

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования «СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №2.3

Дисциплина: «Основы кроссплатформенного программирования»

Тема: «Работа со строками в языке Python»

Выполнил: студент 1 курса,
группы ИВТ-б-о-21-1
Богдан Александр Анатольевич

Ставрополь 2022

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

Практическая часть:

Создал репозиторий на GitHub

```
D:\WAR\gi>git clone https://github.com/Alekbs/Lab6
Cloning into 'Lab6'...
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 1. Клонирование репозитория



Рисунок 2. Изменение файла .gitignor

```

D:\WAR\gi\Lab6>git flow init

Which branch should be used for bringing forth production releases?
- develop
- main
Branch name for production releases: [main] develop

Which branch should be used for integration of the "next release"?
- main
Branch name for "next release" development: [] main

How to name your supporting branch prefixes?
Feature branches? [feature/] pre
Bugfix branches? [bugfix/] bug
Release branches? [release/] release
Hotfix branches? [hotfix/] hot
Support branches? [support/] support
Version tag prefix? [] pre
Hooks and filters directory? [D:/WAR/gi/Lab6/.git/hooks] hooks

```

Рисунок 3. Организация репозитория в соответствии с git-flow

The screenshot shows an IDE with a project named 'pri1.py'. The file explorer on the left shows the project structure, including a 'venv' directory and files like '.gitignore', 'pyvenv.cfg', 'main.py', and 'pri1.py'. The main editor displays the code for 'pri1.py':

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     s = input("Введите предложение: ")
6     r = s.replace(' ', '_')
7     print(f"Предложение после замены: {r}")

```

Below the code editor, the 'Run' console shows the execution of the script using the command: `D:\WAR\gi\Lab6\Project\venv\Scripts\python.exe D:/WAR/gi/Lab6/Project/pri1.py`. The output shows the user input 'probka probka' and the resulting output 'Предложение после замены: probka_probka'.

Рисунок 4. Проработка 1 примера

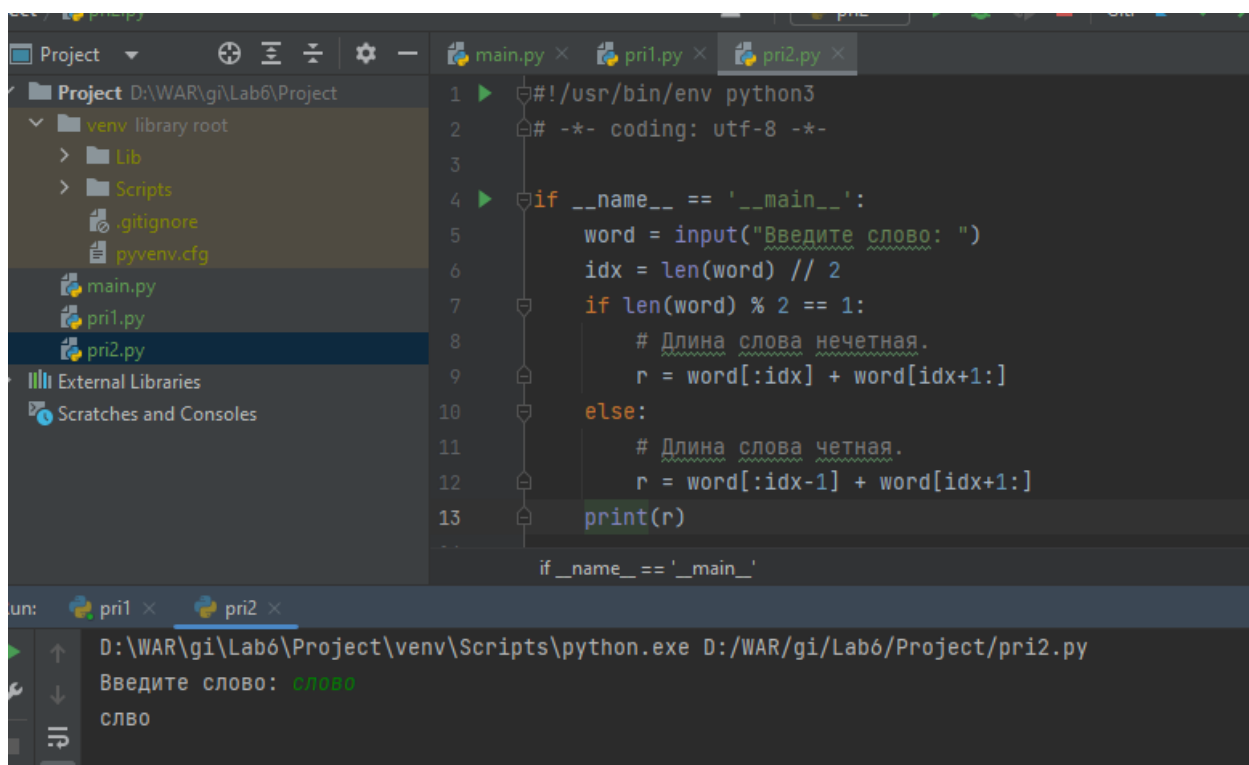


Рисунок 5. Проработка 2 примера

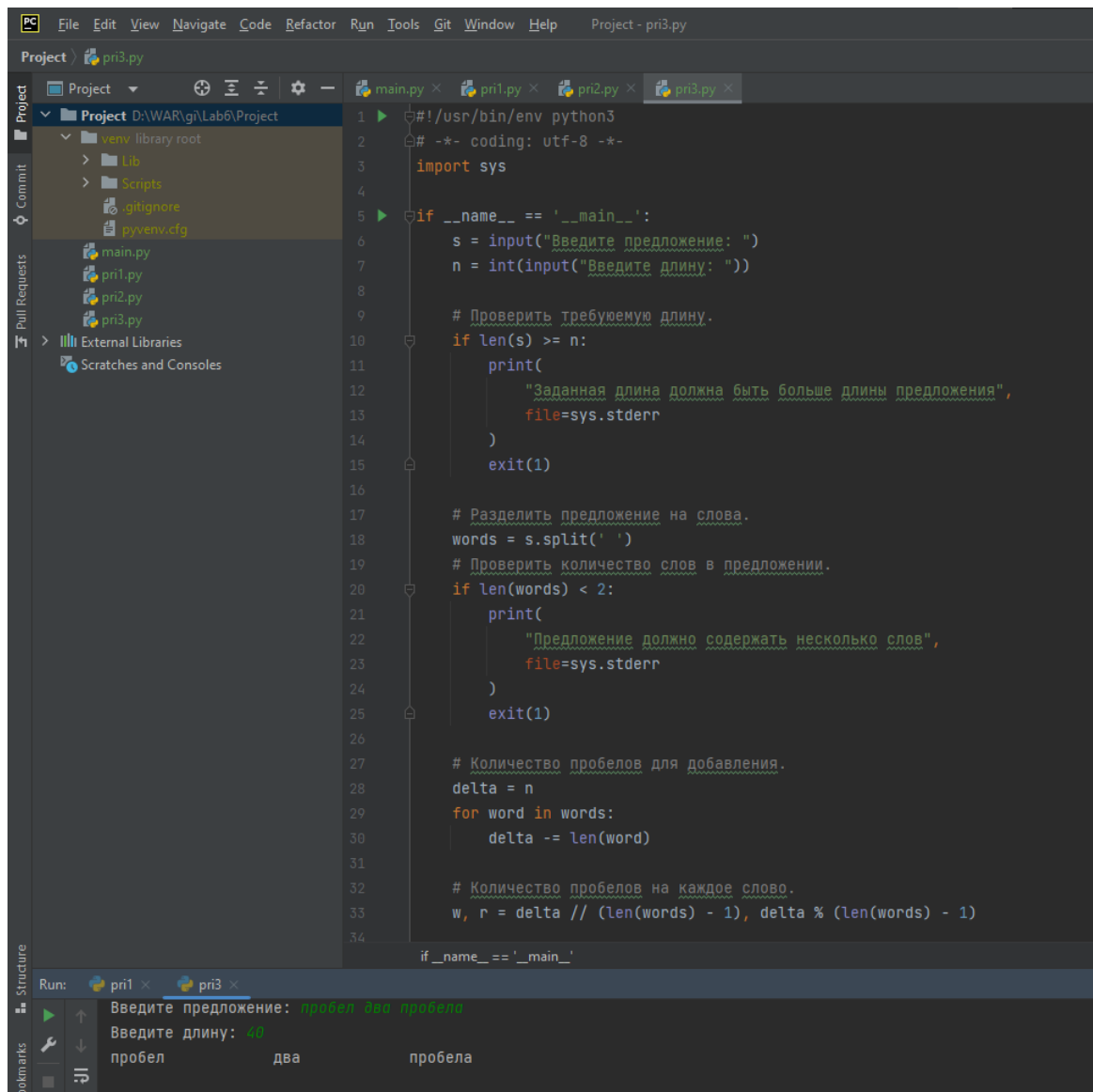


Рисунок 6. Проработка 3 примера

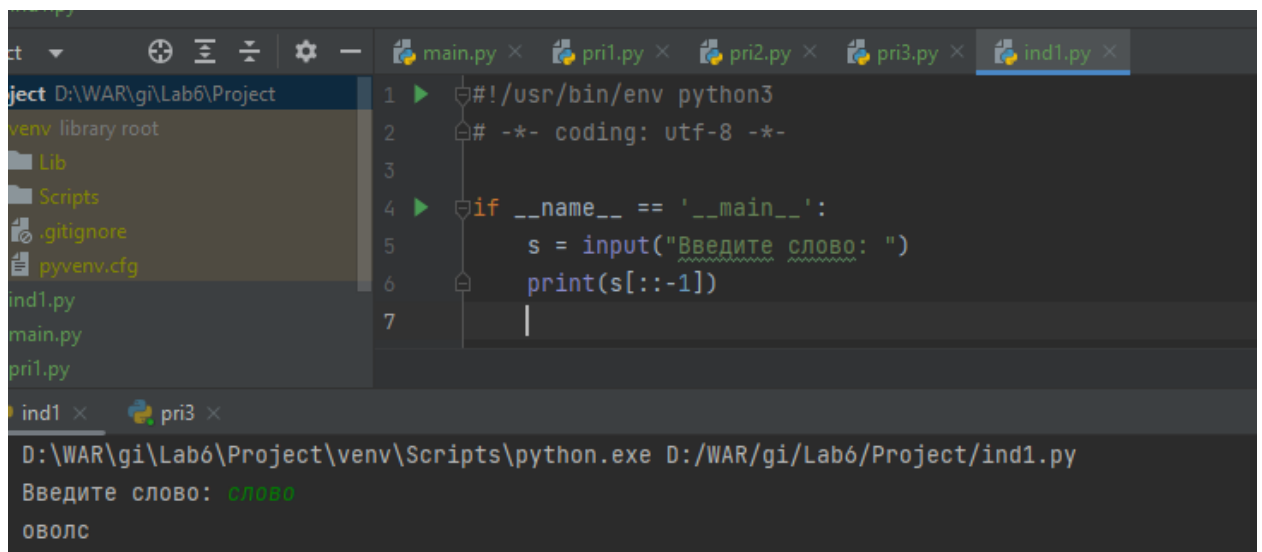
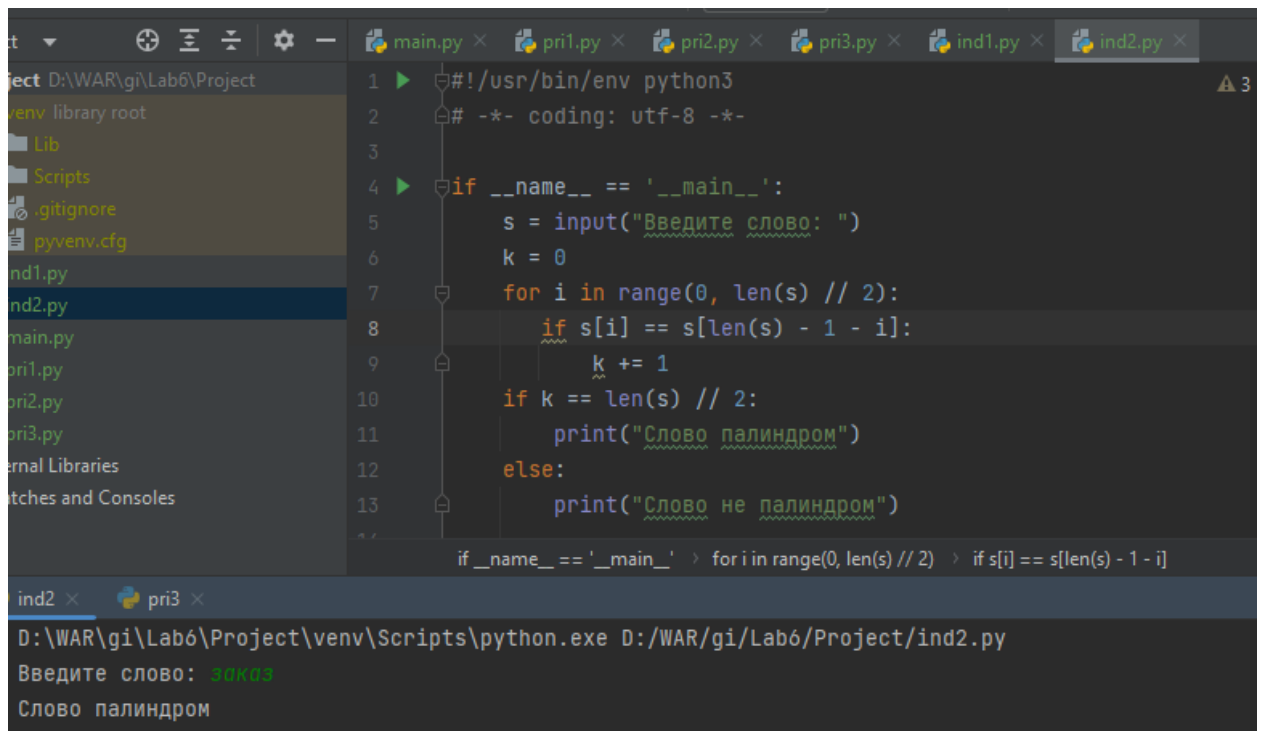


Рисунок 7. Результат работы 1 индивидуального задания

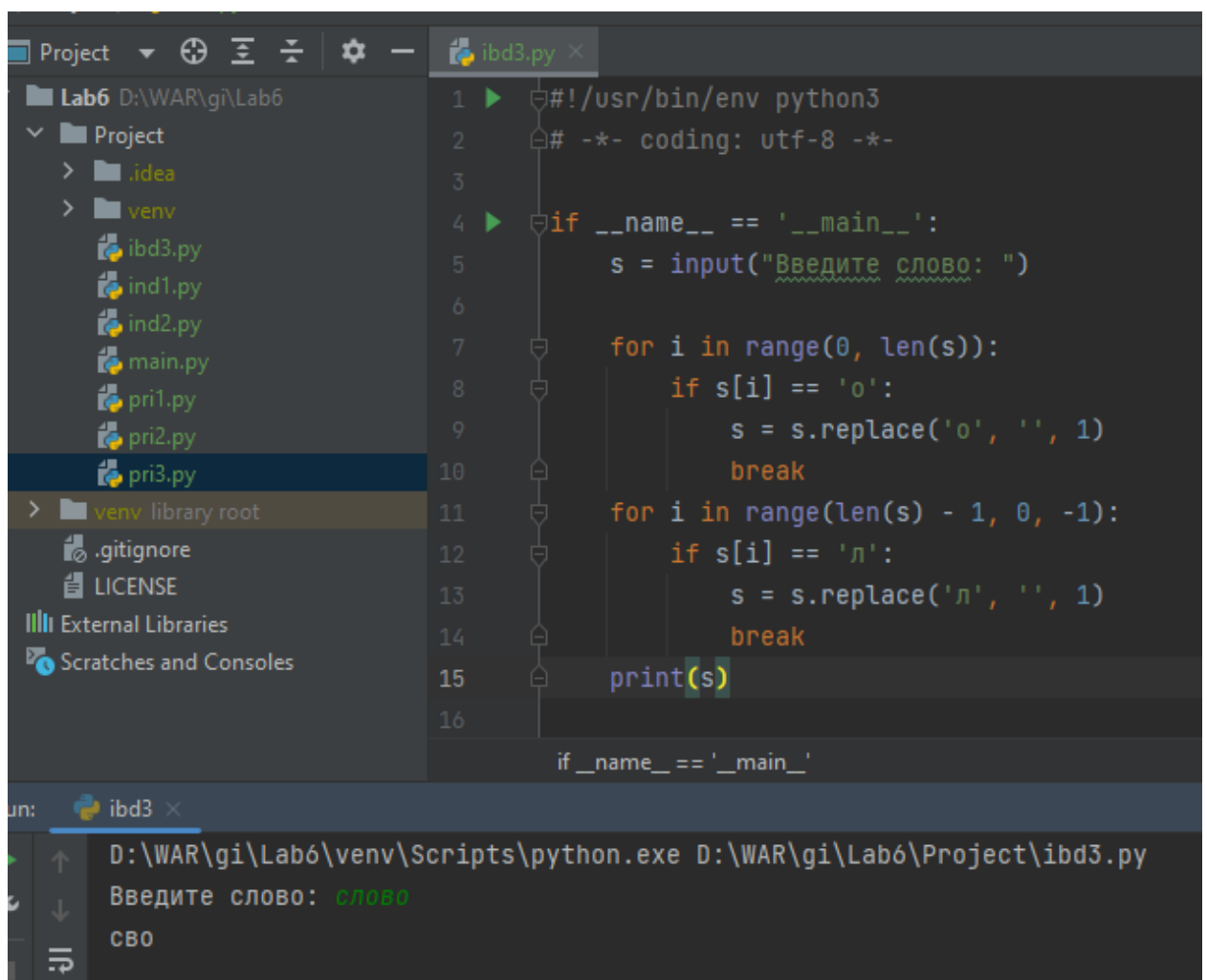


```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     s = input("Введите слово: ")
6     k = 0
7     for i in range(0, len(s) // 2):
8         if s[i] == s[len(s) - 1 - i]:
9             k += 1
10    if k == len(s) // 2:
11        print("Слово палиндром")
12    else:
13        print("Слово не палиндром")
```

if __name__ == '__main__' > for i in range(0, len(s) // 2) > if s[i] == s[len(s) - 1 - i]

D:\WAR\gi\Lab6\Project\venv\Scripts\python.exe D:/WAR/gi/Lab6/Project/ind2.py
Введите слово: закон
Слово палиндром

Рисунок 8. Результат работы 2 индивидуального задания

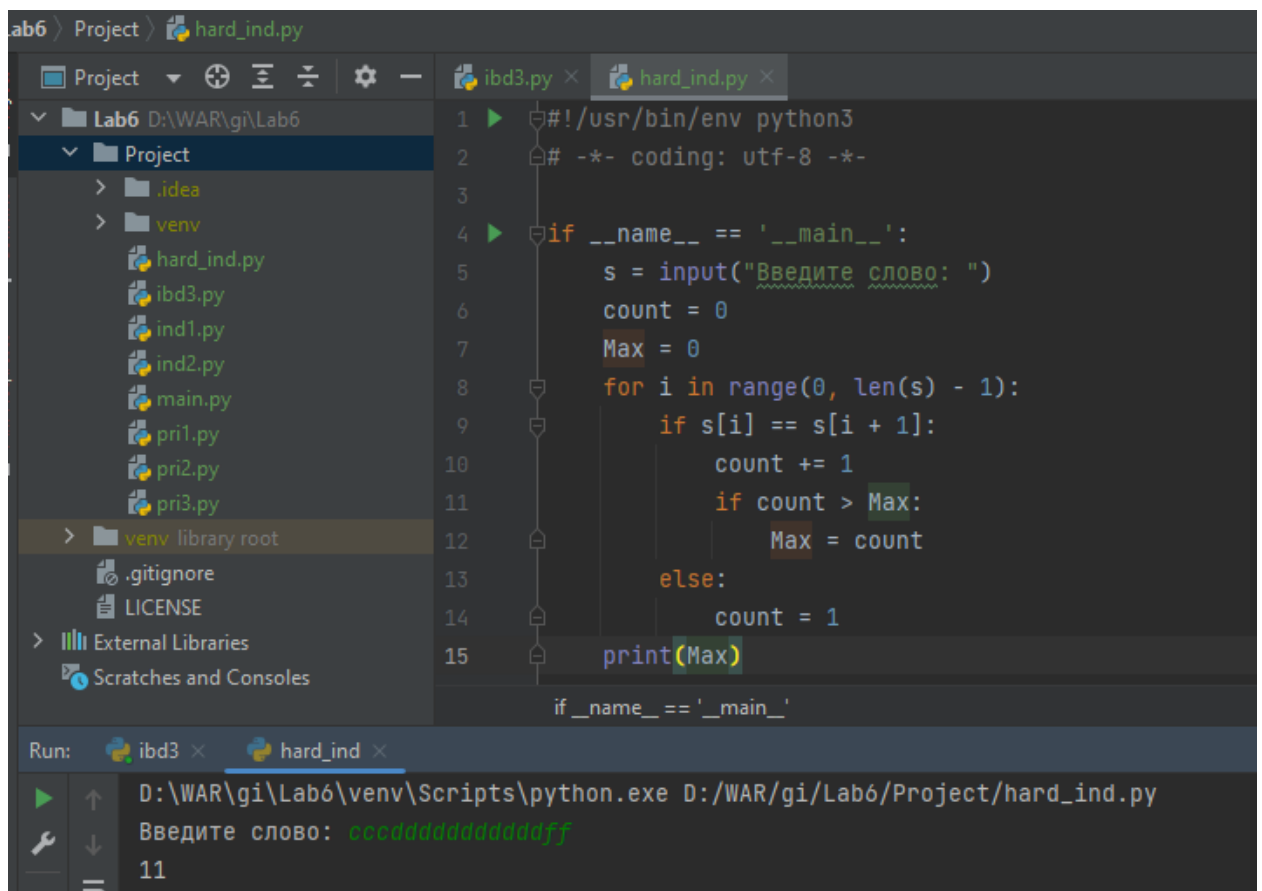


```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     s = input("Введите слово: ")
6
7     for i in range(0, len(s)):
8         if s[i] == 'о':
9             s = s.replace('о', '', 1)
10            break
11
12    for i in range(len(s) - 1, 0, -1):
13        if s[i] == 'л':
14            s = s.replace('л', '', 1)
15            break
16
17    print(s)
```

if __name__ == '__main__'

D:\WAR\gi\Lab6\venv\Scripts\python.exe D:\WAR\gi\Lab6\Project\ibd3.py
Введите слово: слово
сво

Рисунок 9. Результат работы 3 индивидуального задания



The screenshot shows an IDE with a project named 'Lab6'. The file explorer on the left shows a 'Project' folder containing several Python files: 'hard_ind.py', 'ibd3.py', 'ind1.py', 'ind2.py', 'main.py', 'pri1.py', 'pri2.py', and 'pri3.py'. The 'Run' tab at the bottom shows the execution of 'hard_ind.py' using 'python.exe'. The command prompt displays the input 'Введите слово: cccdddddffffff' and the output '11'.

```
1 #!/usr/bin/env python3
2 #- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     s = input("Введите слово: ")
6     count = 0
7     Max = 0
8     for i in range(0, len(s) - 1):
9         if s[i] == s[i + 1]:
10             count += 1
11             if count > Max:
12                 Max = count
13         else:
14             count = 1
15     print(Max)
```

Рисунок 10. Результат работы усложненного задания

```
D:\WAR\gi\Lab6>git add .
D:\WAR\gi\Lab6>git commit -m "prog"
[develop 805b4a8] prog
8 files changed, 142 insertions(+)
create mode 100644 Project/hard_ind.py
create mode 100644 Project/ibd3.py
create mode 100644 Project/ind1.py
create mode 100644 Project/ind2.py
create mode 100644 Project/main.py
create mode 100644 Project/pri1.py
create mode 100644 Project/pri2.py
create mode 100644 Project/pri3.py
D:\WAR\gi\Lab6>git push
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (11/11), 2.64 KiB | 541.00 KiB/s, done.
Total 11 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Alekbs/Lab6
ba40953..805b4a8 develop -> develop
```

Рисунок 11. Сохранение изменений

Ответы на 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, "сырые" строки - подавляют экранирование, строки в тройных апострофах или кавычках.

3. Какие операции и функции существуют для строк?

Оператор сложения (+), умножения (*), принадлежности подстроки (in).

Функции:

chr() - преобразует целое число в символ;

ord() - преобразует символ в целое число;

len() - возвращает длину строки;

str() - изменяет тип объекта на string.

4. Как осуществляется индексирование строк?

В Python строки являются упорядоченными последовательностями символьных данных и могут быть проиндексированы. Доступ к отдельным символам в строке можно получить, указав имя строки, за которым следует число в квадратных скобках [].

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.

Существует еще один вариант синтаксиса среза, о котором стоит упомянуть. Добавление дополнительного `:` и третьего индекса означает шаг, который указывает, сколько символов следует пропустить после извлечения каждого символа в срезе.

6. Почему строки Python относятся к неизменяемому типу данных?

Строки — один из типов данных, которые Python считает неизменяемыми, что означает невозможность их изменять. Как вы ниже увидите, python дает возможность изменять (заменять и перезаписывать) строки.

На самом деле нет особой необходимости изменять строки. Обычно вы можете легко сгенерировать копию исходной строки с необходимыми изменениями. Есть минимум 2 способа сделать это в python.

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

`string.istitle()` определяет, начинаются ли слова строки с заглавной буквы.

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

`string.find()`.

9. Как найти индекс первого вхождения подстроки в строку?

`s.partition(<sep>).`

10. Как подсчитать количество символов в строке?

`len(s).`

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

`s.count(<sub>).`

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

Иногда (а точнее, довольно часто) возникают ситуации, когда нужно сделать строку, подставив в неё некоторые данные, полученные в процессе выполнения программы (пользовательский ввод, данные из файлов и т. д.). Подстановку данных можно сделать с помощью форматирования строк. Форматирование можно сделать с помощью оператора `%`, либо с помощью метода `format`, либо с помощью так называемых f-строк. Форматирование с помощью оператора `%` относится к устаревшим способам форматирования, поэтому в рамках данной лабораторной работы будут рассмотрены только метод `format` и f-строки.

13. Как найти подстроку в заданной части строки?

`s.find(значение, начало, конец).`

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

`print('{}'.format(s)).`

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

`s.isdigit().`

16. Как разделить строку по заданному символу?

`str.split()`.

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

`string.isalpha()`.

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

`string.istitle()` определяет, начинаются ли слова строки с заглавной буквы.

`s.istitle()` возвращает `True` когда `s` не пустая строка и первый алфавитный символ каждого слова в верхнем регистре, а все остальные буквенные символы в каждом слове строчные. Возвращает `False`, если нет.

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

Нет.

20. Как «перевернуть» строку?

`s.reverse()`.

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

`str.split('-')`.

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

`s.upper()`, `s.lower()`.

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

`s.capitalize()`.

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

`s.isupper()`.

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

`splitlines()` делит `s` на строки и возвращает их в списке. Любой из следующих символов или последовательностей символов считается границей строки.

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

`s.replace(old, new)`.

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

`str.startswith()` и `str.endswith()`.

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

`s.isspace()`.

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

Она напечатается 3 раза.

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

`s.title()`.

31. Как пользоваться методом `partition()`?

Разбивает строку при первом появлении строки аргумента и возвращает кортеж, содержащий часть перед разделителем, строку аргумента и часть после разделителя.

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

`s.rfind(₎` возвращает индекс последнего вхождения подстроки `<sub>` в `s`, который соответствует началу `<sub>`.

Вывод: в результате выполнения работы были приобретены навыки по работе со строками при написании программ с помощью языка программирования Python версии 3.x.