

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение
высшего образования**

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра информационных систем и технологий

Отчет по лабораторной работе №6.

Дисциплина: «Основы программной инженерии»

Выполнил:

Студент группы ПИЖ-б-о-22-1,
направление подготовки: 09.03.04

«Программная инженерия»

ФИО: Джараян Арег Александрович

Проверил:

Воронкин Р. А.

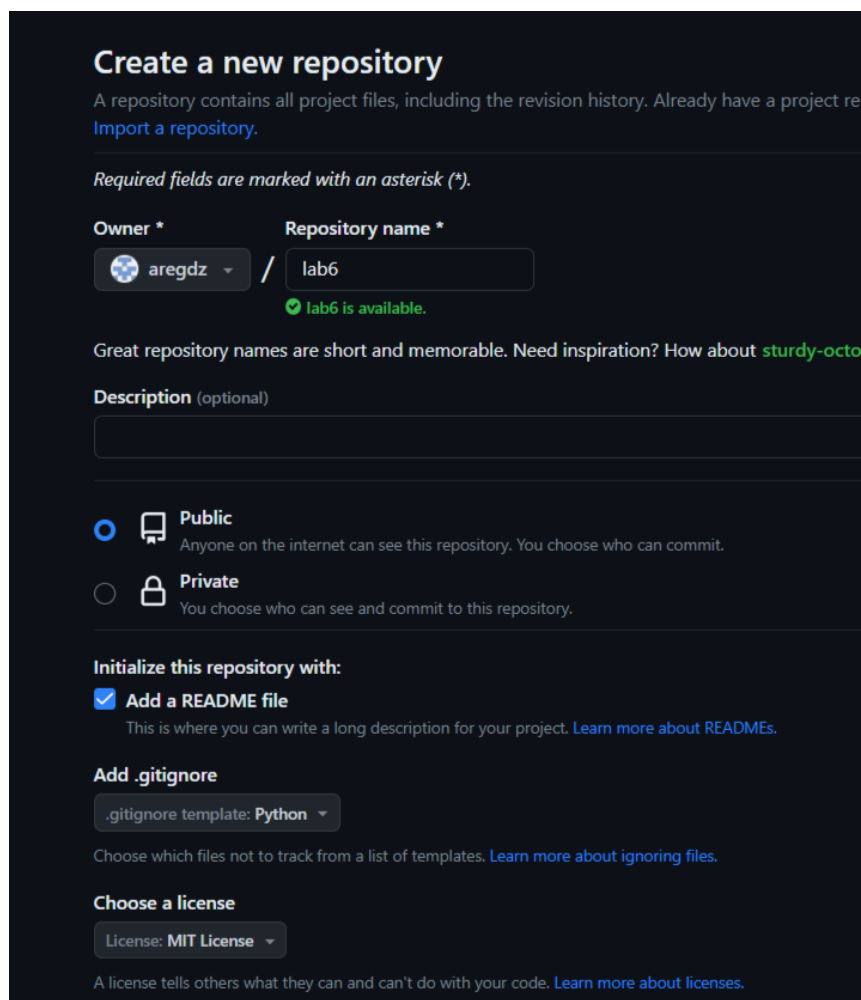
Ставрополь 2022

Тема: Лабораторная работа 2.1 Основы языка Python

Цель работы: исследование процесса установки и базовых возможностей языка Python версии 3.x.

Выполнение работы:

1. Изучил теоретический материал работы.
2. Создал репозиторий на git.hub.



The screenshot shows the GitHub 'Create a new repository' page. The title is 'Create a new repository'. Below it, a subtitle says 'A repository contains all project files, including the revision history. Already have a project repository? [Import a repository.](#)'. A note states 'Required fields are marked with an asterisk (*)'. The form has two main sections: 'Owner' and 'Repository name'. The 'Owner' is set to 'aregdz' with a dropdown arrow. The 'Repository name' is 'lab6', with a green checkmark indicating 'lab6 is available.'. Below this, a message says 'Great repository names are short and memorable. Need inspiration? How about [sturdy-octo](#)'. There is an optional 'Description' field. The 'Visibility' section has two options: 'Public' (selected with a radio button) and 'Private'. The 'Initialize this repository with:' section has a checked checkbox for 'Add a README file'. Below this is a section for 'Add .gitignore' with a dropdown menu set to 'Python'. The 'Choose a license' section has a dropdown menu set to 'MIT License'. At the bottom, there is a note about licenses: 'A license tells others what they can and can't do with your code. [Learn more about licenses.](#)'

Рисунок 1 – создание репозитория

3. Клонировал репозиторий.

```
MINGW64:/c/Users/aregd/OneDrive/Рабочий стол/git6
cd
aregd@DESKTOP-5KV9QA9 MINGW64 ~
$ cd "C:\Users\aregd\OneDrive\Рабочий стол\git6"

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git6
$ git clone https://github.com/aregdz/lab6.git
Cloning into 'lab6'...
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.

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git6
$
```

Рисунок 2 – клонирование репозитория 4.

Дополнить файл gitignore необходимыми правилами.

```
.gitignore – Блокнот
Файл Правка Формат Вид Справка
# Created by .ignore support plugin (hsz.mobi)
### Python template
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
env/
build/
develop-eggs/
dist/
downloads/
```

Рисунок 3 - .gitignore для IDE PyCharm

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

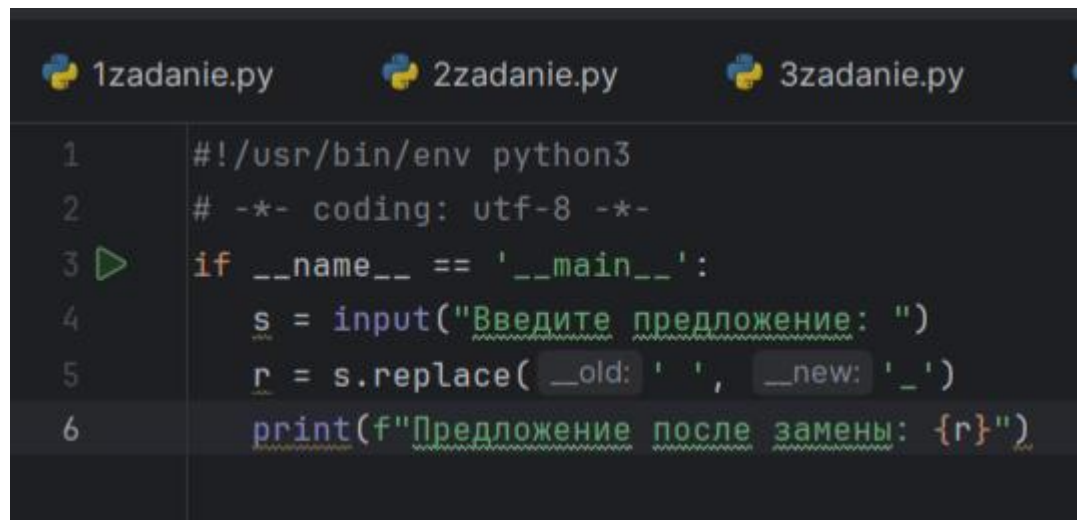
```
aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git 5/lab5 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git 5/lab5 (develop)
$ |
```

Рисунок 4 – создание ветки develop

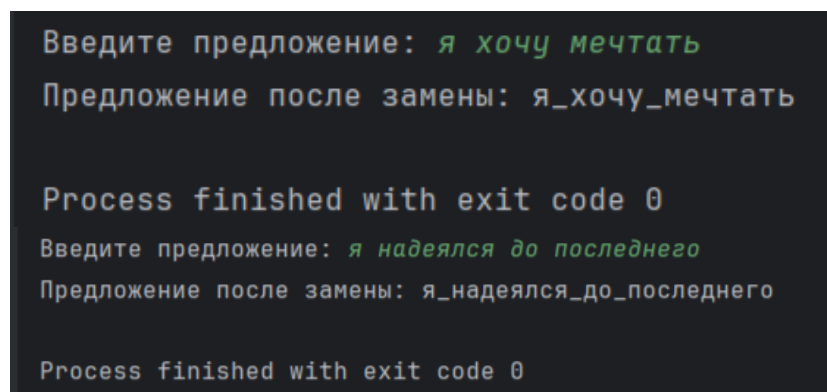
6. Проработать примеры лабораторной работы. Создать для каждого примера отдельный модуль языка Python. Зафиксировать изменения в

репозитории. Привести в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных вводимых с клавиатуры.



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

Рисунок 5 – пример 1



```
Введите предложение: я хочу мечтать
Предложение после замены: я_хочу_мечтать

Process finished with exit code 0
Введите предложение: я надеялся до последнего
Предложение после замены: я_надеялся_до_последнего

Process finished with exit code 0
```

Рисунок 6 – примеры выполнения для примера 1

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

Рисунок 7 – пример 2

```
Введите слово: жизнь
жиль

Process finished with exit code 0

Введите слово: life
le

Process finished with exit code 0
```

Рисунок 8 – примеры выполнения для 2 примера

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 import sys
4 if __name__ == '__main__':
5     s = input("Введите предложение: ")
6     n = int(input("Введите длину: "))
7     # Проверить требуемую длину.
8     if len(s) >= n:
9         print(
10             "Заданная длина должна быть больше длины предложения",
11             file=sys.stderr
12         )
13         exit(1)
14     # Разделить предложение на слова.
15     words = s.split(' ')
16     # Проверить количество слов в предложении.
17     if len(words) < 2:
18         # Количество пробелов для добавления.
19         delta = n
20         for word in words:
21             delta -= len(word)
22         # Количество пробелов на каждое слово.
23         w, r = delta // (len(words) - 1), delta % (len(words) - 1)
24         # Сформировать список для хранения слов и пробелов.
25         lst = []
26         # Пронумеровать все слова в списке и перебрать их.
27         for i, word in enumerate(words):
28             lst.append(word)
29             # Если слово не является последним, добавить пробелы.
30             if i < len(words) - 1:
31                 # Определить количество пробелов.
32                 width = w
33                 if r > 0:
34                     width += 1
35                     r -= 1
36                 # Добавить заданное количество пробелов в список.
37                 if width > 0:
38                     lst.append(' ' * width)
39         # Вывести новое предложение, объединив все элементы списка lst.
40         print(''.join(lst))
```

Рисунок 9 – пример 3

```
Введите предложение: быть или не быть
Введите длину: 20
быть или не быть

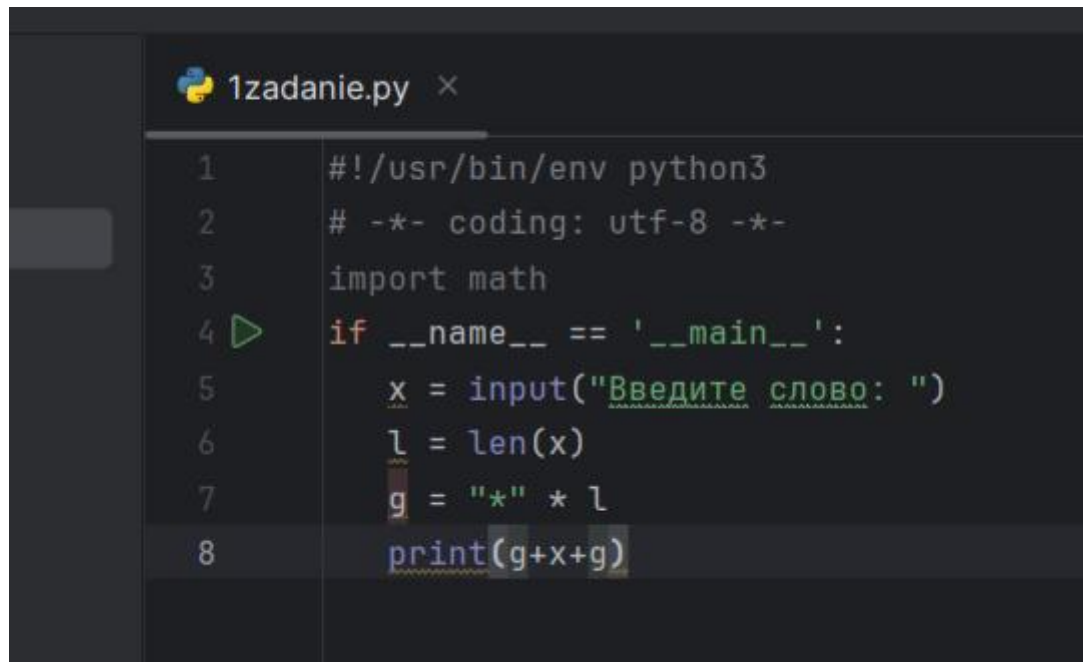
Process finished with exit code 0

Введите предложение: я пронсулся и понял что зря
Введите длину: 34
я пронсулся и понял что зря

Process finished with exit code 0
```

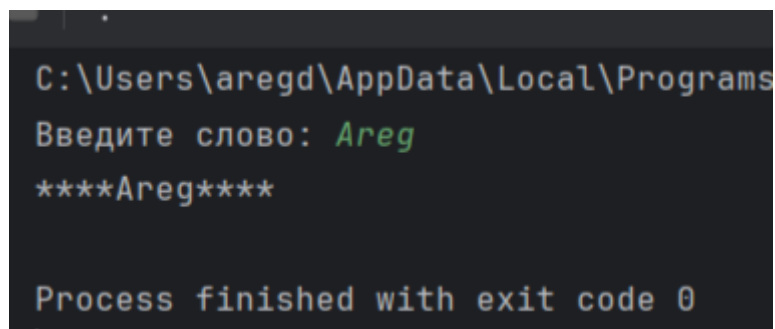
Рисунок 10 – примеры выполнения для примера 3

7. Дано слово. Добавить к нему в начале и конце столько звездочек, сколько букв в этом слове.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import math
4  if __name__ == '__main__':
5      x = input("Введите слово: ")
6      l = len(x)
7      g = "*" * l
8      print(g+x+g)
```

Рисунок 11 – решение задания 1



```
C:\Users\aregd\AppData\Local\Programs
Введите слово: Areg
****Areg****

Process finished with exit code 0
```

Рисунок 12– результат выполнения задания 1

8. Даны два слова. Определить, сколько начальных букв первого слов совпадает с начальными буквами второго слова. Рассмотреть два случая: известно, что слова разные; слова могут быть одинаковыми.

```
1zadanie.py 2zadanie.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import math
4  if __name__ == '__main__':
5      word1 = input("Введите 1 слово: ")
6      word2 = input("Введите 2 слово: ")
7      if word1 == word2:
8          print(f"Слова полностью совпадают, их длина: {len(word1)} букв")
9      else:
10         if len(word1) < len(word2):
11             min_len = len(word1)
12         else:
13             min_len = len(word2)
14         count = 0
15         for i in range(min_len):
16             if word1[i] == word2[i]:
17                 count += 1
18             else:
19                 break
20         if count == 0:
21             print("Слова не имеют совпадающих начальных букв.")
22         else:
23             print(f"Количество совпадающих начальных букв: {count}")
```

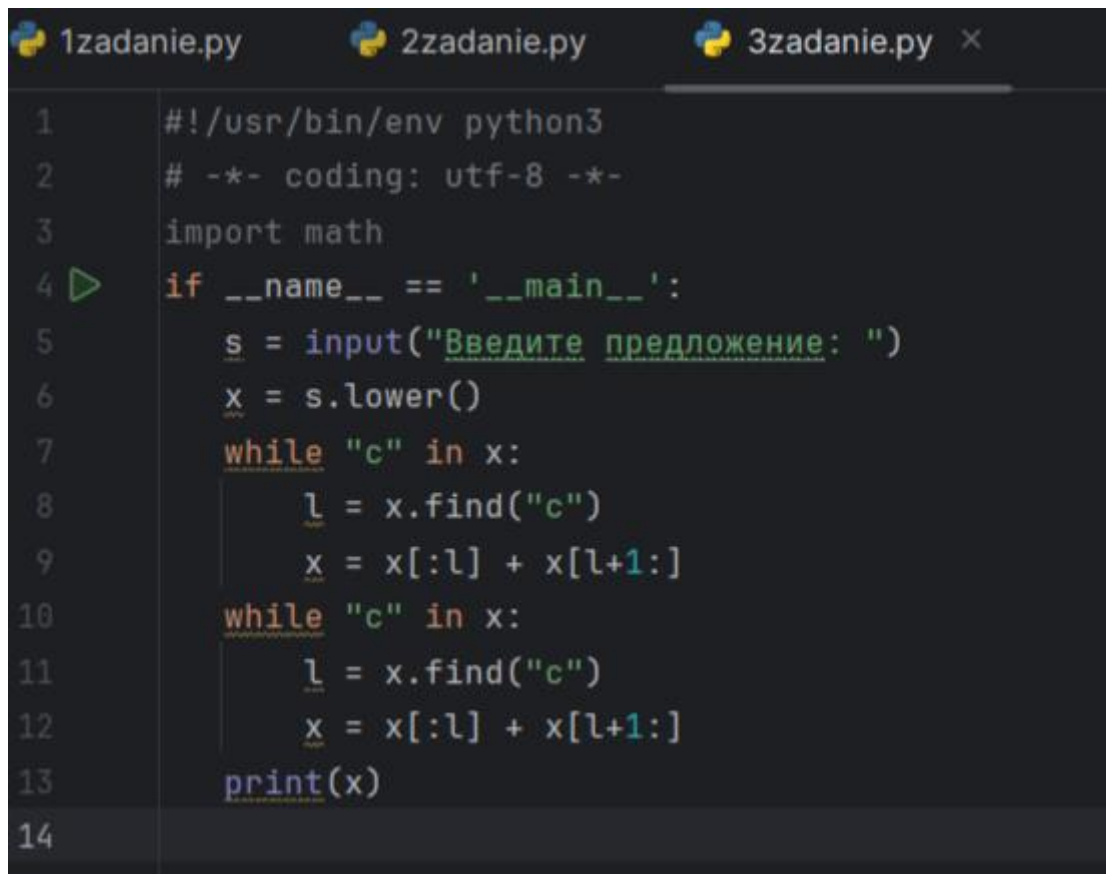
Рисунок 13 – решение задания 2

```
C:\Users\aregd\AppData\Local\Programs\Python
Введите 1 слово: areg
Введите 2 слово: aregpizh-22-1
Количество совпадающих начальных букв: 4

Process finished with exit code 0
```

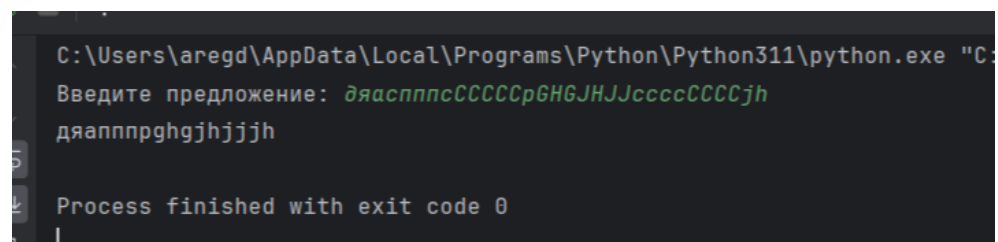
Рисунок 14 – результат выполнения задания 2

9. Дано предложение. Удалить из него все буквы с (как в кириллице так и на латинице).



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import math
4  if __name__ == '__main__':
5      s = input("Введите предложение: ")
6      x = s.lower()
7      while "c" in x:
8          l = x.find("c")
9          x = x[:l] + x[l+1:]
10     while "c" in x:
11         l = x.find("c")
12         x = x[:l] + x[l+1:]
13     print(x)
14
```

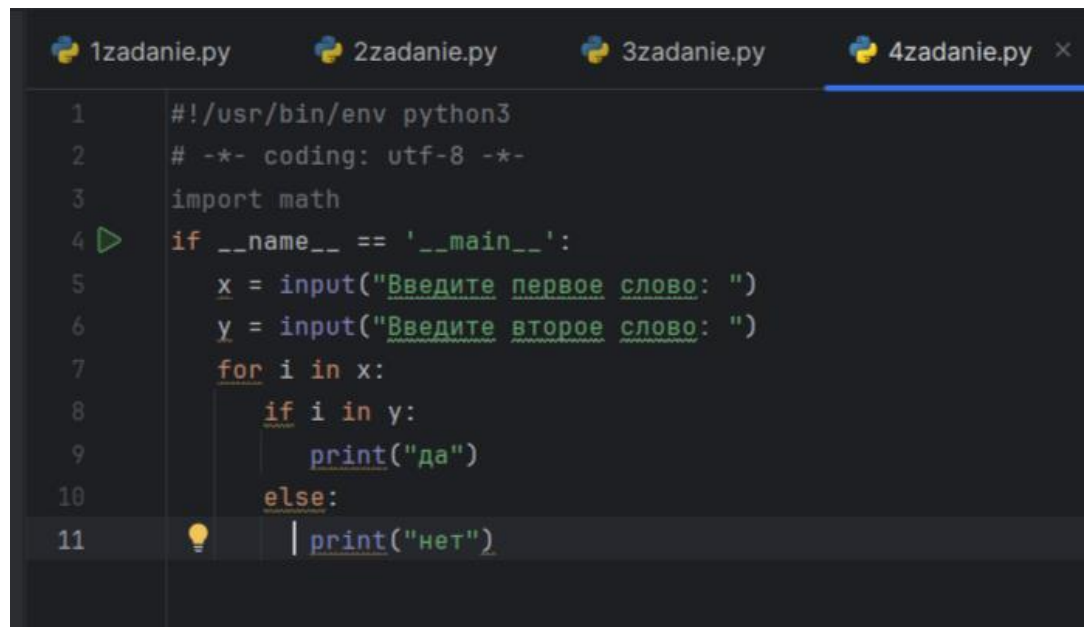
Рисунок 15 – решение задания 3



```
C:\Users\aregd\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\aregd\AppData\Local\Programs\Python\Python311\python.exe"
Введите предложение: дяапппсСССССрGHGJHJJccccCCCCjh
дяапппргhgjhjjjh
Process finished with exit code 0
```

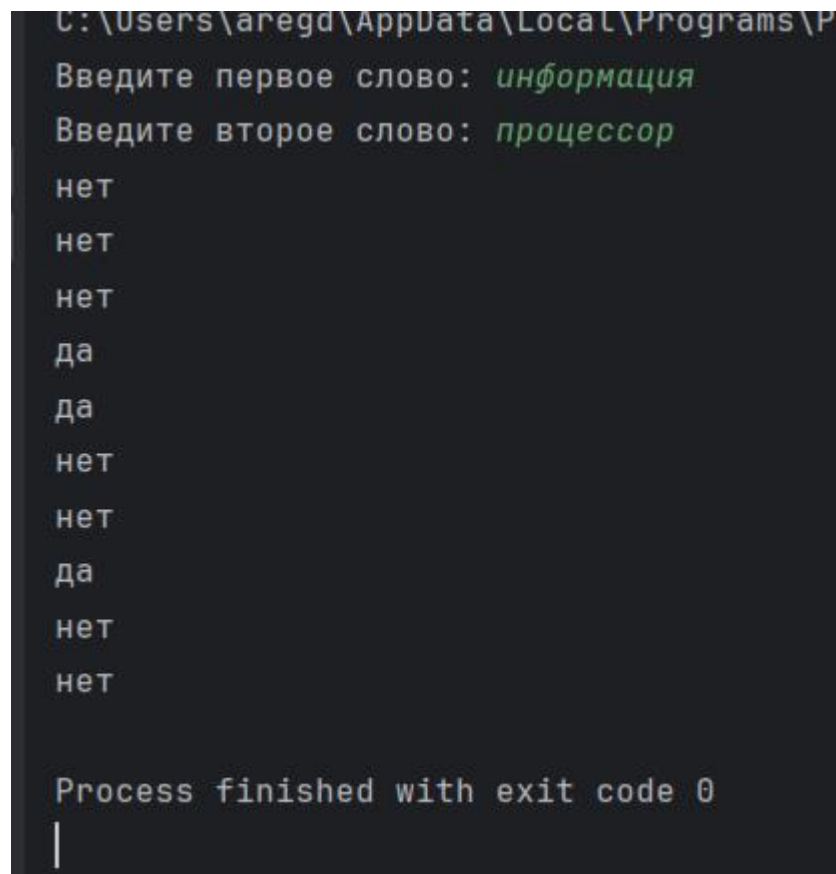
Рисунок 16 – результат выполнения задания 3

10. Даны два слова. Для каждой буквы первого слова (в том числе для повторяющихся в этом слове букв) определить, входит ли она во второе слово. Например, если заданные слова информация и процессор, то для букв первого из них ответом должно быть: нет нет нет да да нет нет да нет нет.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import math
4  if __name__ == '__main__':
5      x = input("Введите первое слово: ")
6      y = input("Введите второе слово: ")
7      for i in x:
8          if i in y:
9              print("да")
10         else:
11             print("нет")
```

Рисунок 17 – решение задания повышенной сложности



```
C:\Users\aregd\AppData\Local\Programs\Python\Python39\python.exe C:\Users\aregd\AppData\Local\Programs\Python\Python39\python.exe
Введите первое слово: информация
Введите второе слово: процессор
нет
нет
нет
да
да
нет
нет
да
нет
нет

Process finished with exit code 0
```

Рисунок 18 – результат выполнения задания повышенной сложности

11.Зафиксировал все изменения в github в ветке develop.

```
MINGW64/c:/Users/aregd/OneDrive/Рабочий стол/git6/lab6
$ cd "C:\Users\aregd\OneDrive\Рабочий стол\git6\lab6"
aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git6/lab6 (develop)
$ git add .
aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git6/lab6 (develop)
$ git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   PyCharm/1primer.py
    new file:   PyCharm/1zadanie.py
    new file:   PyCharm/2primer.py
    new file:   PyCharm/2zadanie.py
    new file:   PyCharm/3primer.py
    new file:   PyCharm/3zadanie.py
    new file:   PyCharm/4zadanie.py

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git6/lab6 (develop)
$ git commit -m"Сохранение изменений"
[develop 2678294] Сохранение изменений
7 files changed, 118 insertions(+)
create mode 100644 PyCharm/1primer.py
create mode 100644 PyCharm/1zadanie.py
create mode 100644 PyCharm/2primer.py
create mode 100644 PyCharm/2zadanie.py
create mode 100644 PyCharm/3primer.py
create mode 100644 PyCharm/3zadanie.py
create mode 100644 PyCharm/4zadanie.py

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git6/lab6 (develop)
$ git push
fatal: The current branch develop has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin develop

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git6/lab6 (develop)
$
```

Рисунок 19 – фиксация изменений в ветку develop

12. Слил ветки.

```
aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git6/lab6 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git6/lab6 (main)
$ git merge develop
Updating a023992..2678294
Fast-forward
 PyCharm/1primer.py | 6 ++++++
 PyCharm/1zadanie.py | 8 +++++++
 PyCharm/2primer.py | 12 ++++++++
 PyCharm/2zadanie.py | 23 ++++++++
 PyCharm/3primer.py | 45 ++++++++
 PyCharm/3zadanie.py | 13 ++++++++
 PyCharm/4zadanie.py | 11 ++++++++
7 files changed, 118 insertions(+)
create mode 100644 PyCharm/1primer.py
create mode 100644 PyCharm/1zadanie.py
create mode 100644 PyCharm/2primer.py
create mode 100644 PyCharm/2zadanie.py
create mode 100644 PyCharm/3primer.py
create mode 100644 PyCharm/3zadanie.py
create mode 100644 PyCharm/4zadanie.py

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git6/lab6 (main)
$
```

Рисунок 20 – сливание ветки develop в ветку main

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?

Работа со строками в Python очень удобна. Существует несколько литералов строк. (строки в апострофах и в кавычках, экранированные последовательности - служебные символы, строки в тройных апострофах или кавычках).

3. Какие операции и функции существуют для строк?

Некоторые функции: chr(), ord(), len(), str(). Строки можно умножать на числа, строки можно прибавлять между собой.

4.Как осуществляется индексирование строк?

Индексирование строк осуществляется с использованием квадратных скобок [], и индексы начинаются с 0. Вы можете получить доступ к отдельным символам в строке или извлекать подстроки, указывая индексы. Можно указывать отрицательные индексы, тогда счёт пойдет с обратной стороны.

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.

6.Почему строки Python относятся к неизменяемому типу данных?

Строки — один из типов данных, которые Python считает неизменяемыми, что означает невозможность их изменять. 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?

Можно проверить оператором “in”.

9. Как найти индекс первого вхождения подстроки в строку?

string.find(<sub>[, <start>[, <end>]]) ищет в строке заданную подстроку. s.find(<sub>) - возвращает первый индекс в s который соответствует началу строки <sub>

10. Как подсчитать количество символов в строке?

Можно воспользоваться `len(строка)`.

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

`string.count(<sub>[, <start>[, <end>]])` -подсчитывает количество вхождений подстроки в строку.

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?

Для поиска подстроки в заданной части строки в Python вы можете использовать метод строки `find()`, который вернет индекс начала первого вхождения подстроки в заданной части строки. Если подстрока не найдена, метод вернет -1.

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()` возвращает `true` когда строка `s` не пустая и все ее символы являются цифрами, а `false` если нет:

16 Как разделить строку по заданному символу?

Для разделения строки по заданному символу или подстроке в Python вы можете использовать метод строки `split()`. Этот метод разбивает строку на список подстрок с использованием указанного разделителя и возвращает этот список. `string.rsplit(sep=None, maxsplit=-1)` делит строку на список из подстрок.

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

`s.isupper()` возвращает `true` , если строка `s` не пустая, и все содержащиеся в ней буквенные

символы являются заглавными, и в `false` , если нет.

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?

Можно указать `[::-1]`

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

Для объединения списка строк в одну строку, где элементы разделяются дефисами, вы можете использовать метод строки `join()`

22 Как привести всю строку к верхнему или нижнему регистру?

Для приведения строки к верхнему (заглавному) или нижнему (строчному) регистру в Python, вы можете использовать методы строк `upper()` и `lower()`, соответственно.

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

Для преобразования первого и последнего символов строки к верхнему регистру в Python, вы можете 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()` возвращает `true`, если строка `s` не пустая, и все содержащиеся в нем буквенные

символы строчные, а `false` если нет.

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

Когда нужно делить `s` на строки и возвращать 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?

Для замены всех вхождений определенной подстроки в заданной строке в Python, вы можете использовать метод строки `replace()`. Этот метод заменяет все вхождения подстроки на другую подстроку и возвращает новую строку с заменами.

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

Можно сравнить нужную нам последовательность символов с индексом той части строки, в которой должна быть последовательность символов.

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

`s.isspace()` возвращает `True`, если `s` не пустая строка, и все символы являются пробельными, а `False`, если нет

29 Что случится, если умножить некую строку на 3?

Строка повторится 3 раза.

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

`s.title()` возвращает копию, `s` в которой первая буква каждого слова преобразуется в

верхний регистр, а остальные буквы — в нижний регистр:

31 Как пользоваться методом `partition()` ?

`partition()` - это метод строки в Python, который позволяет разделить строку на три части с использованием заданного разделителя. Метод возвращает кортеж, в котором первый элемент - это часть строки до первого вхождения разделителя, второй элемент - сам разделитель, и третий элемент - часть строки после первого вхождения разделителя.

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

Метод `rfind()` используется в Python для поиска последнего вхождения подстроки в строке. Он возвращает индекс последнего вхождения заданной подстроки в строке. Если подстрока не найдена, метод `rfind()` возвращает `-1`. Когда нам нужно найти последнее вхождение определенной подстроки в строке, `rfind()` позволяет это сделать без необходимости итерации с конца строки.