

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.5**  
**дисциплины «Основы программной инженерии»**

Выполнил:  
Соколов Михаил Романович  
2 курс, группа ПИЖ-б-о-22-1,  
09.03.04 «Программная инженерия»,  
направленность (профиль) «Разработка  
и сопровождение программного  
обеспечения», очная форма обучения

---

(подпись)

Руководитель практики:  
Богданов С.С., ассистент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

Тема: Работа с кортежами в языке Python.

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

Ход выполнения работы:

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и .gitignore файл для языка 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 (\*).*

**Repository template**

No template ▾

Start your repository with a template repository's contents.

**Owner \*** **Repository name \***

PoTtaTto ▾ / fundamentals\_of\_software

✔ fundamentals\_of\_software\_engineering\_lab2\_5 is available.

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

**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 ▾

Рисунок 1 – Создание репозитория с заданными настройками

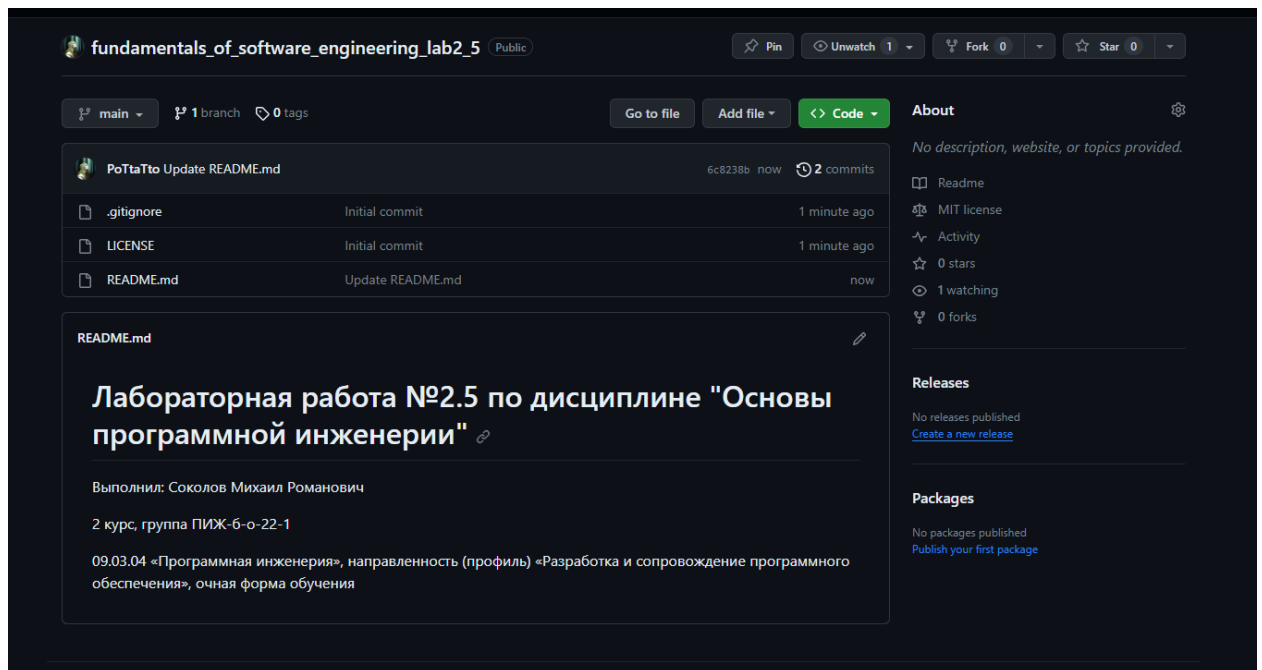


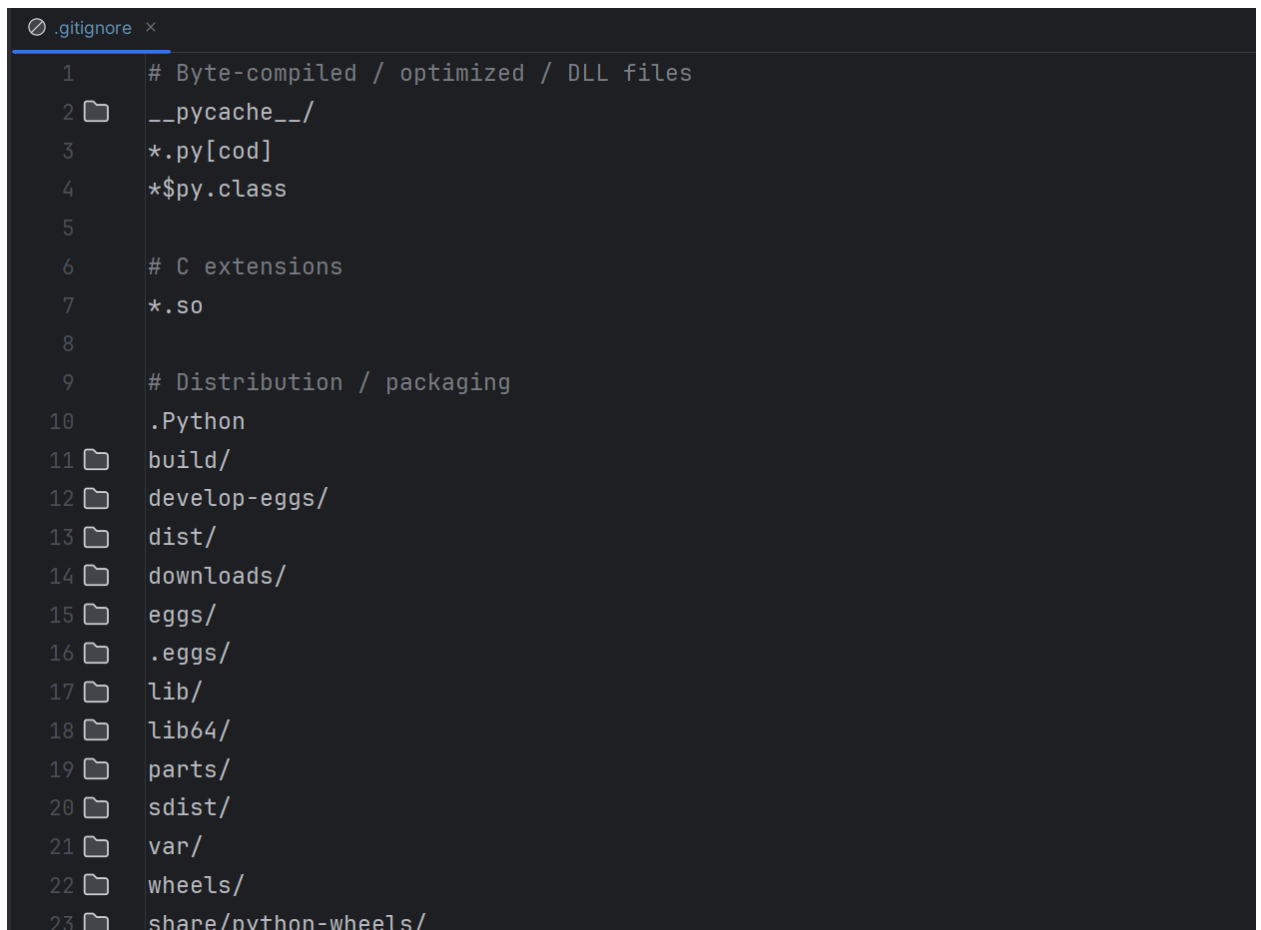
Рисунок 2 – Созданный репозиторий

```
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.5> git clone https://github.com/PoTtaTto/fundamentals_of_software_engineering_lab2_5
Cloning into 'fundamentals_of_software_engineering_lab2_5'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.5> |
```

Рисунок 3 – Клонирование репозитория

```
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.5> git checkout -b develop
Switched to a new branch 'develop'
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.5> fundamentals_of_software_engineering_lab2_5>
```

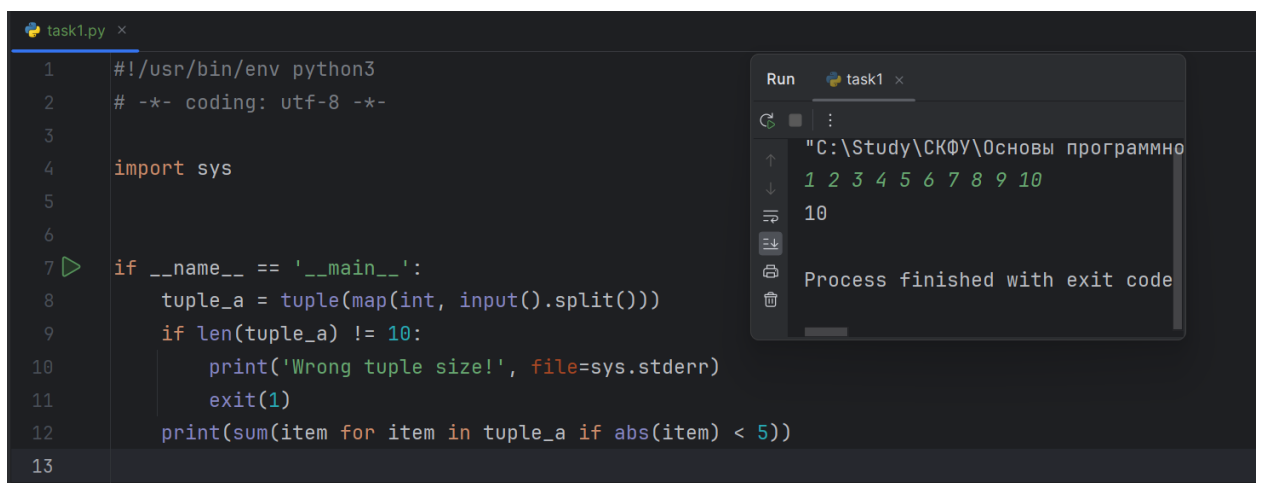
Рисунок 4 – Создание ветки develop, где будут происходить изменения проекта до его полного релиза

A screenshot of a code editor showing a .gitignore file. The file contains a list of patterns to be ignored by Git, including byte-compiled files, C extensions, and various distribution/packaging directories. The lines are numbered from 1 to 23.

```
1 # Byte-compiled / optimized / DLL files
2 __pycache__/
3 *.py[cod]
4 *$py.class
5
6 # C extensions
7 *.so
8
9 # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/nvthon-wheels/
```

Рисунок 5 – Часть .gitignore файла, созданного GitHub

2. Проработаем пример лабораторной работы, фиксируя изменения.  
Создадим для примера отдельный модуль языка Python:

A screenshot showing a Python script named task1.py and its execution. The script prompts the user to enter 10 integers, which are then processed to find the sum of those with an absolute value less than 5. The output shows the input sequence and the final sum of 10.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 if __name__ == '__main__':
8     tuple_a = tuple(map(int, input().split()))
9     if len(tuple_a) != 10:
10         print('Wrong tuple size!', file=sys.stderr)
11         exit(1)
12     print(sum(item for item in tuple_a if abs(item) < 5))
13
```

Run task1 x

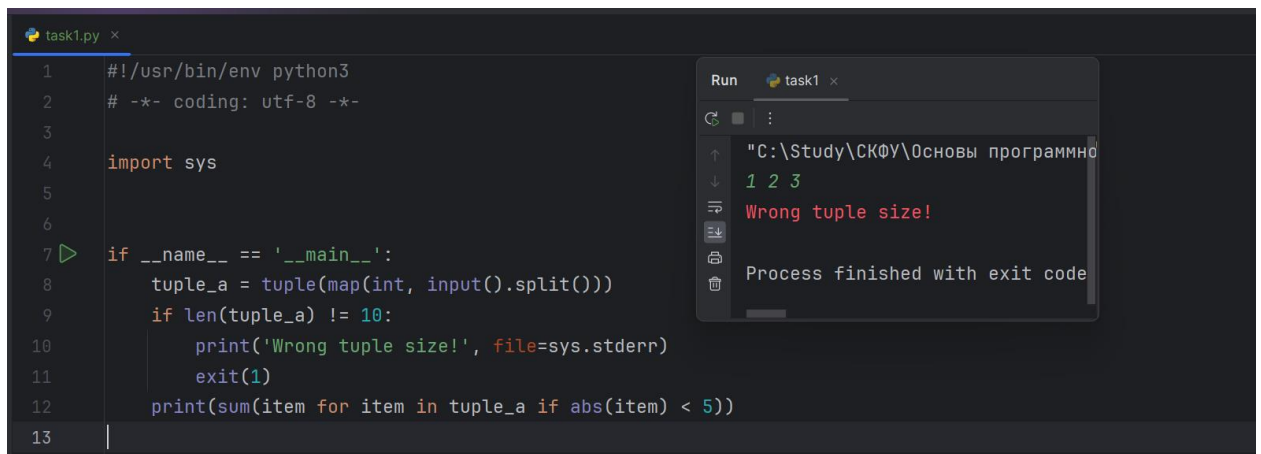
"C:\Study\СКФУ\Основы программирования"

1 2 3 4 5 6 7 8 9 10

10

Process finished with exit code

Рисунок 6 – Ввести кортеж А из 10 элементов, найти сумму элементов, меньших по модулю 5, и вывести ее на экран (задание №1) (1)



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 if __name__ == '__main__':
8     tuple_a = tuple(map(int, input().split()))
9     if len(tuple_a) != 10:
10         print('Wrong tuple size!', file=sys.stderr)
11         exit(1)
12     print(sum(item for item in tuple_a if abs(item) < 5))
13
```

Run task1

"C:\Study\СКФУ\Основы программирования"

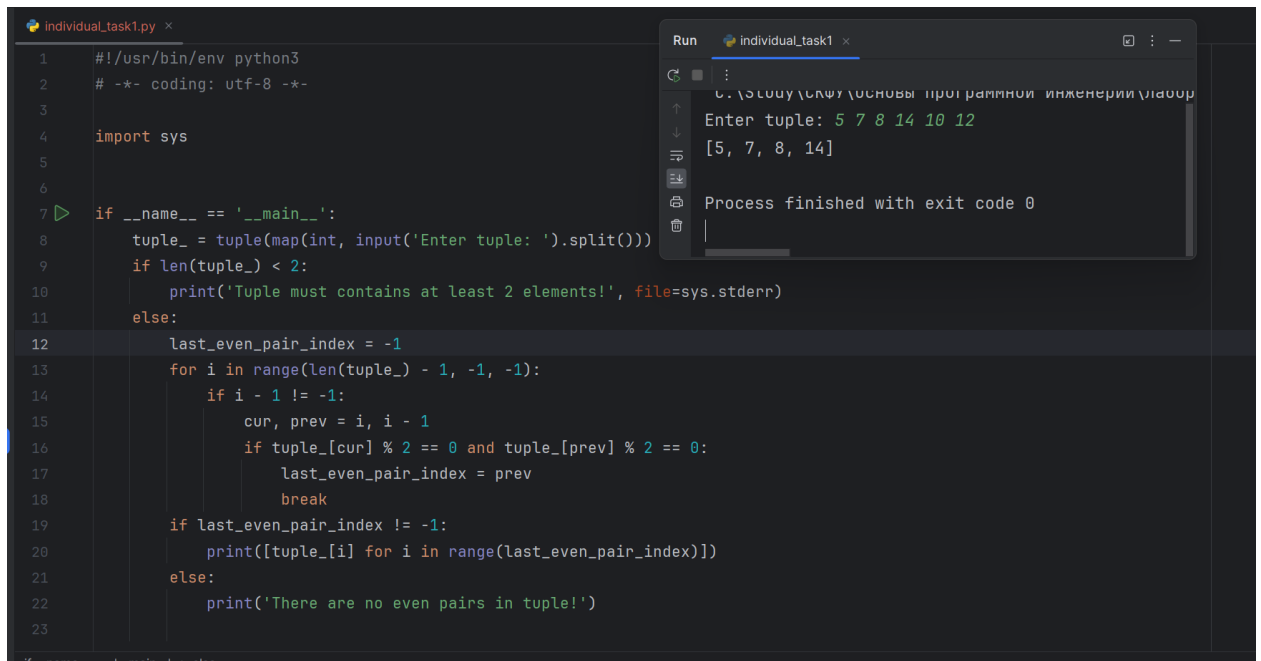
1 2 3

Wrong tuple size!

Process finished with exit code

Рисунок 7 – Ввести кортеж А из 10 элементов, найти сумму элементов, меньших по модулю 5, и вывести ее на экран (задание №1) (2)

### 3. Проработаем индивидуальное задание (вариант №7):



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 if __name__ == '__main__':
8     tuple_ = tuple(map(int, input('Enter tuple: ').split()))
9     if len(tuple_) < 2:
10         print('Tuple must contains at least 2 elements!', file=sys.stderr)
11     else:
12         last_even_pair_index = -1
13         for i in range(len(tuple_) - 1, -1, -1):
14             if i - 1 != -1:
15                 cur, prev = i, i - 1
16                 if tuple_[cur] % 2 == 0 and tuple_[prev] % 2 == 0:
17                     last_even_pair_index = prev
18                     break
19         if last_even_pair_index != -1:
20             print([tuple_[i] for i in range(last_even_pair_index)])
21         else:
22             print('There are no even pairs in tuple!')
23
```

Run individual\_task1

"C:\Study\СКФУ\Основы программирования\инженерии\лаборатории"

Enter tuple: 5 7 8 14 10 12

[5, 7, 8, 14]

Process finished with exit code 0

Рисунок 8 – Найти в кортеже целых чисел последнюю пару четных чисел и напечатать все элементы до нее (индивидуальное задание №1) (1)

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 if __name__ == '__main__':
8     tuple_ = tuple(map(int, input('Enter tuple: ').split()))
9     if len(tuple_) < 2:
10         print('Tuple must contains at least 2 elements!', file=sys.stderr)
11     else:
12         last_even_pair_index = -1
13         for i in range(len(tuple_) - 1, -1, -1):
14             if i - 1 != -1:
15                 cur, prev = i, i - 1
16                 if tuple_[cur] % 2 == 0 and tuple_[prev] % 2 == 0:
17                     last_even_pair_index = prev
18                     break
19         if last_even_pair_index != -1:
20             print([tuple_[i] for i in range(last_even_pair_index)])
21         else:
22             print('There are no even pairs in tuple!')
```

Run individual\_task1

Enter tuple: 1

Tuple must contains at least 2 elements!

Process finished with exit code 0

Рисунок 9 – Найти в кортеже целых чисел последнюю пару четных чисел и напечатать все элементы до нее (индивидуальное задание №1) (2)

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 if __name__ == '__main__':
8     tuple_ = tuple(map(int, input('Enter tuple: ').split()))
9     if len(tuple_) < 2:
10         print('Tuple must contains at least 2 elements!', file=sys.stderr)
11     else:
12         last_even_pair_index = -1
13         for i in range(len(tuple_) - 1, -1, -1):
14             if i - 1 != -1:
15                 cur, prev = i, i - 1
16                 if tuple_[cur] % 2 == 0 and tuple_[prev] % 2 == 0:
17                     last_even_pair_index = prev
18                     break
19         if last_even_pair_index != -1:
20             print([tuple_[i] for i in range(last_even_pair_index)])
21         else:
22             print('There are no even pairs in tuple!')
```

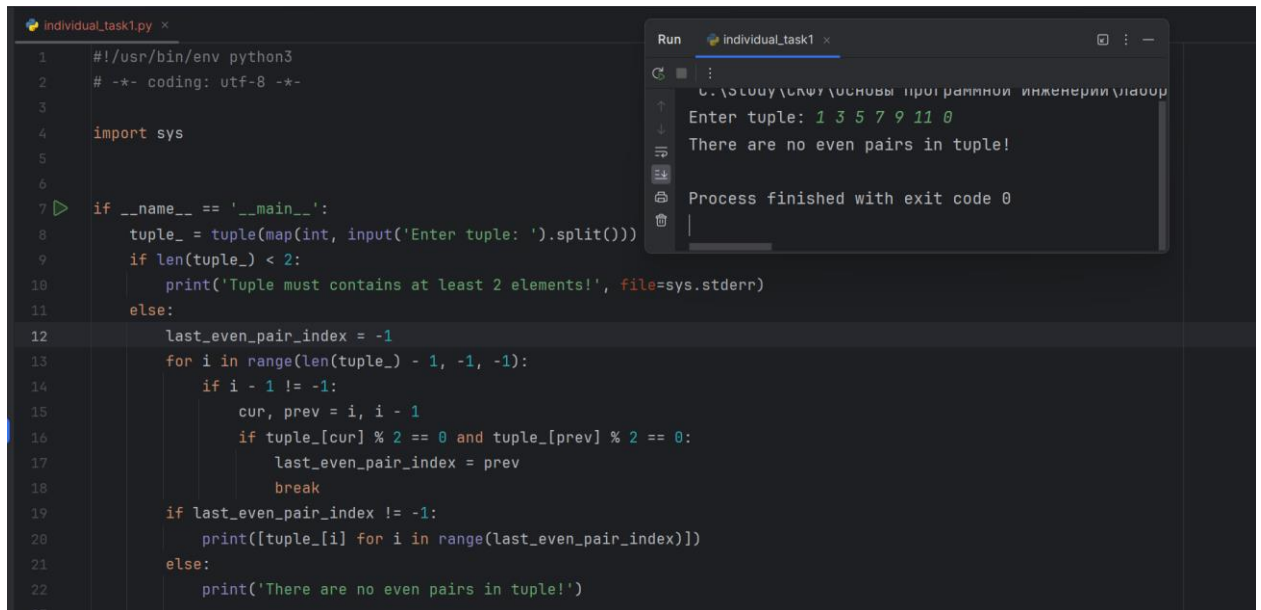
Run individual\_task1

Enter tuple: 2 4

[]

Process finished with exit code 0

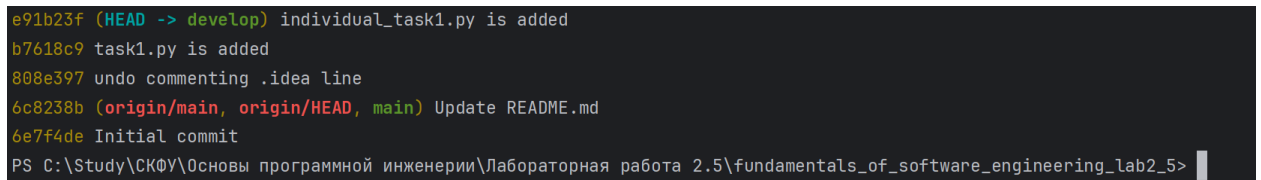
Рисунок 10 – Найти в кортеже целых чисел последнюю пару четных чисел и напечатать все элементы до нее (индивидуальное задание №1) (3)



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 if __name__ == '__main__':
8     tuple_ = tuple(map(int, input('Enter tuple: ').split()))
9     if len(tuple_) < 2:
10         print('Tuple must contains at least 2 elements!', file=sys.stderr)
11     else:
12         last_even_pair_index = -1
13         for i in range(len(tuple_) - 1, -1, -1):
14             if i - 1 != -1:
15                 cur, prev = i, i - 1
16                 if tuple_[cur] % 2 == 0 and tuple_[prev] % 2 == 0:
17                     last_even_pair_index = prev
18                     break
19         if last_even_pair_index != -1:
20             print([tuple_[i] for i in range(last_even_pair_index)])
21         else:
22             print('There are no even pairs in tuple!')
```

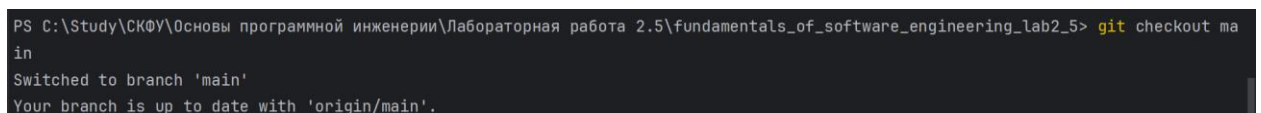
Рисунок 11 – Найти в кортеже целых чисел последнюю пару четных чисел и напечатать все элементы до нее (индивидуальное задание №1) (4)

#### 4. Слив веток и push на удаленные репозиторий:



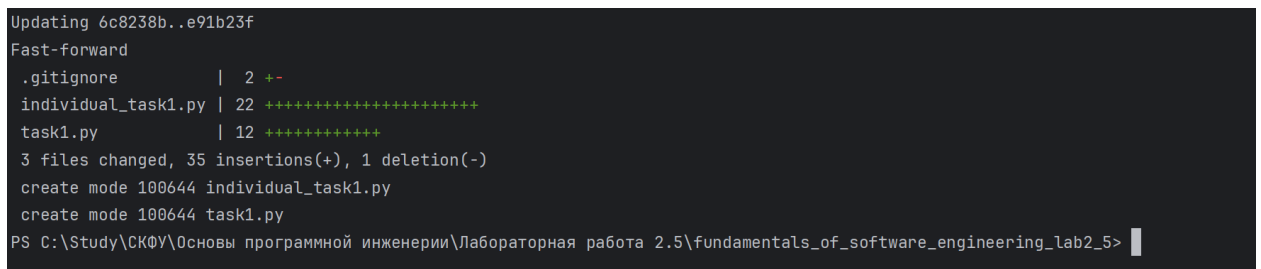
```
e91b23f (HEAD -> develop) individual_task1.py is added
b7618c9 task1.py is added
808e397 undo commenting .idea line
6c8238b (origin/main, origin/HEAD, main) Update README.md
6e7f4de Initial commit
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.5\fundamentals_of_software_engineering_lab2_5>
```

Рисунок 12 – Коммиты проекта



```
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.5\fundamentals_of_software_engineering_lab2_5> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Рисунок 13 – Переключение на ветку main



```
Updating 6c8238b..e91b23f
Fast-forward
 .gitignore      | 2 +-
 individual_task1.py | 22 ++++++
 task1.py         | 12 ++++++
 3 files changed, 35 insertions(+), 1 deletion(-)
 create mode 100644 individual_task1.py
 create mode 100644 task1.py
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.5\fundamentals_of_software_engineering_lab2_5>
```

Рисунок 14 – Слияние веток

```
Enumerating objects: 11, done.  
Counting objects: 100% (11/11), done.  
Delta compression using up to 12 threads  
Compressing objects: 100% (9/9), done.  
Writing objects: 100% (9/9), 1.30 KiB | 1.30 MiB/s, done.  
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (3/3), completed with 1 local object.  
To https://github.com/PoTtaTto/fundamentals\_of\_software\_engineering\_lab2\_5  
 6c8238b..e91b23f  main -> main  
PS C:\Study\СКФ\Основы программной инженерии\Лабораторная работа 2.5\fundamentals_of_software_engineering_lab2_5>
```

Рисунок 15 – Отправка на удаленный репозиторий



Ответы на контрольные вопросы:

1. Что такое списки в языке Python?

Списки в Python – упорядоченный изменяемый набор объектов произвольных типов, пронумерованных от 0. Они используются для хранения и работы с данными.

2. Каково назначение кортежей в языке Python?

Кортеж (tuple) – это неизменяемая структура данных, которая по своему подобию очень похожа на список. Существует несколько причин, по которым стоит использовать кортежи вместо списков. Одна из них – это обезопасить данные от случайного изменения. Если мы получили откуда-то массив данных, и у нас есть желание поработать с ним, но при этом непосредственно менять данные мы не собираемся, тогда, это как раз тот случай, когда кортежи придутся как нельзя кстати. Используя их в данной задаче, мы дополнительно получаем сразу несколько бонусов – во-первых, это экономия места. Дело в том, что кортежи в памяти занимают меньший объем по сравнению со списками.

3. Как осуществляется создание кортежей?

Пример: `a = ()` или `a = tuple()` или `a = (1, ...)`

4. Как осуществляется доступ к элементам кортежа?

Доступ к элементам кортежа осуществляется также как к элементам списка – через указание индекса.

5. Зачем нужна распаковка (деструктуризация) кортежа?

Обращение по индексу, это не самый удобный способ работы с кортежами. Дело в том, что кортежи часто содержат значения разных типов, и помнить, по какому индексу что лежит — очень непросто. Поэтому можно воспользоваться следующей конструкцией для извлечения значений кортежа: `number, string = (42, 'Hello')`.

6. Какую роль играют кортежи в множественном присваивании?

Используя множественное присваивание, можно проверить интересный трюк: обмен значениями между двумя переменными.

7. Как выбрать элементы кортежа с помощью среза?

С помощью операции взятия среза можно получить другой кортеж. Общая форма операции взятия среза для кортежа следующая:  $T2 = T1[i:j]$ , где  $i$  и  $j$  – это верхняя и нижняя границы среза.

8. Как выполняется конкатенация и повторение кортежей?

Конкатенация соединяет кортежи в один (знак +), а повторение создает новый кортеж с дублированными  $n$ -раз значениями кортежа (знак \*).

9. Как выполняется обход элементов кортежа?

Также как и в списках: через циклы `for` и `while`.

10. Как проверить принадлежность элемента кортежу?

Оператором `in`.

11. Какие методы работы с кортежами Вам известны?

Методы: `count()` – подсчет вхождений элемента в кортеж; `index()` – поиск индекса первого вхождения заданного аргумента и многие другие. С ними можно ознакомиться в официальной документации языка Python.

12. Допустимо ли использование функций агрегации таких как `len()` , `sum()` и т. д. при работе с кортежами?

Да, допустимо.

13. Как создать кортеж с помощью спискового включения.

Пример: `tuple_ = tuple([number for number in range(100)])`