

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 №8**  
**дисциплины «Основы программной инженерии»**

Выполнила:  
Панюкова Ксения Юрьевна  
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:  
Воронкин Р. А., доцент кафедры  
инфокоммуникаций

---

(подпись)

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

Ставрополь, 2023 г.

# Ход работы

## 1. Я изучила теоретический материал работы

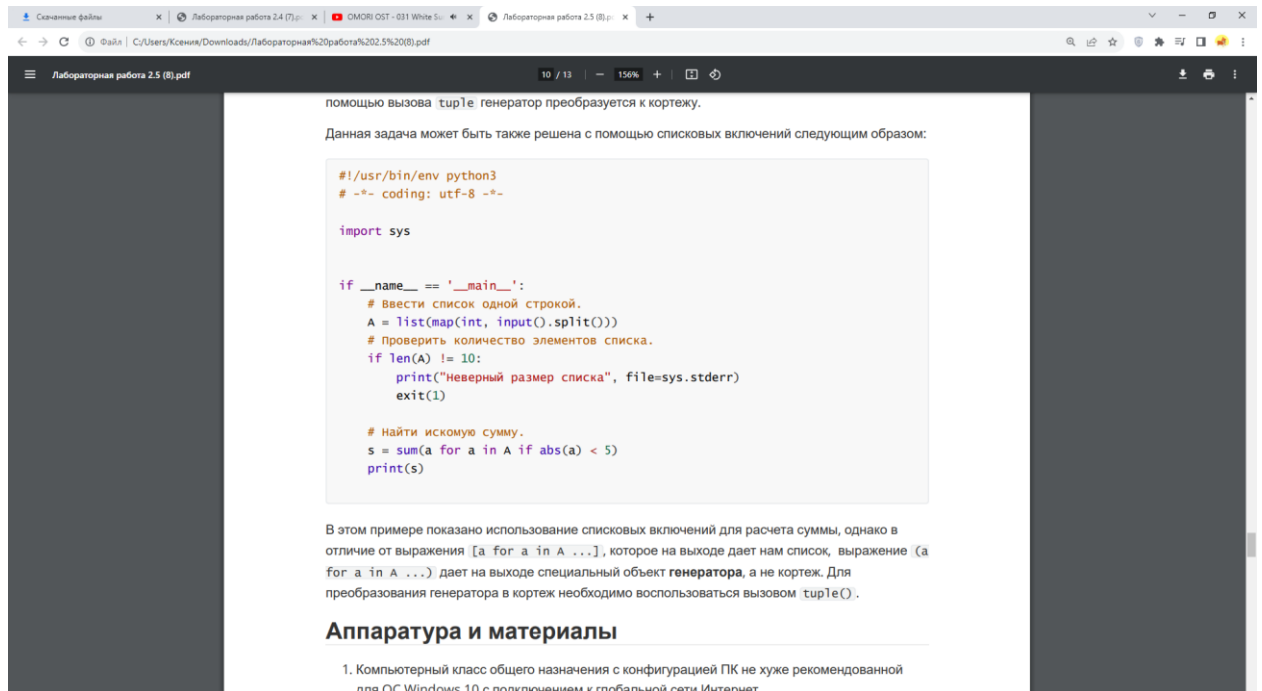


Рисунок 1.1 – Изучение материала для лабораторной работы

2. Создала общедоступный репозиторий на GitHub, в котором будет использована лицензия 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 \*

Repository name \*

MrFinicheck

/

PythonsFavoriteFoodIsAntelope

✔ Your new repository will be created as Pythons-FavoriteFoodIsAntelope.

The repository name can only contain ASCII letters, digits, and the characters ., -, and \_.

Great repository names are short and memorable. Need inspiration? How about **potential-fortnight** ?

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 2.1 – Настройка репозитория

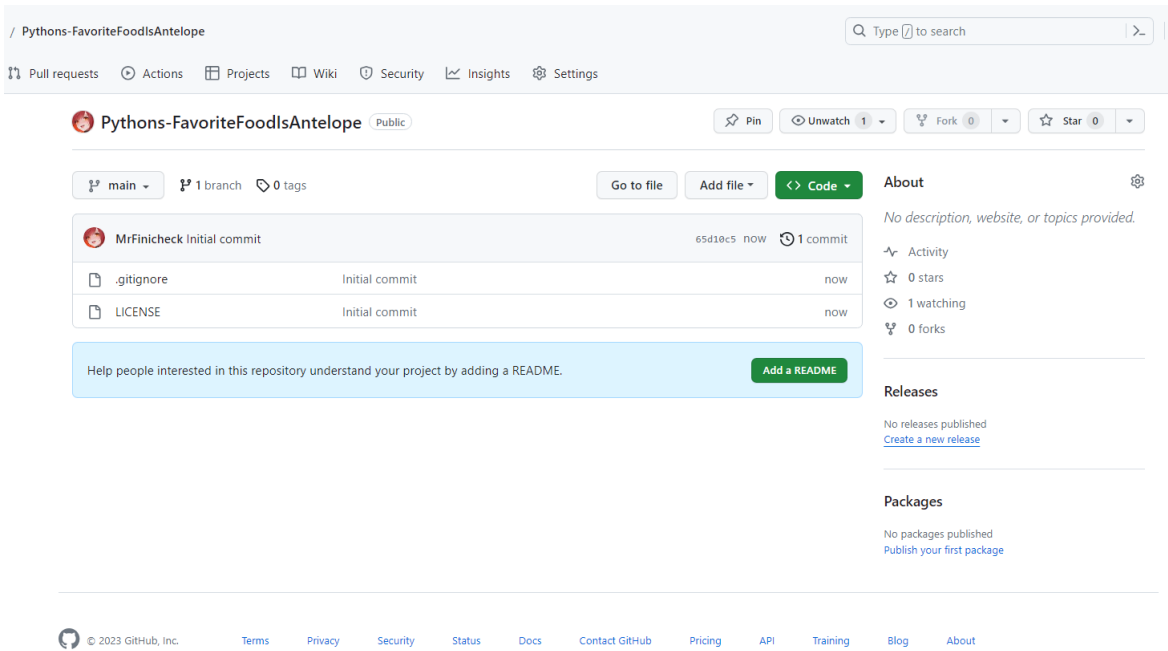


Рисунок 2.2 – Готовый репозиторий

### 3. Выполняю клонирование созданного репозитория

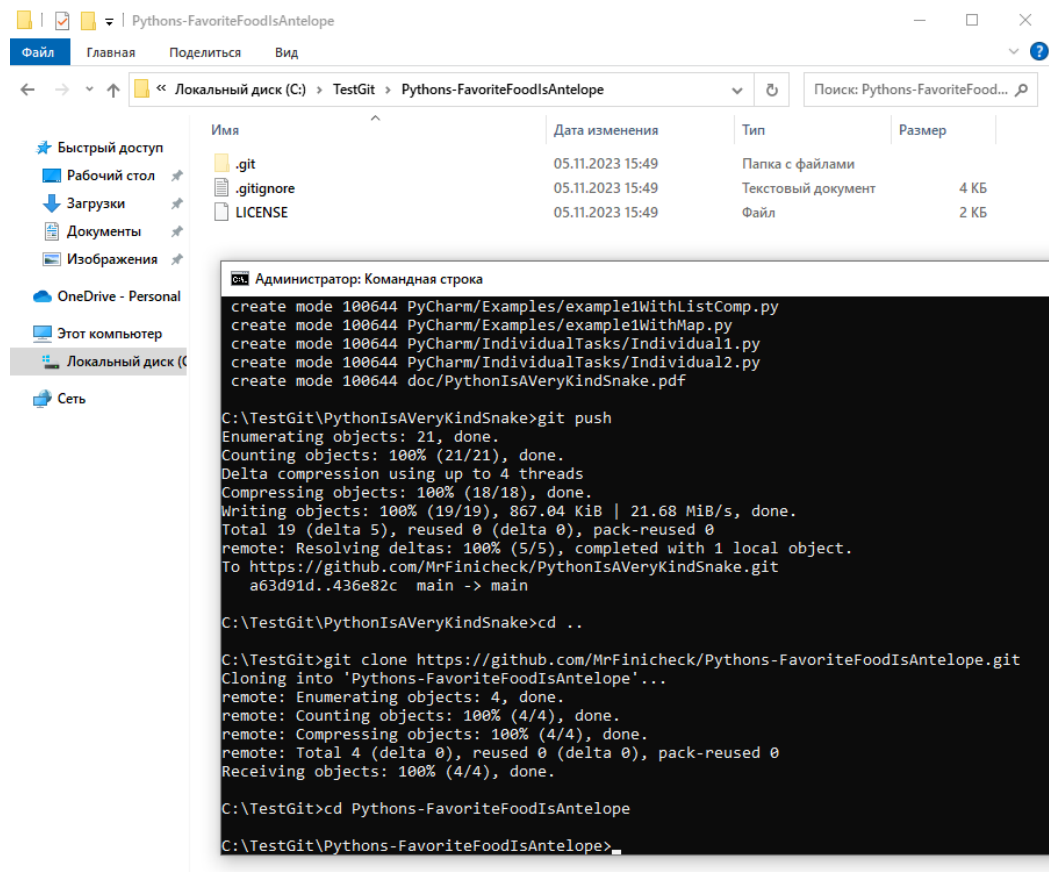
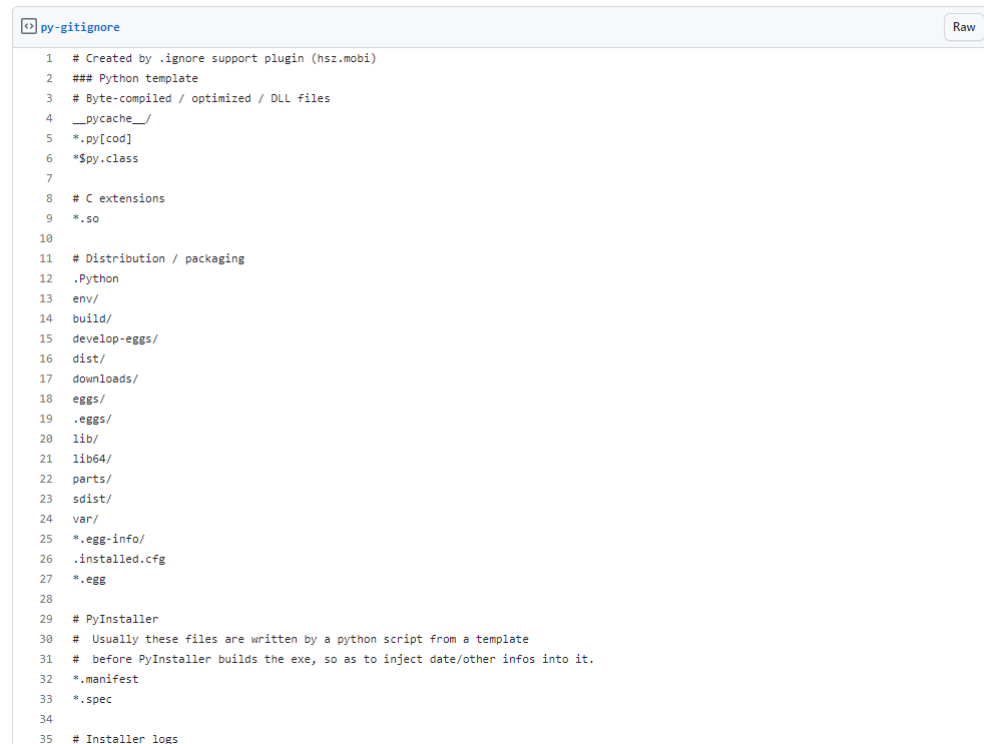


Рисунок 3.1 – Клонирование репозитория на локальный диск

### 4. Дополнила файл .gitignore необходимыми правилами для работы с IDE PyCharm



```

1 # Created by .ignore support plugin (hsz.mobi)
2 ### Python template
3 # Byte-compiled / optimized / DLL files
4 __pycache__/
5 *.py[cod]
6 *$py.class
7
8 # C extensions
9 *.so
10
11 # Distribution / packaging
12 .Python
13 env/
14 build/
15 develop-eggs/
16 dist/
17 downloads/
18 eggs/
19 .eggs/
20 lib/
21 lib64/
22 parts/
23 sdist/
24 var/
25 *.egg-info/
26 .installed.cfg
27 *.egg
28
29 # PyInstaller
30 # Usually these files are written by a python script from a template
31 # before PyInstaller builds the exe, so as to inject date/other infos into it.
32 *.manifest
33 *.spec
34
35 # Installer logs

```

Рисунок 4.1 – .gitignore для IDE PyCharm

## 5. Организовала свой репозиторий в соответствии с моделью ветвления git-flow

```

C:\TestGit\Pythons-FavoriteFoodIsAntelope>git branch develop

C:\TestGit\Pythons-FavoriteFoodIsAntelope>git checkout develop
Switched to branch 'develop'

C:\TestGit\Pythons-FavoriteFoodIsAntelope>git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/MrFinicheck/Pythons-FavoriteFoodIsAntelope/pull/new/develop
remote:
To https://github.com/MrFinicheck/Pythons-FavoriteFoodIsAntelope.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

C:\TestGit\Pythons-FavoriteFoodIsAntelope>

```

Рисунок 5.1 – Создание ветки develop от ветки main



```
example1WithListComp.py  example1WithMap.py  Example2.py  Individual1.py  Individual2.py

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      # Ввести кортеж одной строкой.
8      A = tuple(map(int, input().split()))
9      # Проверить количество элементов кортежа.
10     if len(A) != 10:
11         print("Неверный размер кортежа", file=sys.stderr)
12         exit(1)
13
14     # Найти искомую сумму.
15     s = 0
16     for item in A:
17         if abs(item) < 5:
18             s += item
19
20     print(s)
```

Рисунок 7.1 – Проработка примера 1 с применением map()

```
example1WithMap.py  Example2.py  Individual1.py  Individual2.py  Example1Without.py

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      # Ввести список одной строкой.
8      A = list(map(int, input().split()))
9      # Проверить количество элементов списка.
10     if len(A) != 10:
11         print("Неверный размер списка", file=sys.stderr)
12         exit(1)
13
14     # Найти искомую сумму.
15     s = sum(a for a in A if abs(a) < 5)
16     print(s)
```

Рисунок 7.2 – Проработка примера 1 с применением списковых включений

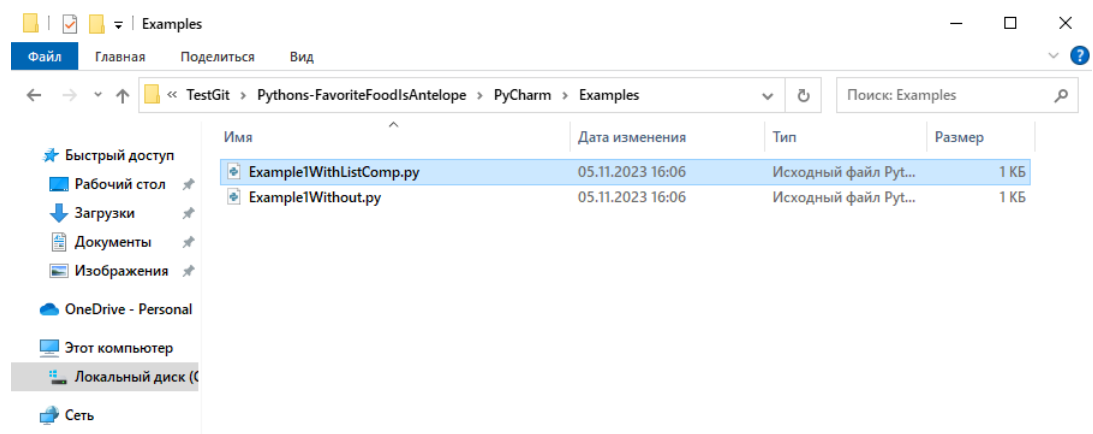


Рисунок 7.4 – Создание отдельных модулей для каждого из примеров

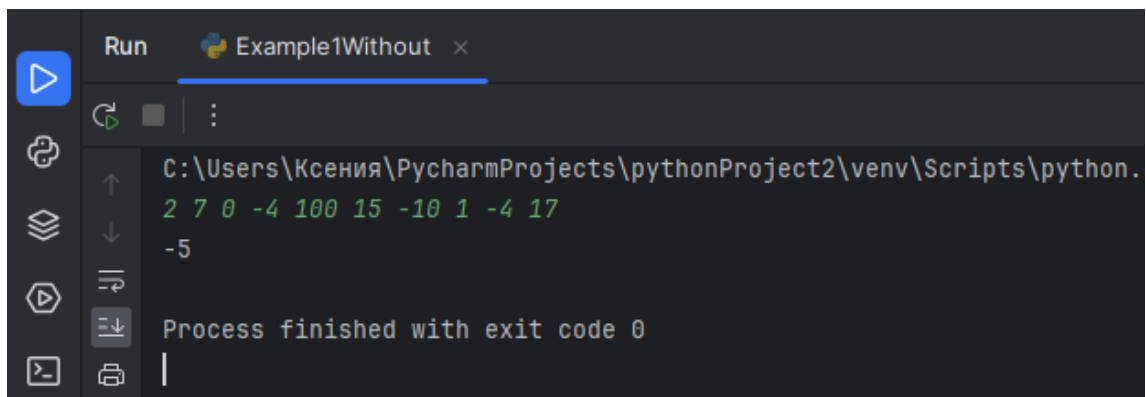
```
C:\TestGit\Pythons-FavoriteFoodIsAntelope>git add PyCharm

C:\TestGit\Pythons-FavoriteFoodIsAntelope>git commit -m"adding examples"
[develop c6dbbbc] adding examples
3 files changed, 38 insertions(+)
create mode 100644 PyCharm/Examples/Example1WithListComp.py
create mode 100644 PyCharm/Examples/Example1Without.py
delete mode 100644 PyCharm/python.txt

C:\TestGit\Pythons-FavoriteFoodIsAntelope>
```

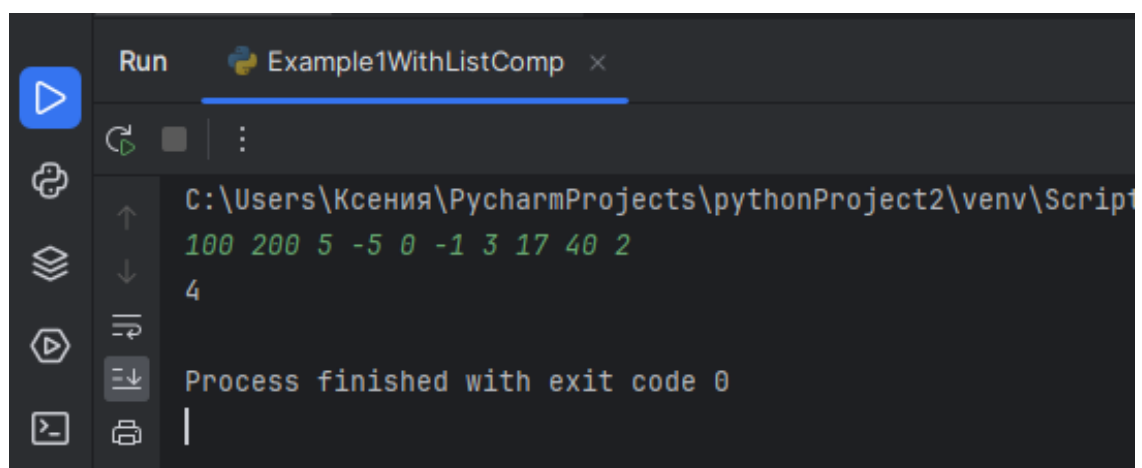
Рисунок 7.5 – Фиксирование изменений в репозитории

8. Привела в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных, вводимых с клавиатуры.

A screenshot of the PyCharm Run console. The title bar shows 'Run' and 'Example1Without'. The console output shows the command 'C:\Users\Ксения\PycharmProjects\pythonProject2\venv\Scripts\python.' followed by the input '2 7 0 -4 100 15 -10 1 -4 17' and the output '-5'. The message 'Process finished with exit code 0' is displayed at the bottom.

```
Run Example1Without x
C:\Users\Ксения\PycharmProjects\pythonProject2\venv\Scripts\python.
2 7 0 -4 100 15 -10 1 -4 17
-5
Process finished with exit code 0
```

Рисунок 8.1 – Результат примера 1 с применением map()

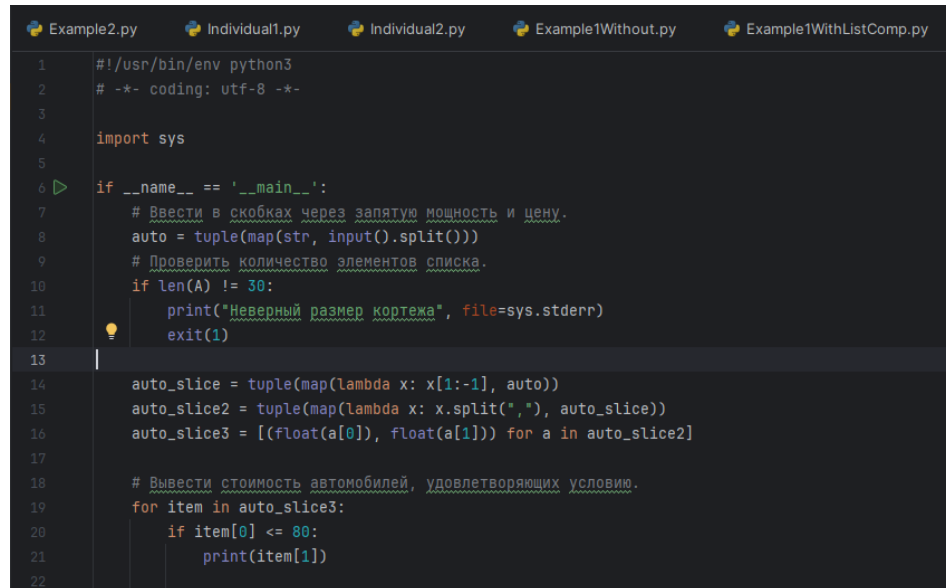
A screenshot of the PyCharm Run console. The title bar shows 'Run' and 'Example1WithListComp'. The console output shows the command 'C:\Users\Ксения\PycharmProjects\pythonProject2\venv\Scripts\python.' followed by the input '100 200 5 -5 0 -1 3 17 40 2' and the output '4'. The message 'Process finished with exit code 0' is displayed at the bottom.

```
Run Example1WithListComp x
C:\Users\Ксения\PycharmProjects\pythonProject2\venv\Scripts\python.
100 200 5 -5 0 -1 3 17 40 2
4
Process finished with exit code 0
```

Рисунок 8.2 – Результат примера 2 с применением списковых включений



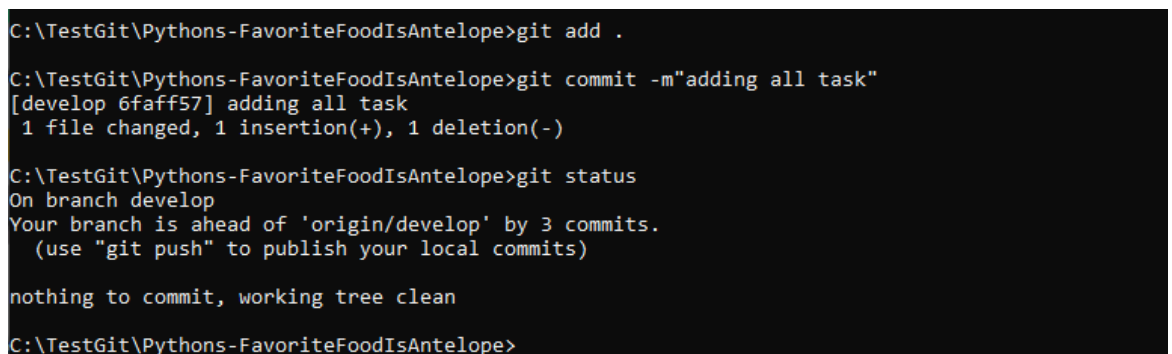
9. Привела в отчете скриншоты работы программ решения индивидуальных заданий



```
Example2.py Individual1.py Individual2.py Example1Without.py Example1WithListComp.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      # Ввести в скобках через запятую мощность и цену.
8      auto = tuple(map(str, input().split()))
9      # Проверить количество элементов списка.
10     if len(A) != 30:
11         print("Неверный размер кортежа", file=sys.stderr)
12         exit(1)
13
14     auto_slice = tuple(map(lambda x: x[1:-1], auto))
15     auto_slice2 = tuple(map(lambda x: x.split(","), auto_slice))
16     auto_slice3 = [(float(a[0]), float(a[1])) for a in auto_slice2]
17
18     # Вывести стоимость автомобилей, удовлетворяющих условию.
19     for item in auto_slice3:
20         if item[0] <= 80:
21             print(item[1])
22
```

Рисунок 9.1 – Код программы INdividual1.py в IDE PyCharm

10. Зафиксировала сделанные изменения в репозитории



```
C:\TestGit\Pythons-FavoriteFoodIsAntelope>git add .
C:\TestGit\Pythons-FavoriteFoodIsAntelope>git commit -m"adding all task"
[develop 6fa9f57] adding all task
 1 file changed, 1 insertion(+), 1 deletion(-)
C:\TestGit\Pythons-FavoriteFoodIsAntelope>git status
On branch develop
Your branch is ahead of 'origin/develop' by 3 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
C:\TestGit\Pythons-FavoriteFoodIsAntelope>
```

Рисунок 10.1 – Коммит файлов в репозитории git

## Контрольные вопросы

1. Что такое кортежи в языке Python?

Кортеж (tuple) – это неизменяемая структура данных, которая по своему подобию очень похожа на список. С кортежем мы не можем производить такие операции, т. к. элементы его изменять нельзя.

2. Каково назначение кортежей в языке Python?

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.

Во-вторых – прирост производительности, который связан с тем, что кортежи работают быстрее, чем списки (т. е. на операции перебора элементов и т. п. будет тратиться меньше времени). Важно также отметить, что кортежи можно использовать в качестве ключа у словаря.

### 3. Как осуществляется создание кортежей?

Кортеж с заданным содержанием создается так же, как список, только вместо квадратных скобок используются круглые. При желании можно воспользоваться функцией `tuple()`.

### 4. Как осуществляется доступ к элементам кортежа?

Доступ к элементам кортежа осуществляется так же, 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. В этом случае можно воспользоваться деструктуризацией.

### 6. Какую роль играют кортежи в множественном присваивании?

Благодаря тому, что кортежи легко собирать и разбирать, в Python удобно делать такие вещи, как множественное присваивание. 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]`

8. Как выполняется конкатенация и повторение кортежей?

Для кортежей можно выполнять операцию конкатенации, которая обозначается символом `+`. Кортеж может быть образован путем операции повторения, обозначаемой символом `*`.

9. Как выполняется обход элементов кортежа?

Элементы кортежа можно последовательно просмотреть с помощью операторов цикла `while` или `for`.

10. Как проверить принадлежность элемента кортежу?

Проверить принадлежность элемента кортежу можно с помощью операции `in`.

11. Какие методы работы с кортежами Вам известны?

Чтобы получить индекс (позицию) элемента в кортеже, нужно использовать метод `index()`. Чтобы определить количество вхождений заданного элемента в кортеж используется метод `count`

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

Да, можно использовать функции агрегации, такие как `len()`, `sum()` и другие, при работе с кортежами в Python.

13. Как создать кортеж с помощью спискового включения.

`my_tuple = tuple(expression for item in iterable)`