height = float(input("Введите высоту цилиндра: "))
22
23             side_area = 2 * math.pi * radius * height
24             full_area = side_area + 2 * circle(radius)
25
26             choice = input("Хотите получить только площадь боковой поверхности? (yes/no): ").lower()
27
28             match choice:
29                 case 'yes':
30                     print(f"Площадь боковой поверхности цилиндра: {side_area:.2f}")
31                 case 'no':
32                     print(f"Полная площадь цилиндра: {full_area:.2f}")
33                 case _:
34                     print("Некорректный ввод. Пожалуйста, введите 'yes' или 'no'.")
35
36      MrPlatinum
37      def main():
38          """
39             Главная функция программы.
40             """
41             cylinder()
42
43      if __name__ == '__main__':
44          main()
45

```

Рисунок 9 – Задание №2

```

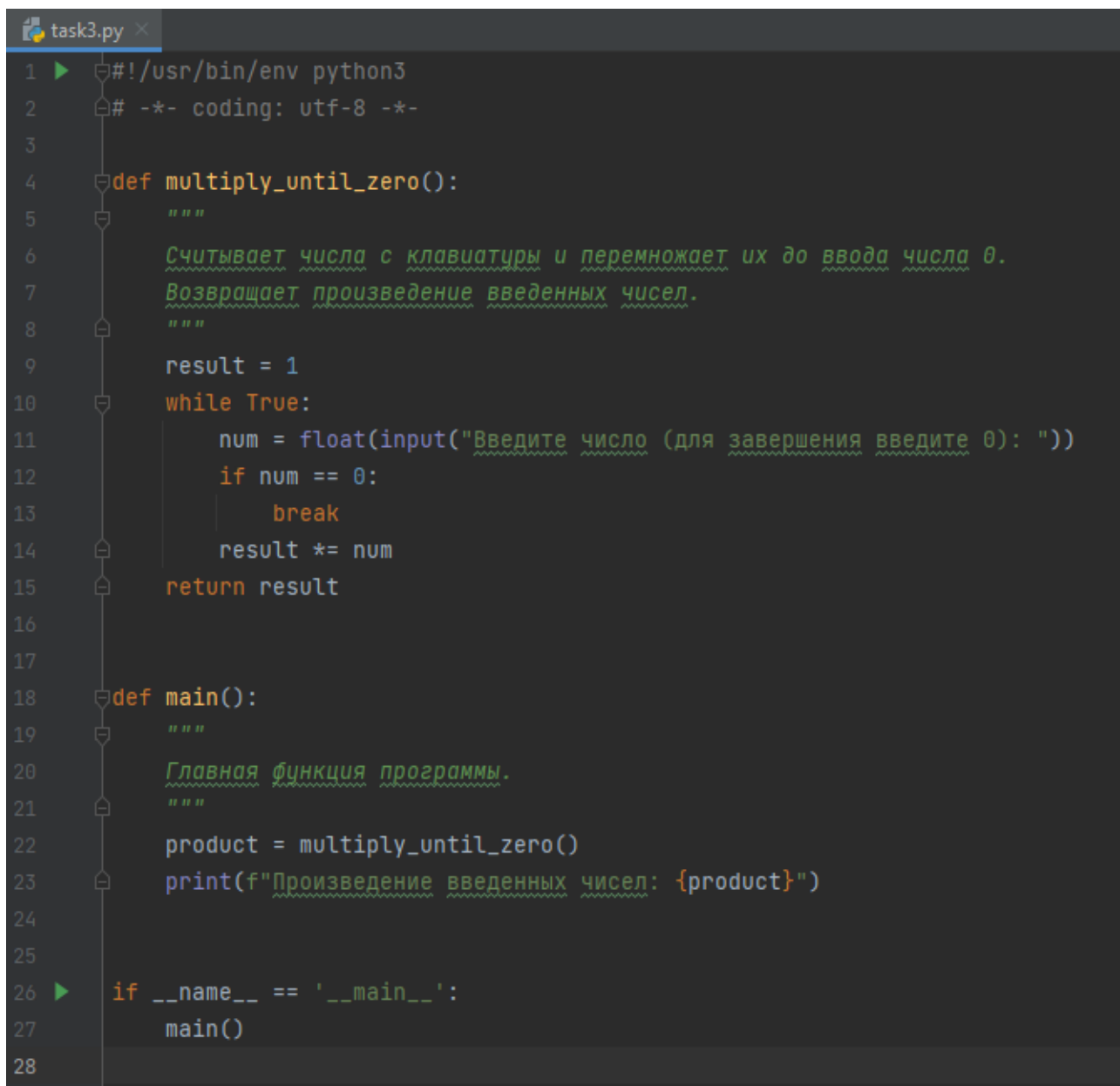
"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/3 семестр/Основы программной инженерии/ЛР11_ДушинAB/task2.py"
Введите радиус цилиндра: 7
Введите высоту цилиндра: 4
Хотите получить только площадь боковой поверхности? (yes/no): no
Полная площадь цилиндра: 282.74

```

Рисунок 10 – Вывод программы

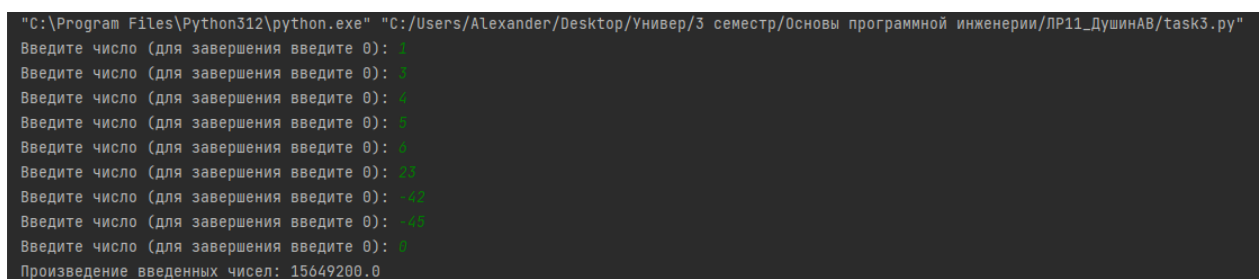
5. Решите следующую задачу: напишите функцию, которая считывает с клавиатуры числа и перемножает их до тех пор, пока не будет введен 0. Функция

должна возвращать полученное произведение. Вызовите функцию и выведите на экран результат ее работы.



```
task3.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def multiply_until_zero():
5      """
6      Считывает числа с клавиатуры и перемножает их до ввода числа 0.
7      Возвращает произведение введенных чисел.
8      """
9      result = 1
10     while True:
11         num = float(input("Введите число (для завершения введите 0): "))
12         if num == 0:
13             break
14         result *= num
15     return result
16
17
18 def main():
19     """
20     Главная функция программы.
21     """
22     product = multiply_until_zero()
23     print(f"Произведение введенных чисел: {product}")
24
25
26 if __name__ == '__main__':
27     main()
28
```

Рисунок 11 – Задание №3



```
"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/3 семестр/Основы программной инженерии/ЛР11_ДушинАВ/task3.py"
Введите число (для завершения введите 0): 1
Введите число (для завершения введите 0): 3
Введите число (для завершения введите 0): 4
Введите число (для завершения введите 0): 5
Введите число (для завершения введите 0): 6
Введите число (для завершения введите 0): 23
Введите число (для завершения введите 0): -42
Введите число (для завершения введите 0): -45
Введите число (для завершения введите 0): 0
Произведение введенных чисел: 15649200.0
```

Рисунок 12 – Вывод программы

6. 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.

Листинг программы:

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

def get_input():
    """
    Запрашивает ввод с клавиатуры и возвращает введенную строку.
    """
    user_input = input("Введите значение: ")
    return user_input

def test_input(value):
    """
    Проверяет, можно ли преобразовать значение к целому числу.
    Возвращает True, если можно, иначе - False.
    """
    try:
        int(value)
        return True
    except ValueError:
        return False

def str_to_int(value):
    """
    Преобразует переданное значение к целому числу.
    Возвращает полученное число.
    """
    return int(value)
```

```

def print_int(value):
    """
    Выводит переданное значение на экран.
    """
    print(value)

def main():
    """
    Главная функция программы.
    """
    # Получаем ввод пользователя
    user_value = get_input()

    # Проверяем, можно ли преобразовать введенное значение к целому числу
    if test_input(user_value):
        # Если можно, преобразуем строку в целое число
        int_value = str_to_int(user_value)
        # Выводим полученное целое число на экран
        print_int(int_value)
    else:
        print("Невозможно преобразовать введенное значение в целое число.")

if __name__ == '__main__':
    main()

```

```

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/3 семестр/Основы программной инженерии/ЛР11_ДушинаВ/task4.py"
Введите значение: 234234
234234

```

Рисунок 13 – Вывод программ с числом

```

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/3 семестр/Основы программной инженерии/ЛР11_ДушинаВ/task4.py"
Введите значение: string
Невозможно преобразовать введенное значение в целое число.

```

Рисунок 14 – Вывод программ со строкой

7. Выполним индивидуальные задания:

Листинг программы:

```

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

def add_train(trains):
    """Добавляет информацию о поезде."""
    destination = input("Название пункта назначения: ")
    train_number = input("Номер поезда: ")
    departure_time = input("Время отправления (в формате ЧЧ:ММ): ")

    train = {
        'название пункта назначения': destination,
        'номер поезда': train_number,
        'время отправления': departure_time,
    }

    trains.append(train)

```

```

trains.sort(key=lambda x: x['название пункта назначения'])

def list_trains(trains):
    """Выводит список всех поездов."""
    line = f'+-{"-" * 35}-+-{"-" * 15}-+-{"-" * 25}-+'
    print(line)
    print(f"| {'Название пункта назначения':^35} | {'Номер поезда':^15} | {'Время  
отправления':^25} | ")

    for train in trains:
        print(line)
        print(
            f"| {train['название пункта назначения':^35} | {train['номер поезда':^15} |  
{train['время отправления':^25} | ")
        print(line)

def select_trains(trains, search_time):
    """Выводит поезда, отправляющиеся после указанного времени."""
    found = False
    result = []

    print(f"Поезда, отправляющиеся после {search_time}:")
    for train in trains:
        train_time = train['время отправления']
        if train_time >= search_time:
            result.append(train)
            found = True
    if found:
        return result

    if not found:
        return "Нет поездов, отправляющихся после указанного времени."

def display_help():
    """Выводит справку о доступных командах."""
    print("Список команд:\n")
    print("add - добавить информацию о поезде;")
    print("list - вывести список всех поездов;")
    print("select <время> - вывести поезда, отправляющиеся после указанного  
времени;")
    print("exit - завершить работу с программой.")

def main():
    """Основная функция управления программой."""
    trains = []

    while True:
        command = input(">>> ").lower()

```

```

match command:
    case 'exit':
        break
    case 'add':
        add_train(trains)
    case 'list':
        list_trains(trains)
    case 'select':
        selected = select_trains(trains, input("Введите время для поиска поездов (в
формате ЧЧ:ММ): "))
        list_trains(selected)
    case 'help':
        display_help()
    case _:
        print(f"Неизвестная команда {command}")

if __name__ == '__main__':
    main()

```

```

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/3 семестр/Основы программной инженерии/ЛР9_ДушинаВ/individual1.py"
>>> add
Название пункта назначения: Москва
Номер поезда: 23
Время отправления (в формате ЧЧ:ММ): 14:00
>>> fvv
Неизвестная команда fvv
>>> add
Название пункта назначения: Омск
Номер поезда: 34
Время отправления (в формате ЧЧ:ММ): 12:30
>>> add Stavropol
Неизвестная команда add stavropol
>>> add
Название пункта назначения: Stavropol
Номер поезда: 24
Время отправления (в формате ЧЧ:ММ): 12:50
>>> list
+-----+-----+-----+
| Название пункта назначения | Номер поезда | Время отправления |
+-----+-----+-----+
| Омск | 34 | 12:30 |
+-----+-----+-----+
| Stavropol | 24 | 12:50 |
+-----+-----+-----+
| Москва | 23 | 14:00 |
+-----+-----+-----+

```

Рисунок 15 – Вывод программы

8. Зафиксируем сделанные изменения, сольем ветки и отправим на удаленный репозиторий:

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering11 (develop)
$ git log --oneline
c16828b (HEAD -> develop) add individual1.py
c77fa5d add task4.py
f3fe159 add task3.py
0f00ca5 add task2.py
e2b5d0f add task1.py
b8375db add example1.py
ac86591 (origin/main, origin/HEAD, main) Initial commit
```

Рисунок 16 – Коммиты ветки develop во время выполнения лабораторной работы

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering11 (develop)
$ git checkout main
Switched to branch 'main'
M       .gitignore
Your branch is up to date with 'origin/main'.

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering11 (main)
$ git merge develop
Updating ac86591..c16828b
Fast-forward
 example1.py | 126 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 individual1.py | 76 +++++++++++++++++++++++++++++++++++++
 task1.py      | 21 +++++
 task2.py      | 42 +++++
 task3.py      | 27 +++++
 task4.py      | 57 +++++
 6 files changed, 349 insertions(+)
 create mode 100644 example1.py
 create mode 100644 individual1.py
 create mode 100644 task1.py
 create mode 100644 task2.py
 create mode 100644 task3.py
 create mode 100644 task4.py

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering11 (main)
$ |
```

Рисунок 17 – Слияние ветки develop в ветку main

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering11 (main)
$ git push origin main
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 12 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (18/18), 5.45 KiB | 5.45 MiB/s, done.
Total 18 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/MrPlatynum/ProgrammEngineering11.git
 ac86591..c16828b  main -> main
```

Рисунок 18 – Отправка на удаленный репозиторий

Ответы на контрольные вопросы:

1. Каково назначение функций в языке программирования Python?

Функции в Python используются для группировки блоков кода с целью выполнения определенной задачи. Они помогают упростить код, делают его модульным, повторно используемым и обеспечивают структурирование программы.

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

`def`: Этот оператор используется для определения функций в Python. Он указывает на начало определения функции и принимает имя функции и её параметры.

`return`: Оператор `return` используется для возврата результата выполнения функции. Он завершает выполнение функции и возвращает указанное значение (или значения) в вызывающую часть программы.

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

Локальные переменные: Объявлены внутри функции и существуют только в её контексте. Они не доступны за пределами функции.

Глобальные переменные: Определены вне функций и доступны в любой части программы. Однако для их изменения внутри функции нужно явно указать, что переменная является глобальной.

4. Как вернуть несколько значений из функции Python?

Это можно сделать, вернув кортеж, список или любую другую коллекцию с несколькими значениями.

5. Какие существуют способы передачи значений в функцию?

По позиции: Значения передаются в порядке объявленных параметров функции.

По ключевым словам: Значения передаются с указанием имени параметров функции.

6. Как задать значения аргументов функции по умолчанию?

Это делается путем присвоения значений параметрам функции в её

определении. Если аргумент не будет передан при вызове функции, будет использовано значение по умолчанию.

7. Каково назначение lambda-выражений в Python?

Это анонимные функции, которые позволяют быстро создавать простые функции в одной строке кода.

8. Как осуществляется документирование кода согласно PEP257?

Это стандарт оформления документации в Python. Он предписывает использование строк документации (docstrings) для описания функций, модулей и классов, а также определенный стиль оформления.

9. В чём особенности однострочных и многострочных форм строк документации?

Однострочные строки документации используются для краткого описания, обычно находятся в одной строке под именем функции/модуля/класса.

Многострочные строки документации используют тройные кавычки и позволяют подробно описать функцию/модуль/класс.