

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 №6
дисциплины «Основы программной инженерии»

Выполнил:

Звездин Алексей Сергеевич
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 изучил теоретический материал работы

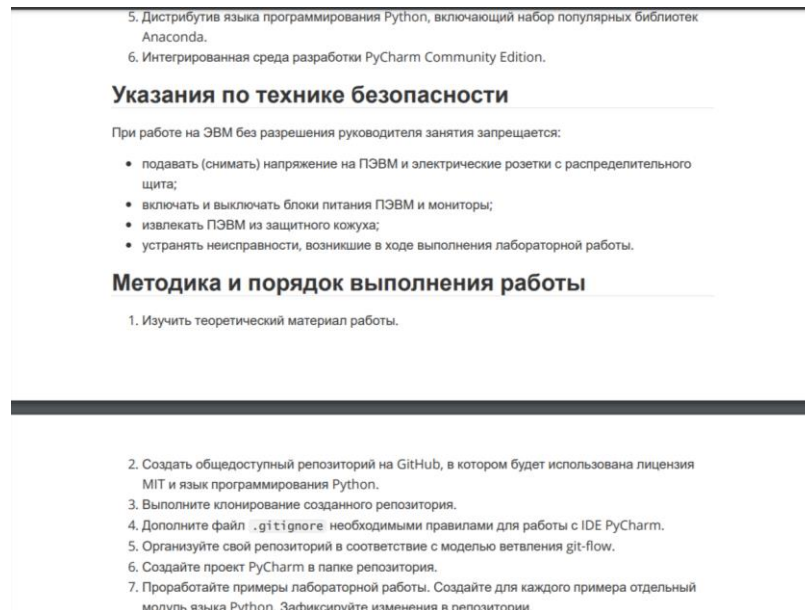


Рисунок 1.1 – Изучение материала для лабораторной работы

2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python

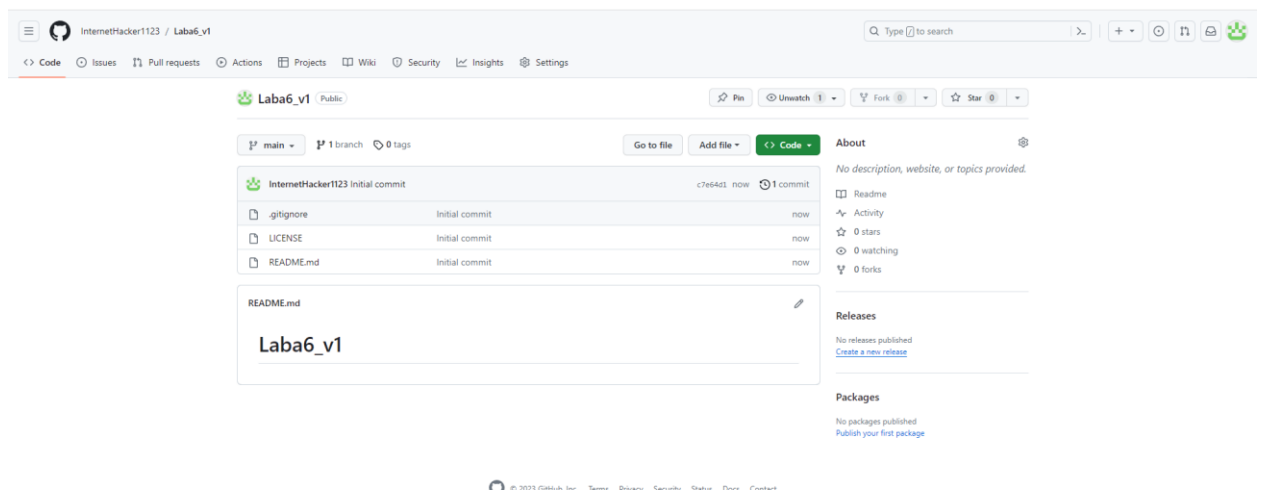
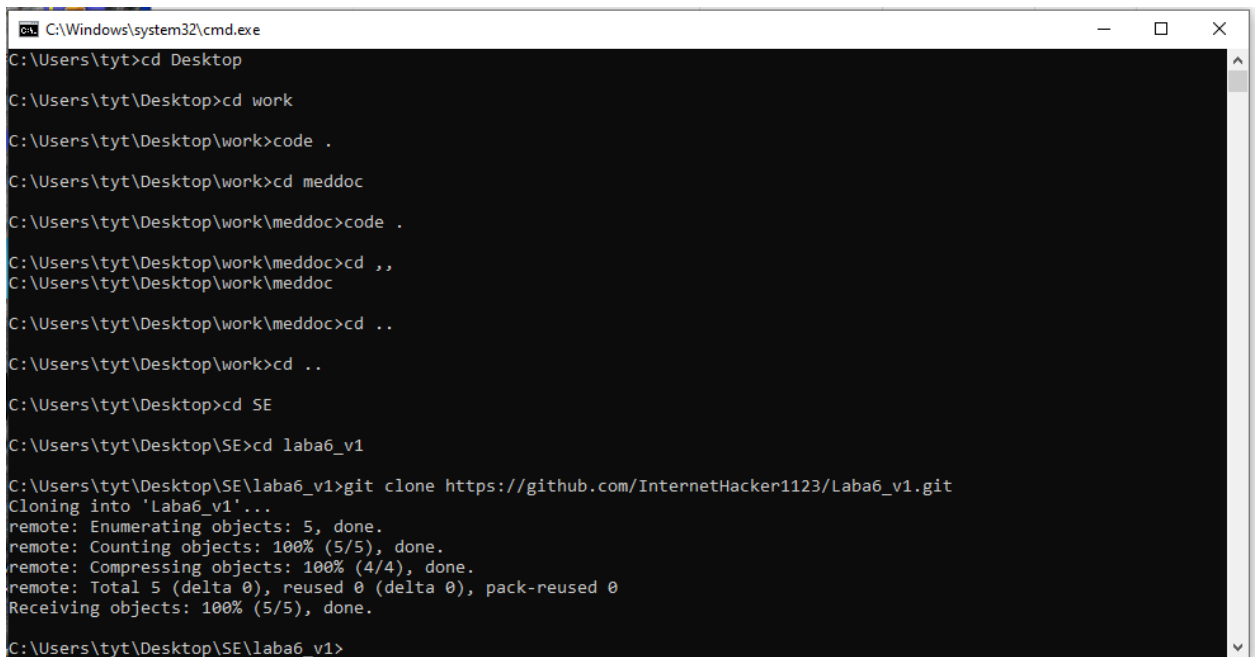


Рисунок 2.1 – Готовый репозиторий

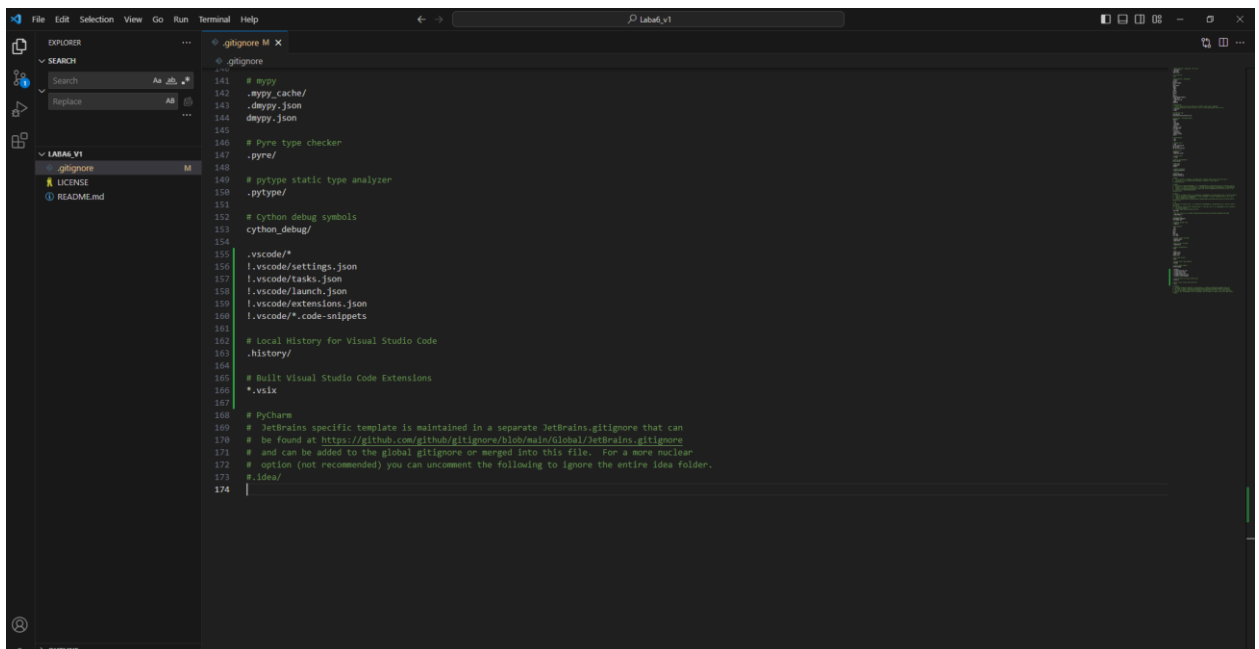
3. Выполняю клонирование созданного репозитория



```
C:\Windows\system32\cmd.exe
C:\Users\tyt>cd Desktop
C:\Users\tyt\Desktop>cd work
C:\Users\tyt\Desktop\work>code .
C:\Users\tyt\Desktop\work>cd meddoc
C:\Users\tyt\Desktop\work\meddoc>code .
C:\Users\tyt\Desktop\work\meddoc>cd ,,
C:\Users\tyt\Desktop\work\meddoc>cd ..
C:\Users\tyt\Desktop\work>cd ..
C:\Users\tyt\Desktop>cd SE
C:\Users\tyt\Desktop\SE>cd laba6_v1
C:\Users\tyt\Desktop\SE\laba6_v1>git clone https://github.com/InternetHacker1123/Laba6_v1.git
Cloning into 'Laba6_v1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
C:\Users\tyt\Desktop\SE\laba6_v1>
```

Рисунок 3.1 – Клонирование репозитория на локальный диск

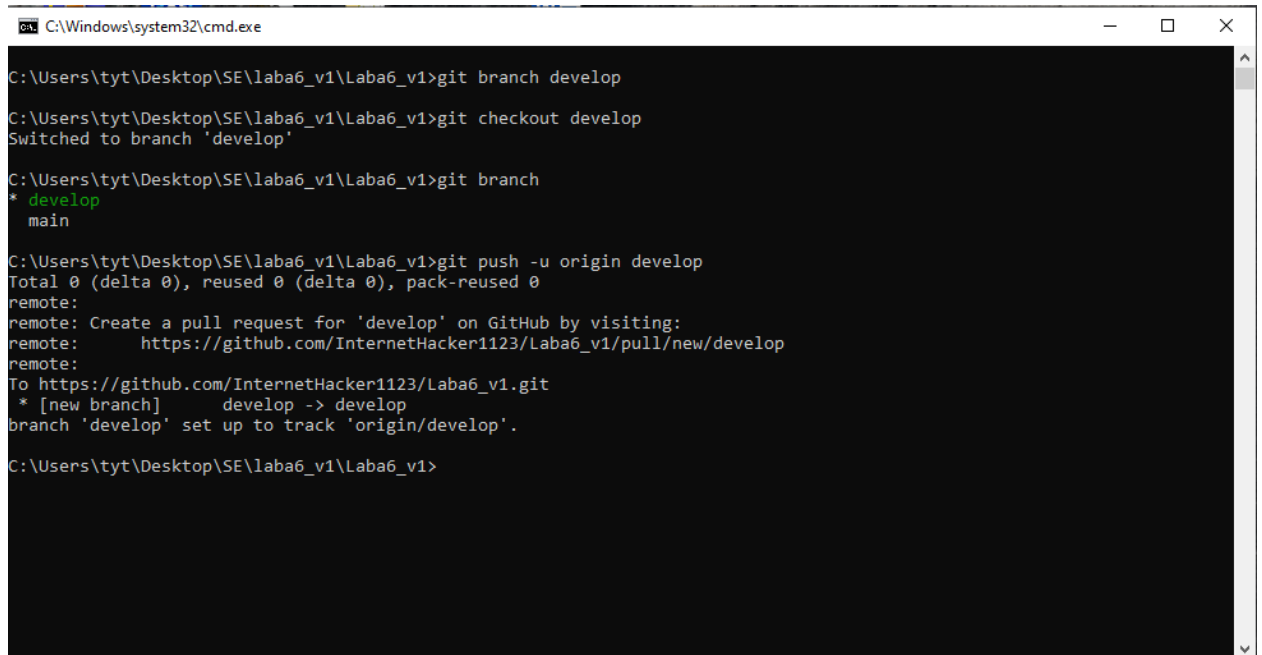
4. Дополнил файл .gitignore необходимыми правилами для работы с VS Code



```
.gitignore
141 # mypy
142 .mypy_cache/
143 .mypy.json
144 .mypy.json
145
146 # Pyre type checker
147 .pyre/
148
149 # pytype static type analyzer
150 .pytype/
151
152 # Cython debug symbols
153 cython_debug/
154
155 # vscode/*
156 !.vscode/settings.json
157 !.vscode/tasks.json
158 !.vscode/launch.json
159 !.vscode/extensions.json
160 !.vscode/*.code-snippets
161
162 # Local History for Visual Studio Code
163 .history/
164
165 # Built Visual Studio Code Extensions
166 *.vsix
167
168 # PyCharm
169 # JetBrains specific template is maintained in a separate JetBrains.gitignore that can
170 # be found at https://github.com/JetBrains/idea/blob/main/global/jetbrains.gitignore
171 # and can be added to the global .gitignore or merged into this file. For a more nuclear
172 # option (not recommended) you can uncomment the following to ignore the entire idea folder.
173 # .idea/
174
```

Рисунок 4.1 – .gitignore для VS Code

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



```
C:\Windows\system32\cmd.exe

C:\Users\tyt\Desktop\SE\laba6_v1\Laba6_v1>git branch develop
C:\Users\tyt\Desktop\SE\laba6_v1\Laba6_v1>git checkout develop
Switched to branch 'develop'

C:\Users\tyt\Desktop\SE\laba6_v1\Laba6_v1>git branch
* develop
  main

C:\Users\tyt\Desktop\SE\laba6_v1\Laba6_v1>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/InternetHacker1123/Laba6_v1/pull/new/develop
remote:
To https://github.com/InternetHacker1123/Laba6_v1.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

C:\Users\tyt\Desktop\SE\laba6_v1\Laba6_v1>
```

Рисунок 5.1 – Создание ветки develop от ветки main

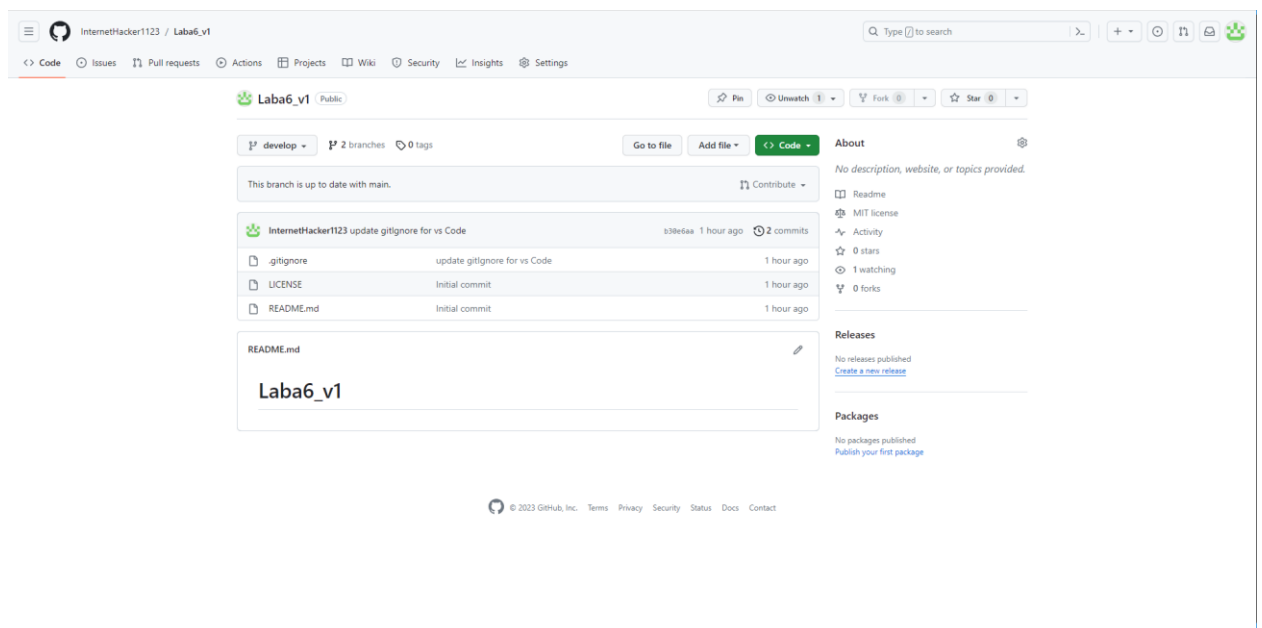


Рисунок 5.2 – Ветка develop на GitHub

6. Создал проект PyCharm в папке репозитория

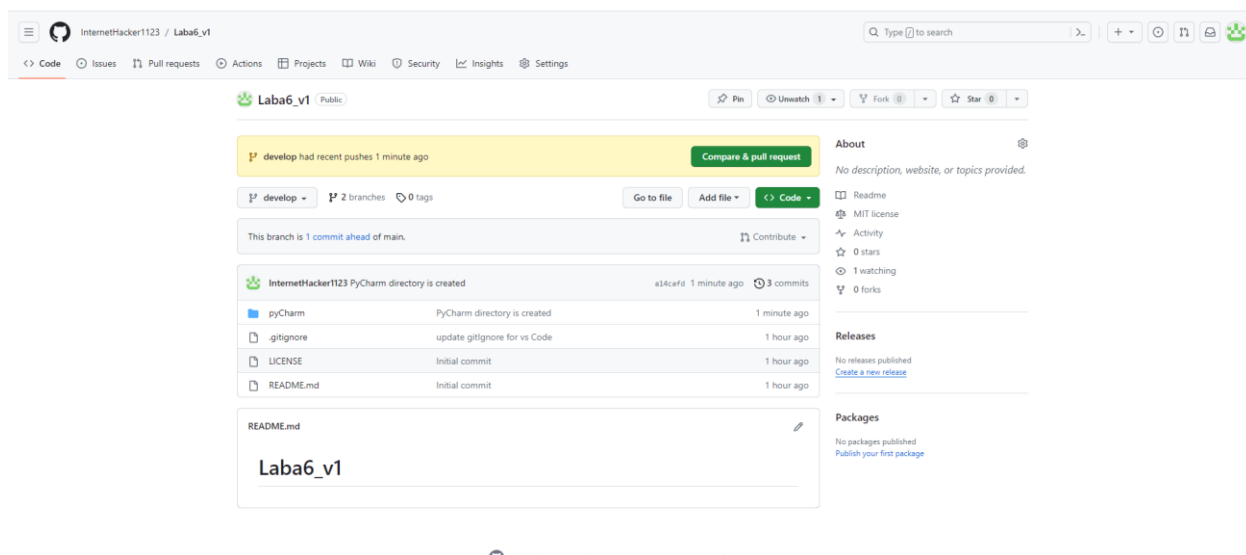
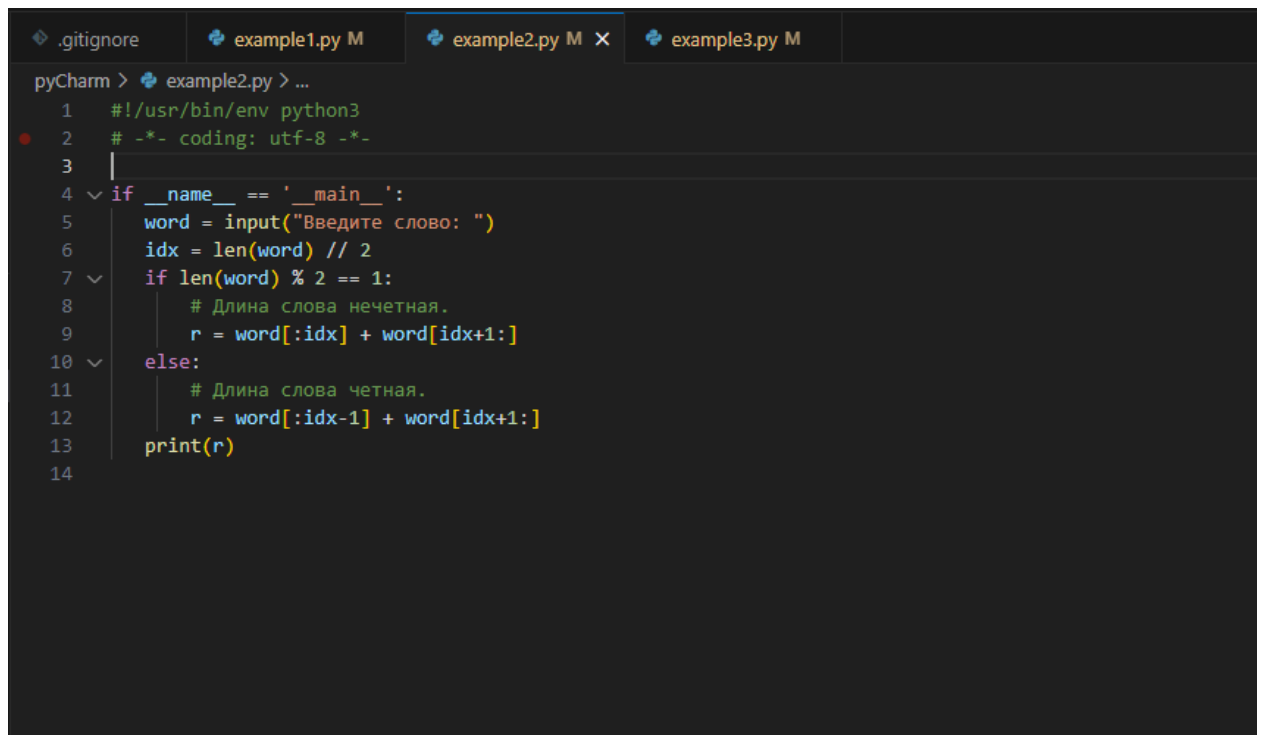


Рисунок 6.1 – Репозиторий с проектом PyCharm

7. Проработал примеры лабораторной работы. Создал для каждого примера отдельный модуль языка Python. Зафиксировала изменения в репозитории.



Рисунок 7.1 – Проработка примера 1



The image shows a PyCharm IDE window with three tabs: `.gitignore`, `example1.py M`, and `example2.py M X`. The active tab is `example2.py M X`. The code in the editor is as follows:

```
pyCharm > example2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      word = input("Введите слово: ")
6      idx = len(word) // 2
7      if len(word) % 2 == 1:
8          # Длина слова нечетная.
9          r = word[idx] + word[idx+1:]
10     else:
11         # Длина слова четная.
12         r = word[:idx-1] + word[idx+1:]
13     print(r)
14
```

Рисунок 7.2 – Проработка примера 2

```
.gitignore example1.py M example2.py M example3.py M X
pyCharm > example3.py > ...

4 import sys
5 if __name__ == '__main__':
6     s = input("Введите предложение: ")
7     n = int(input("Введите длину: "))
8     # Проверить требуемую длину.
9     if len(s) >= n:
10         print(
11             "Заданная длина должна быть больше длины предложения",
12             file=sys.stderr
13         )
14         exit(1)
15     # Разделить предложение на слова.
16     words = s.split(' ')
17     # Проверить количество слов в предложении.
18     if len(words) < 2:
19         print(
20             "Предложение должно содержать несколько слов",
21             file=sys.stderr
22         )
23         exit(1)
24     # Количество пробелов для добавления.
25     delta = n
26     for word in words:
27         delta -= len(word)
28     # Количество пробелов на каждое слово.
29     w, r = delta // (len(words) - 1), delta % (len(words) - 1)
30     # Сформировать список для хранения слов и пробелов.
31     lst = []
32     # Пронумеровать все слова в списке и перебрать их.
33     for i, word in enumerate(words):
34         lst.append(word)
35         # Если слово не является последним, добавить пробелы.
36         if i < len(words) - 1:
37             # Определить количество пробелов.
38             width = w
39             if r > 0:
40                 width += 1
41                 r -= 1
42             # Добавить заданное количество пробелов в список.
43             if width > 0:
44                 lst.append(' ' * width)
45     # Вывести новое предложение, объединив все элементы списка lst.
46     print(''.join(lst))
```

Рисунок 7.3 – Проработка примера 3

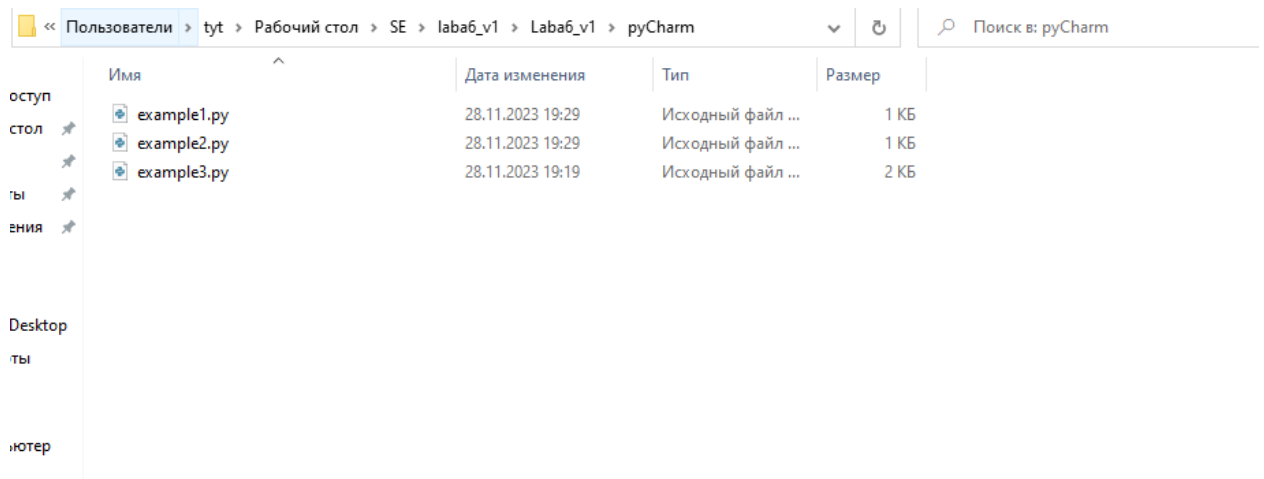


Рисунок 7.4 – Создание отдельных модулей для каждого из примеров

```
C:\Users\tyt\Desktop\SE\laba6_v1\Laba6_v1>git add .
C:\Users\tyt\Desktop\SE\laba6_v1\Laba6_v1>git commit -m"add examples"
[develop f7cf042] add examples
3 files changed, 66 insertions(+)
C:\Users\tyt\Desktop\SE\laba6_v1\Laba6_v1>git push
Enumerating objects: 11, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 6 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.51 KiB | 1.51 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/InternetHacker1123/Laba6_v1.git
a14cafd..f7cf042 develop -> develop
C:\Users\tyt\Desktop\SE\laba6_v1\Laba6_v1>
```

Рисунок 7.5 – Фиксирование изменений в репозитории

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

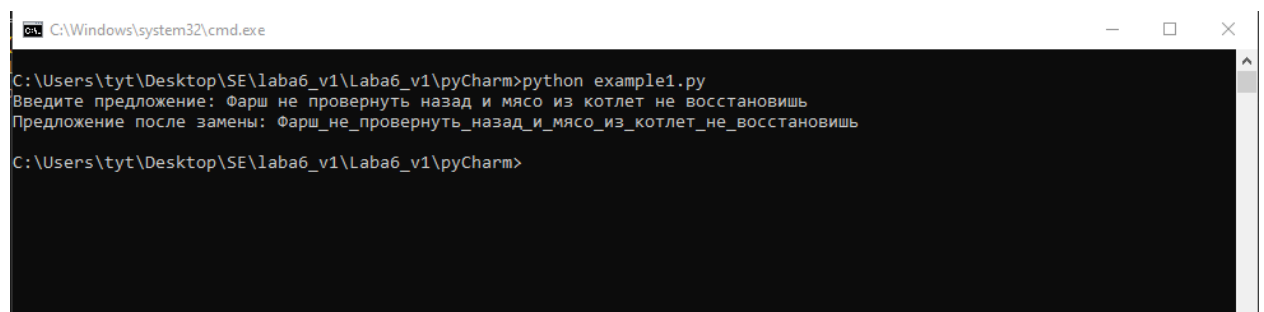


Рисунок 8.1 – Результат примера 1


```
C:\Users\tyt\Desktop\SE\laba6_v1\Laba6_v1\pyCharm>python example2.py
Введите слово: пудж
пж
```

Рисунок 8.2 – Результат примера 2

```
C:\Users\tyt\Desktop\SE\laba6_v1\Laba6_v1\pyCharm>python example3.py
Введите предложение: Чин-чопа чин-чопа
Введите длину: 25
Чин-чопа          чин-чопа

C:\Users\tyt\Desktop\SE\laba6_v1\Laba6_v1\pyCharm>
```

Рисунок 8.3 – Результат примера 3

9. Привел в отчете скриншоты работы программ решения индивидуальных заданий

```
pyCharm > exercise1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      symbol = input("Введите символ: ")
6      sentence = input("Введите предложение: ")
7
8      for i, symb in enumerate(sentence):
9          if symb == symbol:
10             print(symb)
```

Рисунок 9.1 – Код программы Task1.py

```

pyCharm > exercise2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      flag = False
6      sentence = input("Введите предложение: ")
7      sentence = " " + sentence.lower()
8      for i, _ in enumerate(sentence):
9          if sentence[i - 1] == sentence[i]:
10             print(f"Порядковый номер первого символа: {i - 1}\nПорядковый номер второго символа: {i}")
11             flag = True
12             break
13     if flag == False:
14         print("Одинаковых соседних символов нет")
15

```

Рисунок 9.2 – Код программы Task2.py

```

pyCharm > exercise3.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      sentence = input("Введите предложение: ")
6      sp = list(sentence)
7      for i, symb in enumerate(sp):
8          if i % 2 != 0 and symb == " ":
9              del sp[i]
10     sentence = ''.join(sp)
11     print(sentence)

```

Рисунок 9.3 – Код программы MYTask3.py

10. Зафиксировал сделанные изменения в репозитории

```

C:\Users\tyt\Desktop\SE\laba6_v1\Laba6_v1\pyCharm>git push
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 6 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 1.50 KiB | 1.50 MiB/s, done.
Total 8 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/InternetHacker1123/Laba6_v1.git
   b30e6aa..9b0f6bf  main -> main
C:\Users\tyt\Desktop\SE\laba6_v1\Laba6_v1\pyCharm>

```

Рисунок 10.1 – Коммит файлов в репозиторий git

Контрольные вопросы

1. Что такое строки в языке Python?

Строки в Python - упорядоченные последовательности символов, используемые для хранения и представления текстовой информации, поэтому с помощью строк можно работать со всем, что может быть представлено в текстовой форме.

2. Какие существуют способы задания строковых литералов в языке Python?

Строки в апострофах и в кавычках, экранированные последовательности – служебные символы, строки в тройных апострофах или кавычках.

3. Какие операции и функции существуют для строк?

+ — оператор конкатенации строк. Он возвращает строку, состоящую из других строк, как показано здесь. * — оператор создает несколько копий строки. Если s это строка, а n целое число, любое из следующих выражений возвращает строку, состоящую из n объединенных копий s Python предоставляет множество функций, которые встроены в интерпретатор: chr(), ord(), len(), str().

4. Как осуществляется индексирование строк?

Индексация строк начинается с нуля: у первого символа индекс 0, следующего 1 и так далее. Индекс последнего символа в python — “длина строки минус один”.

5. Как осуществляется работа со срезами для строк?

Python также допускает возможность извлечения подстроки из строки, известную как «string slice». Если s это строка, выражение формы s[m:n]

возвращает часть `s` , начинающуюся с позиции `m` , и до позиции `n` , но не включая позицию.

Если пропустить первый индекс, срез начинается с начала строки. Таким образом, `s[:m] = s[0:m]`.

Аналогично, если опустить второй индекс `s[n:]`, срез длится от первого индекса до конца строки. Это хорошая, лаконичная альтернатива более громоздкой `s[n:len(s)]`.

Для любой строки `s` и любого целого `n` числа ($0 \leq n \leq \text{len}(s)$), `s[:n] + s[n:]` будет `s`.

Пропуск обоих индексов возвращает исходную строку. Это не копия, это ссылка на исходную строку.

Отрицательные индексы можно использовать и со срезами.

6. Почему строки Python относятся к неизменяемому типу данных?

Строковую переменную нельзя изменить с помощью операторов, функций и методов.

7. Как проверить то, что каждое слово в строке начинается с заглавной буквы?

`string.istitle()`

8. Как проверить строку на вхождение в неё другой строки?

С помощью оператора `in`.

9. Как найти индекс первого вхождения подстроки в строку?

`string.find()`

10. Как подсчитать количество символов в строке?

С помощью функции `len()`.

11. Как подсчитать то, сколько раз определённый символ встречается в строке?

`string.count()`

12. Что такое f-строки и как ими пользоваться?

Одной простой особенностью f-строк, которые вы можете начать использовать сразу, является интерполяция переменной. Вы можете указать имя переменной непосредственно в f-строковом литерале (f'string'), и python заменит имя соответствующим значением.

13. Как найти подстроку в заданной части строки?

Метод `index()` можно вызывать, передавая ему необязательные аргументы, представляющие индекс начального и конечного фрагмента строки, в пределах которых и нужно осуществлять поиск подстроки.

14. Как вставить содержимое переменной в строку, воспользовавшись методом `format()`?

Переменную нужно указать внутри `{}`.

15. Как узнать о том, что в строке содержатся только цифры?

`s.isdigit()`

16. Как разделить строку по заданному символу?

Метод `split()` разбивает строку на список подстрок с использованием указанного разделителя и возвращает этот список.

17. Как проверить строку на то, что она составлена только из строчных букв?

`s.isupper()`

18. Как проверить то, что строка начинается со строчной буквы?

`str.islower()`. Этот метод возвращает `True`, если первый символ строки является строчной буквой, и `False` в противном случае.

19. Можно ли в Python прибавить целое число к строке?

При попытке выполнения подобной операции будет выдана ошибка `TypeError`.

20. Как «перевернуть» строку?

С помощью среза [: : -1]

21. Как объединить список строк в одну строку, элементы которой разделены дефисами?

Используя метод строки join().

22. Как привести всю строку к верхнему или нижнему регистру?

Использовать методы строк upper() и lower().

23. Как преобразовать первый и последний символы строки к верхнему регистру?

Использовать методы строк upper() и lower() в сочетании с конкатенацией строк

24. Как проверить строку на то, что она составлена только из прописных букв?

s.islower()

25. В какой ситуации вы воспользовались бы методом splitlines()?

Когда надо разделить строку на список строк, используя символы новой строки в качестве разделителя.

26. Как в заданной строке заменить на что-либо все вхождения некоей подстроки?

Для замены всех вхождений определенной подстроки в заданной строке в Python, вы можете использовать метод строки replace().

27. Как проверить то, что строка начинается с заданной последовательности символов, или заканчивается заданной последовательностью символов?

startswith() и endswith()

28. Как узнать о том, что строка включает в себя только пробелы?

s.isspace()

29. Что случится, если умножить некую строку на 3?

Будет создана новая строка, представляющая собой исходную строку, повторённую три раза.

30. Как привести к верхнему регистру первый символ каждого слова в строке?

Существует метод `title()`, приводящий к верхнему регистру первую букву каждого слова в строке.

31. Как пользоваться методом `partition()`?

Метод `partition()` разбивает строку по заданной подстроке. После этого результат возвращается в виде кортежа. При этом подстрока, по которой осуществлялась разбивка, тоже входит в кортеж.

32. В каких ситуациях пользуются методом `rfind()`?

Метод `rfind()` похож на метод `find()`, но он, в отличие от `find()`, просматривает строку не слева направо, а справа налево, возвращая индекс первого найденного вхождения искомой подстроки.