

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»**

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

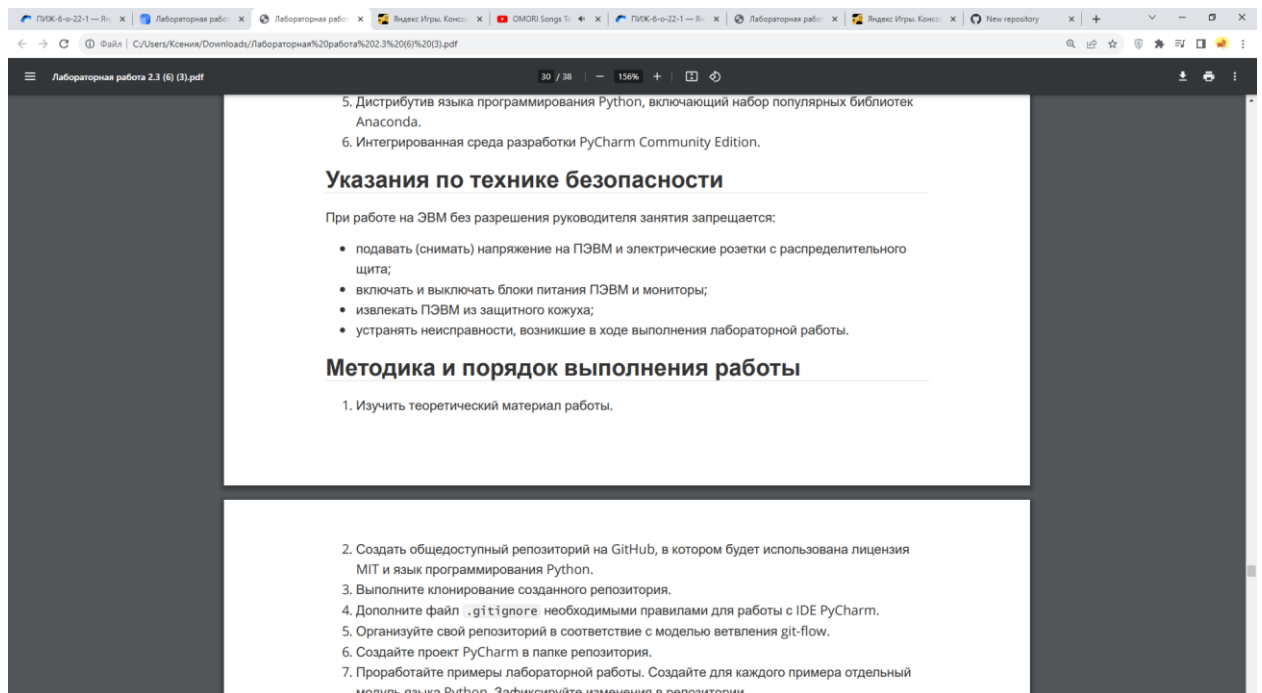


Рисунок 1.1 – Изучение материала для лабораторной работы

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

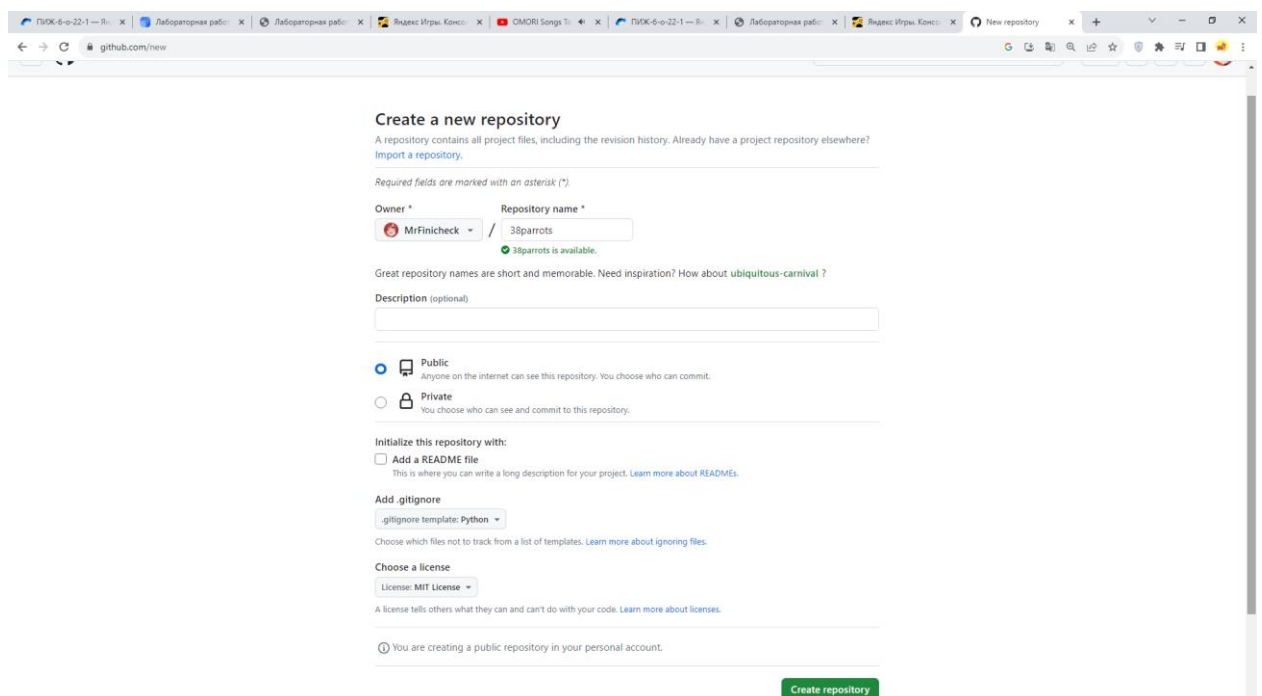


Рисунок 2.1 – Настройка репозитория

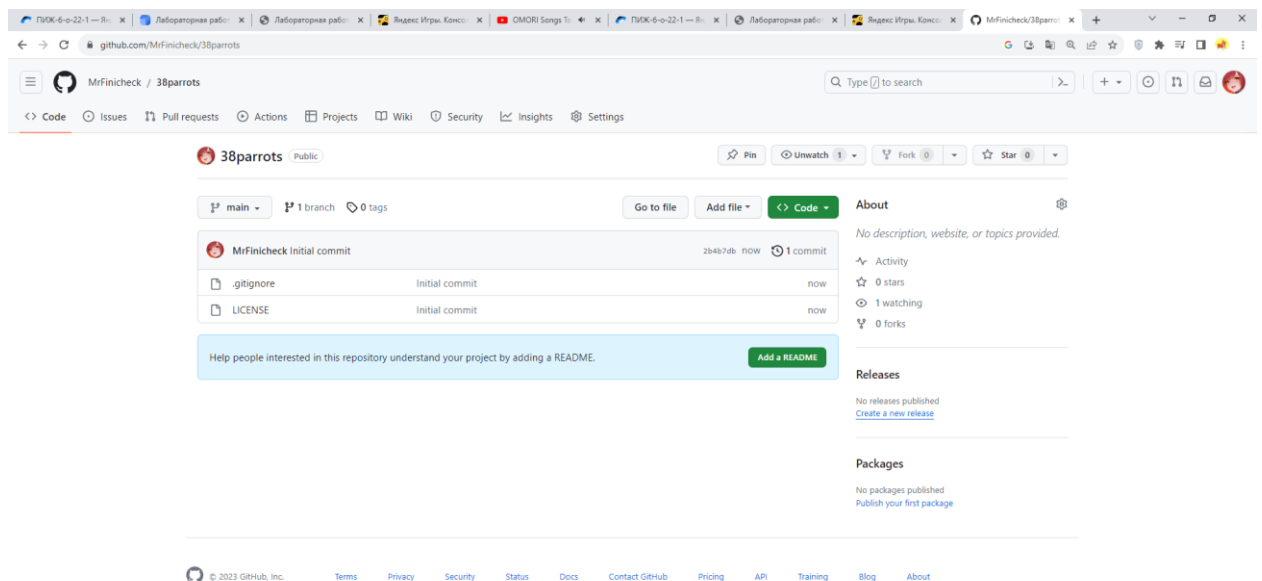


Рисунок 2.2 – Готовый репозиторий

### 3. Выполняю клонирование созданного репозитория

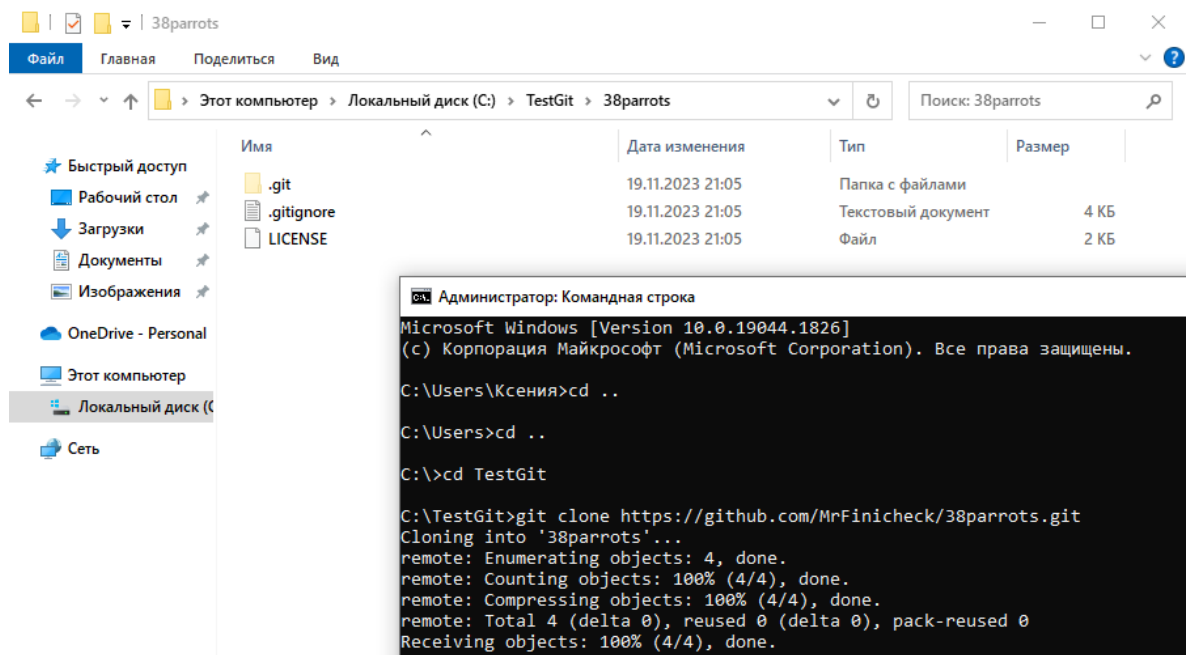
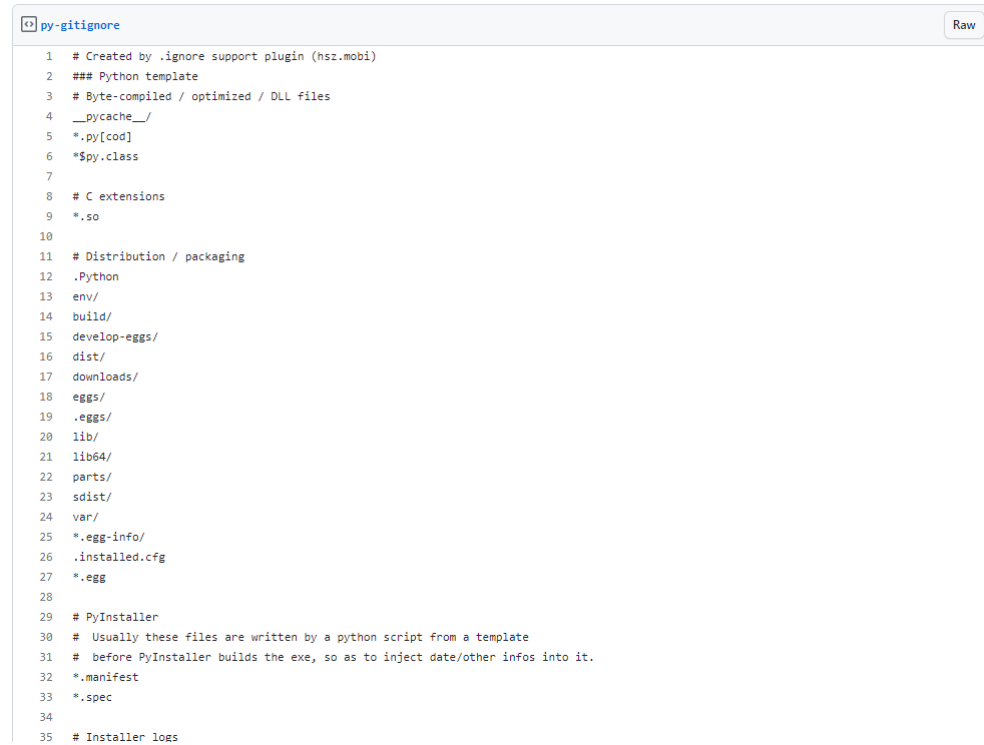


Рисунок 3.1 – Клонирование репозитория на локальный диск

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

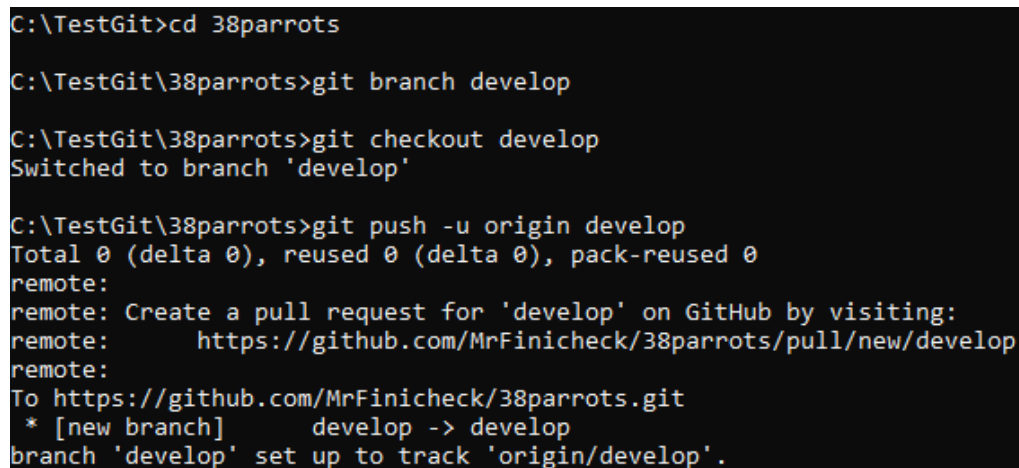
python pycharm gitignore

A screenshot of a code editor window titled 'py-gitignore'. The editor shows a .gitignore file with 35 lines of text. The text includes comments and patterns for ignoring files and directories. The patterns include: \_\_pycache\_\_/, \*.py[co], \*.py.class, \*.so, .Python, env/, build/, develop-eggs/, dist/, downloads/, eggs/, .eggs/, lib/, lib64/, parts/, sdist/, var/, \*.egg-info/, .installed.cfg, \*.egg, \*.manifest, \*.spec, and \*.log. The editor has a 'Raw' button in the top right corner.

```
1 # Created by .ignore support plugin (hsz.mobi)
2 ### Python template
3 # Byte-compiled / optimized / DLL files
4 __pycache__/
5 *.py[co]
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

A terminal window showing a series of git commands and their output. The commands are: 'cd 38parrots', 'git branch develop', 'git checkout develop', and 'git push -u origin develop'. The output shows the branch being created, switched to, and pushed to the origin. It also includes a message from GitHub about creating a pull request.

```
C:\TestGit>cd 38parrots
C:\TestGit\38parrots>git branch develop
C:\TestGit\38parrots>git checkout develop
Switched to branch 'develop'
C:\TestGit\38parrots>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/38parrots/pull/new/develop
remote:
To https://github.com/MrFinicheck/38parrots.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.
```

Рисунок 5.1 – Создание ветки develop от ветки main

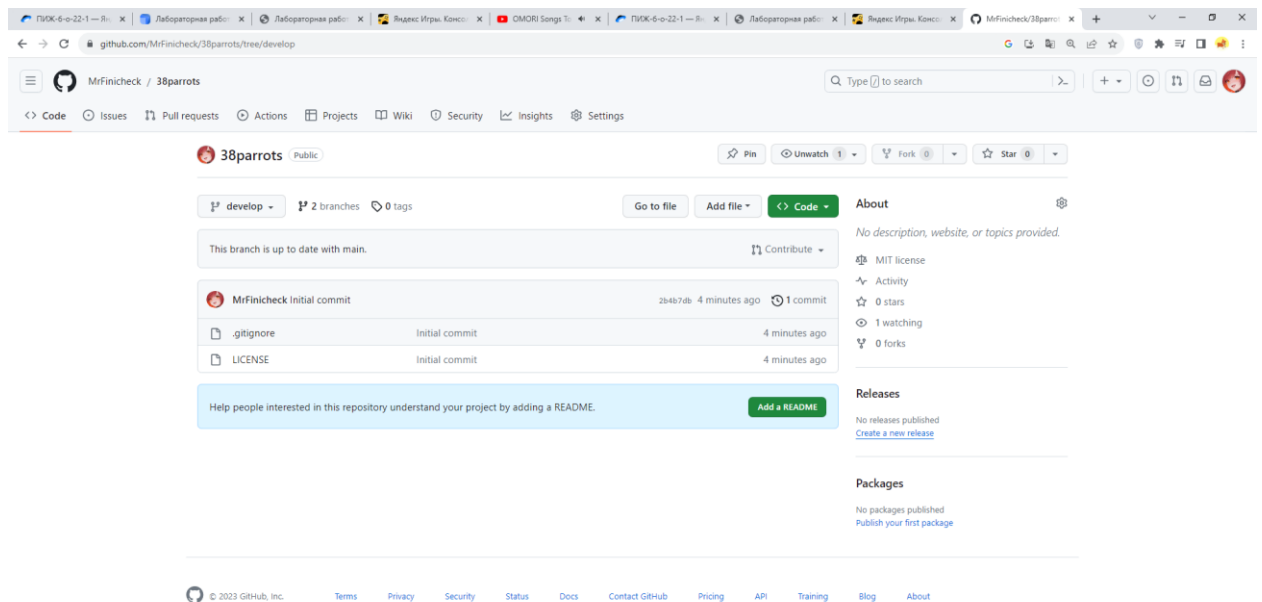


Рисунок 5.2 – Ветка develop на GitHub

## 6. Создала проект PyCharm в папке репозитория

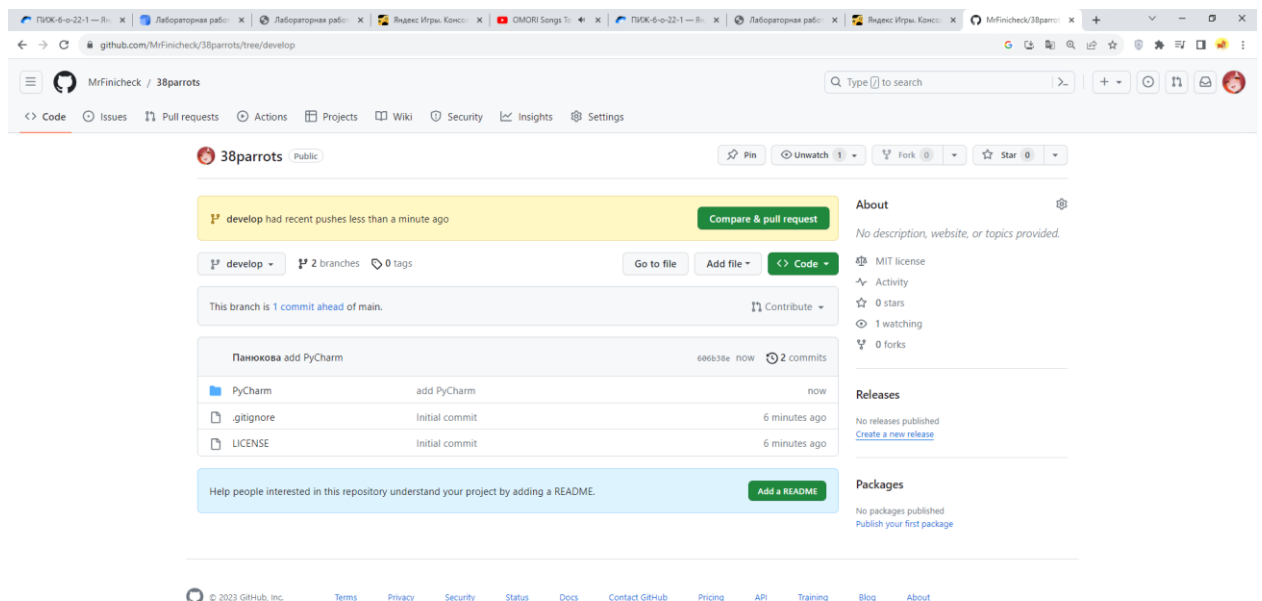


Рисунок 6.1 – Репозиторий с проектом PyCharm

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

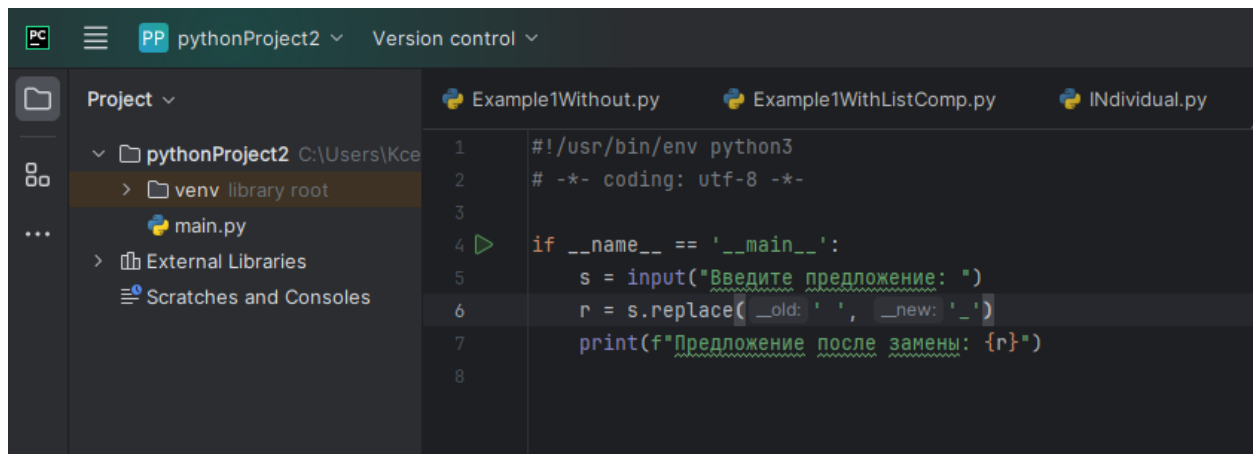


Рисунок 7.1 – Проработка примера 1

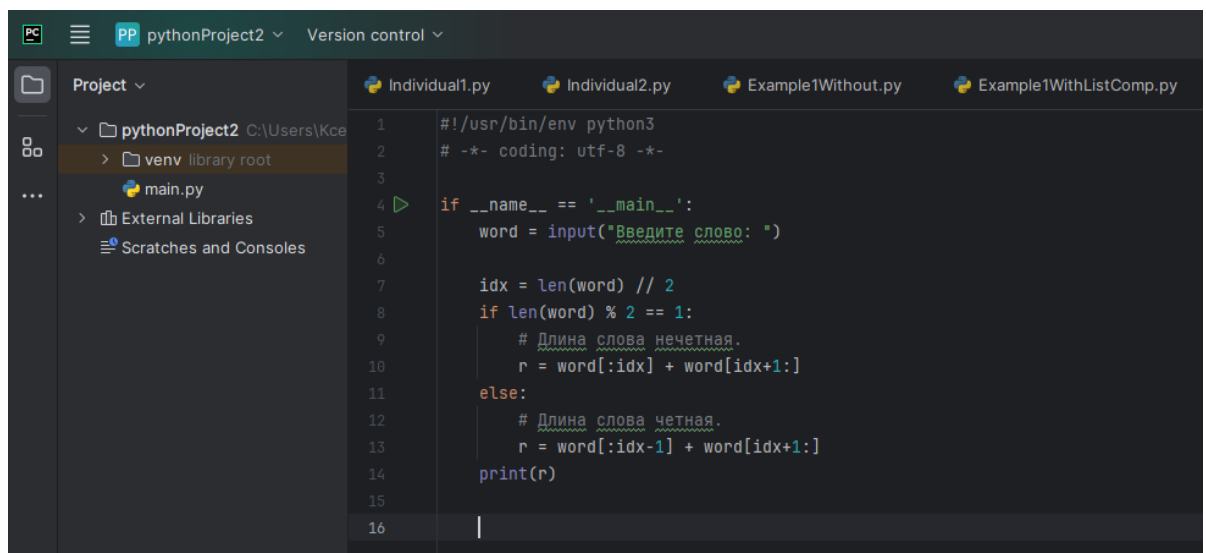


Рисунок 7.2 – Проработка примера 2

```
Example1Without.py Example1WithListComp.py INdividual.py Example111.py Example222.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == '__main__':
8      s = input("Введите предложение: ")
9      n = int(input("Введите длину: "))
10
11     # Проверить требуемую длину.
12     if len(s) >= n:
13         print(
14             "Заданная длина должна быть не больше длины предложения",
15             file=sys.stderr
16         )
17         exit(1)
18
19     # Разделить предложение на слова.
20     words = s.split(' ')
21     # Проверить количество слов в предложении.
22     if len(words) < 2:
23         print(
24             "Предложение должно содержать несколько слов",
25             file=sys.stderr
26         )
27         exit(1)
28
29     # Количество пробелов для добавления.
30     delta = n
31     for word in words:
32         delta -= len(word)
33
34     # Количество пробелов на каждое слово.
35     w, r = delta // (len(words) - 1), delta % (len(words) - 1)
36
37     # Сформировать список для хранения слов и пробелов.
38     lst = []
39
40     # Пронумеровать все слова в списке и перебрать их.
41     for i, word in enumerate(words):
42         lst.append(word)
43
44 if __name__ == '__main__':
45     > Examples > Example333.py
```

Рисунок 7.3 – Проработка примера 3

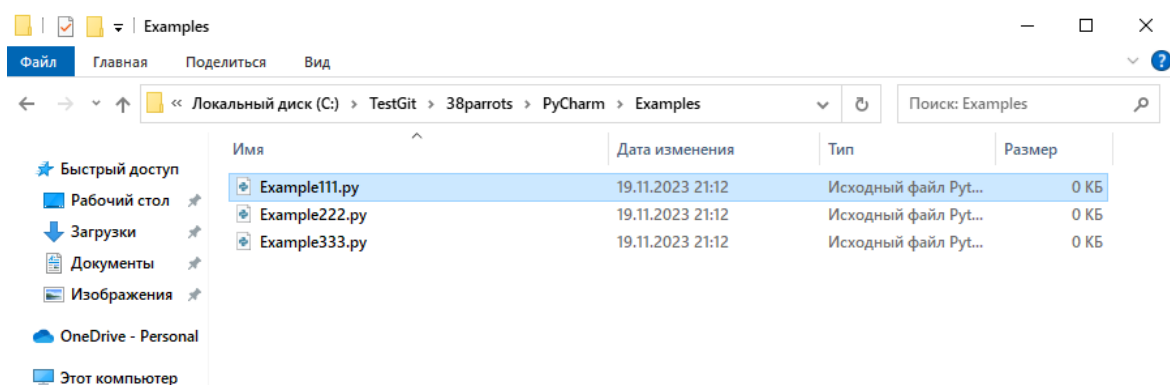


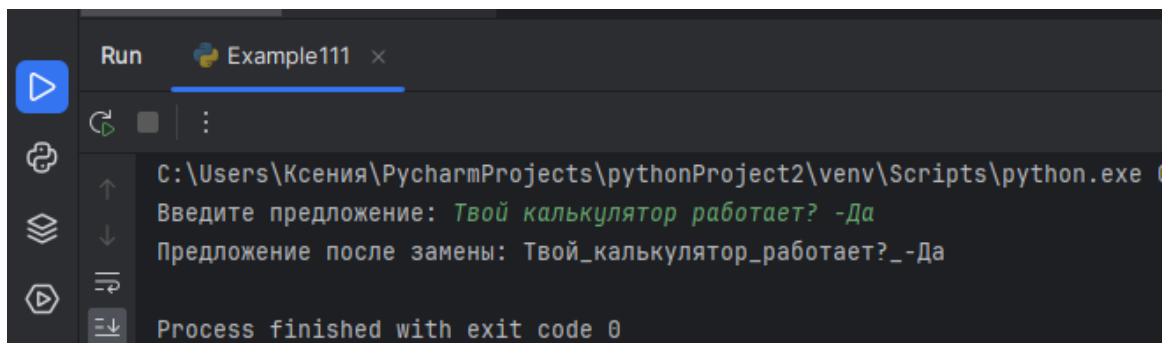
Рисунок 7.4 – Создание отдельных модулей для каждого из примеров

```
Администратор: Командная строка
Microsoft Windows [Version 10.0.19044.1826]
(с) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Ксения>cd ..
C:\Users>cd ..
C:\>cd TestGit
C:\TestGit>cd 38parrots
C:\TestGit\38parrots>git add PyCharm
C:\TestGit\38parrots>git commit -m"adding examples"
[develop 6802fd6] adding examples
4 files changed, 82 insertions(+)
create mode 100644 PyCharm/Examples/Example111.py
create mode 100644 PyCharm/Examples/Example222.py
create mode 100644 PyCharm/Examples/Example333.py
delete mode 100644 "PyCharm/\320\237\320\276\320\274\320\275\320\270.txt"
```

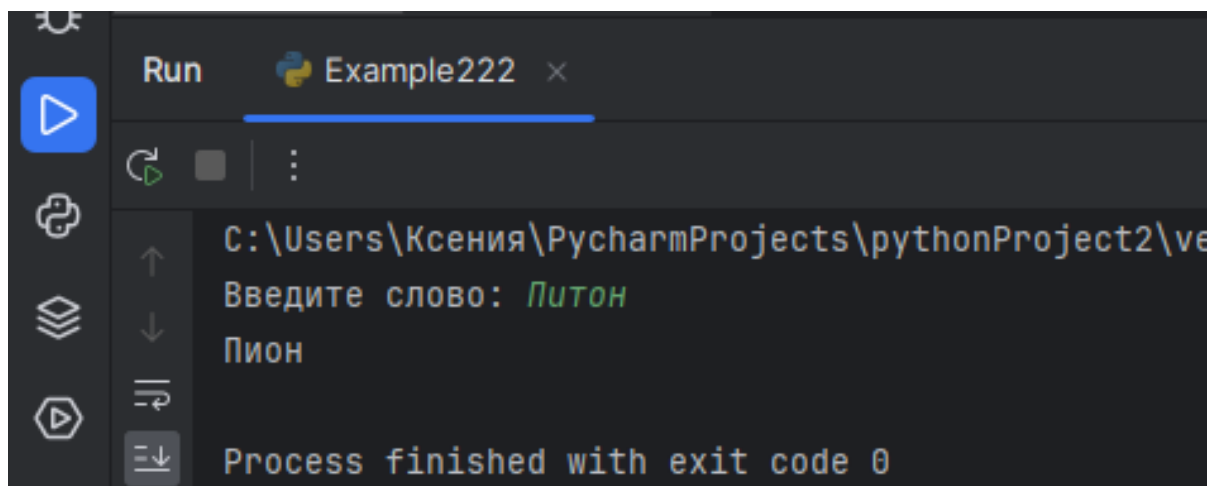
Рисунок 7.5 – Фиксирование изменений в репозитории

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



```
Run Example111 x
C:\Users\Ксения\PycharmProjects\pythonProject2\venv\Scripts\python.exe
Введите предложение: Твой калькулятор работает? -Да
Предложение после замены: Твой_калькулятор_работает?_-Да
Process finished with exit code 0
```

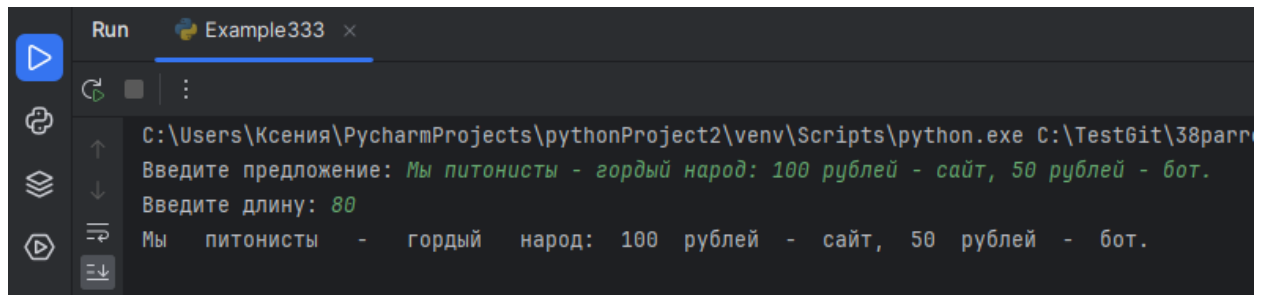
Рисунок 8.1 – Результат примера 1 с применением map()



```
Run Example222 x
C:\Users\Ксения\PycharmProjects\pythonProject2\venv\Scripts\python.exe
Введите слово: Питон
Пион
Process finished with exit code 0
```

Рисунок 8.2 – Результат примера 2 с применением списковых включений

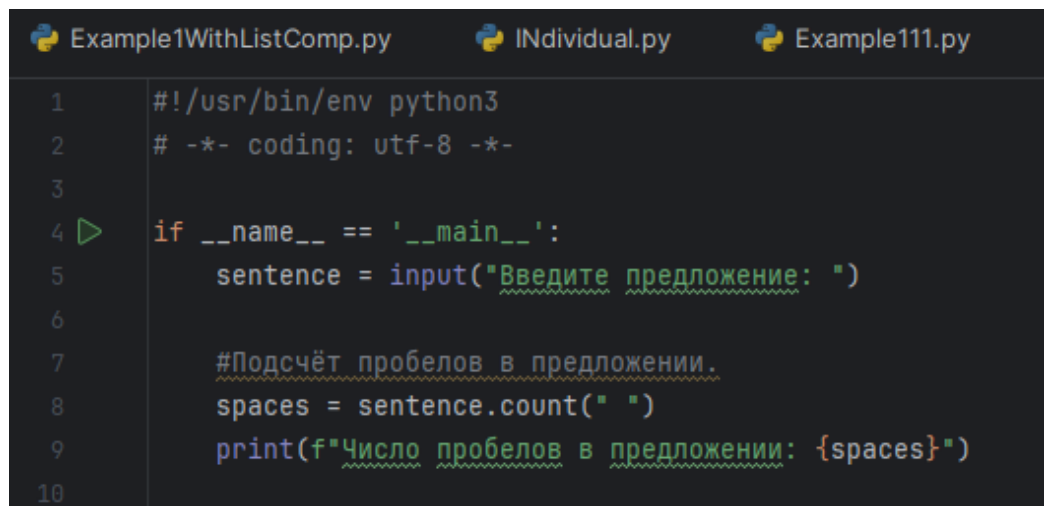




```
Run Example333 x
C:\Users\Ксения\PycharmProjects\pythonProject2\venv\Scripts\python.exe C:\Test6it\38par...
Введите предложение: Мы питонисты - гордый народ: 100 рублей - сайт, 50 рублей - бот.
Введите длину: 80
Мы питонисты - гордый народ: 100 рублей - сайт, 50 рублей - бот.
```

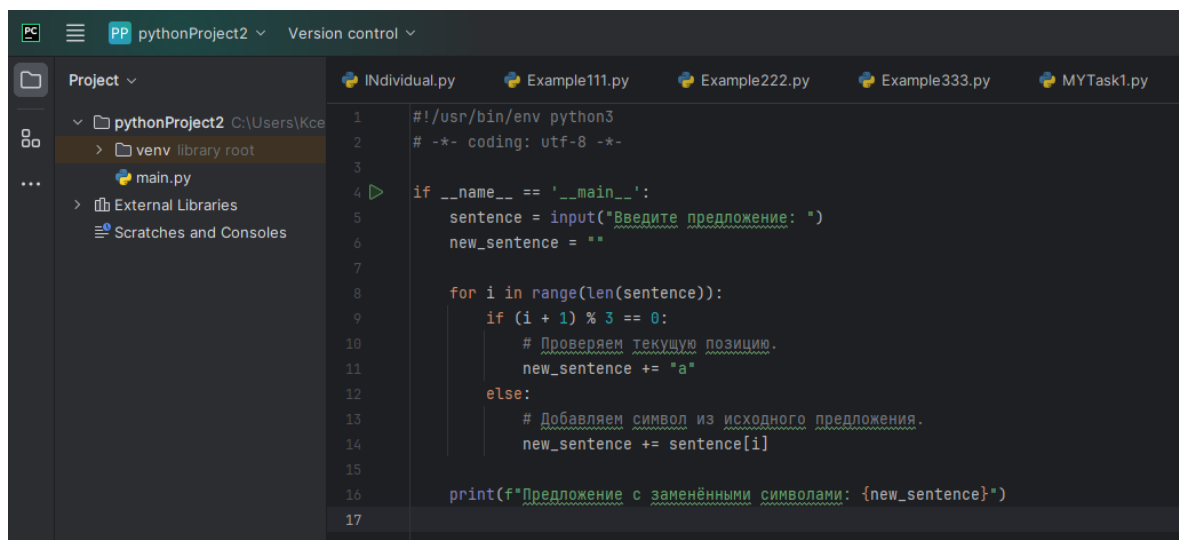
Рисунок 8.3 – Результат примера 3 с применением списковых включений

9. Привела в отчете скриншоты работы программ решения индивидуальных заданий



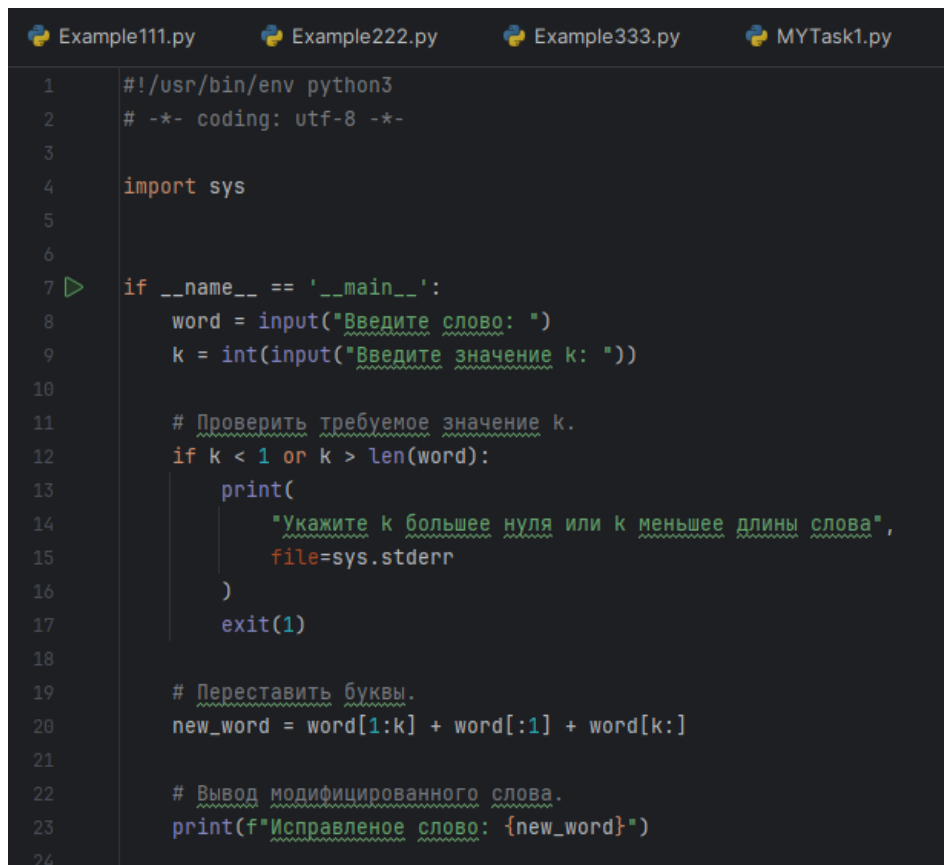
```
Example1WithListComp.py
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     sentence = input("Введите предложение: ")
6
7     #Подсчёт пробелов в предложении.
8     spaces = sentence.count(" ")
9     print(f"Число пробелов в предложении: {spaces}")
10
```

Рисунок 9.1 – Код программы MYTask1.py в IDE PyCharm



```
pythonProject2 Version control
Project
  pythonProject2 C:\Users\Kce
    venv library root
    main.py
  External Libraries
  Scratches and Consoles
  INdividual.py Example111.py Example222.py Example333.py MYTask1.py
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     sentence = input("Введите предложение: ")
6     new_sentence = ""
7
8     for i in range(len(sentence)):
9         if (i + 1) % 3 == 0:
10             # Проверяем текущую позицию.
11             new_sentence += "a"
12         else:
13             # Добавляем символ из исходного предложения.
14             new_sentence += sentence[i]
15
16     print(f"Предложение с заменёнными символами: {new_sentence}")
17
```

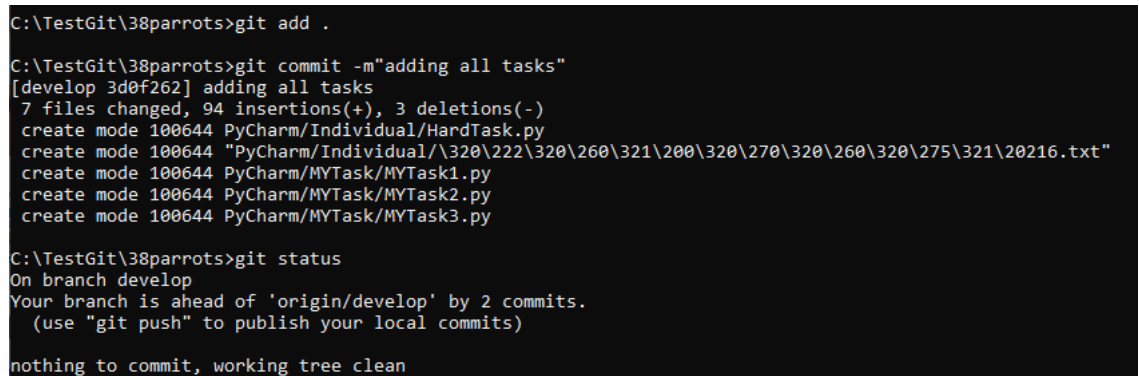
Рисунок 9.2 – Код программы MYTask2.py в IDE PyCharm



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == '__main__':
8      word = input("Введите слово: ")
9      k = int(input("Введите значение k: "))
10
11     # Проверить требуемое значение k.
12     if k < 1 or k > len(word):
13         print(
14             "Укажите k большее нуля или k меньше длины слова",
15             file=sys.stderr
16         )
17         exit(1)
18
19     # Переставить буквы.
20     new_word = word[1:k] + word[:1] + word[k:]
21
22     # Вывод модифицированного слова.
23     print(f"Исправленное слово: {new_word}")
24
```

Рисунок 9.3 – Код программы MYTask3.py в IDE PyCharm

## 10. Зафиксировала сделанные изменения в репозитории



```
C:\TestGit\38parrots>git add .
C:\TestGit\38parrots>git commit -m"adding all tasks"
[develop 3d0f262] adding all tasks
7 files changed, 94 insertions(+), 3 deletions(-)
create mode 100644 PyCharm/Individual/HardTask.py
create mode 100644 "PyCharm/Individual/\320\222\320\260\321\200\320\270\320\260\320\275\321\20216.txt"
create mode 100644 PyCharm/MYTask/MYTask1.py
create mode 100644 PyCharm/MYTask/MYTask2.py
create mode 100644 PyCharm/MYTask/MYTask3.py
C:\TestGit\38parrots>git status
On branch develop
Your branch is ahead of 'origin/develop' by 2 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
```

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.