

**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

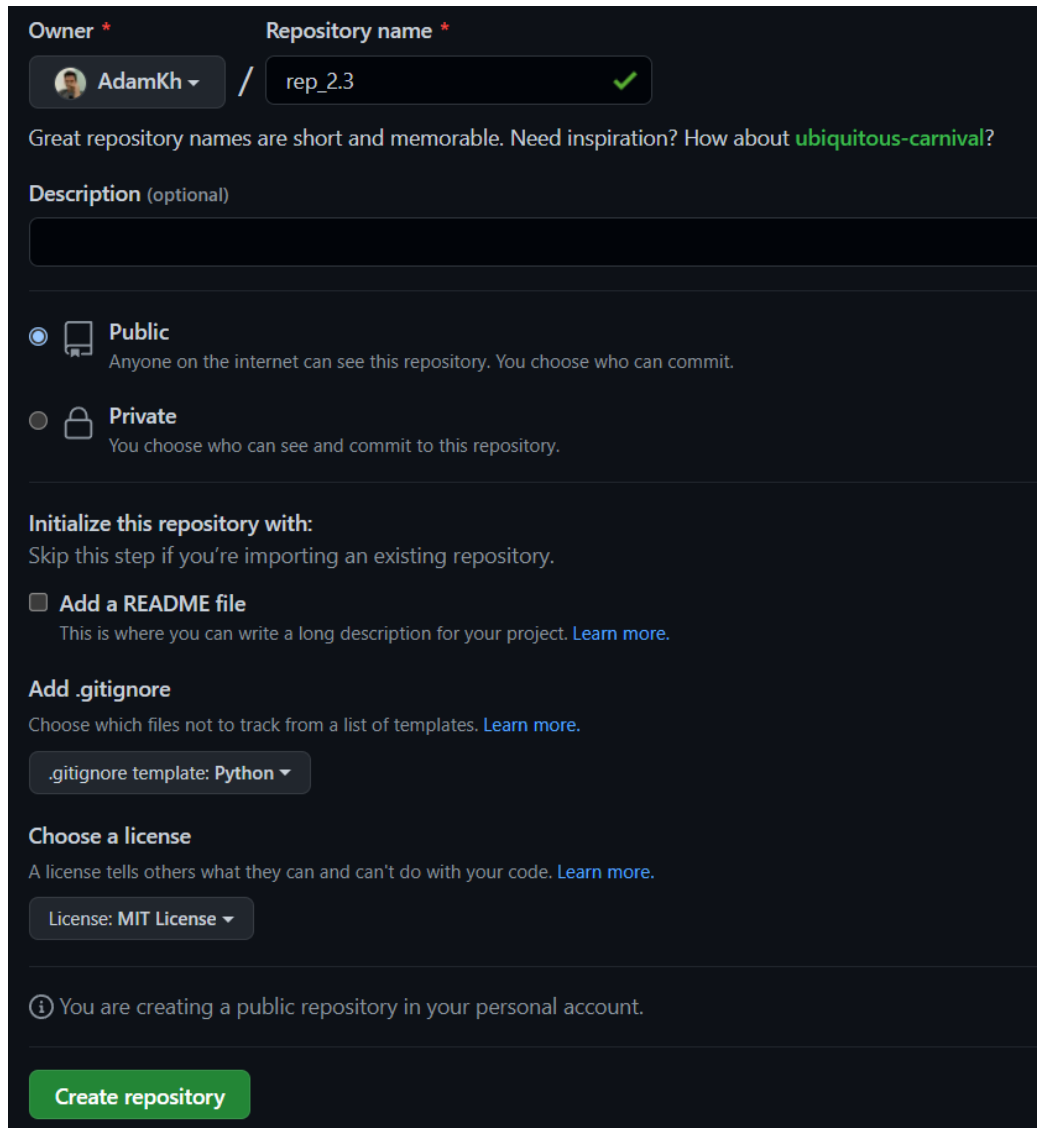
группы ИВТ-б-о-21-1

Хашиев Адам Мухарбекович

Ставрополь 2022

## Выполнение работы:

1. Создал репозиторий в GitHub «rep 2.2» в который добавил .gitignore, который дополнил правила для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на лок. сервер и организовал в соответствии с моделью ветвления git-flow.



Owner \* AdamKh / Repository name \* rep\_2.3 ✓

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-carnival?](#)

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:  
Skip this step if you're importing an existing repository.

☐ Add a README file  
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▾

Choose a license  
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▾

ⓘ You are creating a public repository in your personal account.

Create repository

Рисунок 1.1 Создание репозитория

```
C:\Users\adamk\OneDrive\Рабочий стол>git clone https://github.com/AdamKh/rep_2.3.git
Cloning into 'rep_2.3'...
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 1.2 Клонирование репозитория

```

C:\Users\adamk\OneDrive\Рабочий стол\rep_2.3>git flow init
Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/adamk/OneDrive/Рабочий стол/rep_2.3/.git/hooks]

```

Рисунок 1.3 Организация репозитория в соответствии с моделью ветвления  
git-flow

```

... 153 .gitignore
@@ -1,3 +1,117 @@
1 +
2 + # Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
3 + # Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm
4 +
5 + ### PyCharm ###
6 + # Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
7 + # Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839
8 +
9 + # User-specific stuff
10 + .idea/**/workspace.xml
11 + .idea/**/tasks.xml
12 + .idea/**/usage.statistics.xml
13 + .idea/**/dictionaries
14 + .idea/**/shelf
15 +
16 + # AWS User-specific
17 + .idea/**/aws.xml
18 +
19 + # Generated files
20 + .idea/**/contentModel.xml
21 +
22 + # Sensitive or high-churn files
23 + .idea/**/dataSources/
24 + .idea/**/dataSources.ids

```

Рисунок 1.4 Изменение .gitignore

2. Создал проект PyCharm в папке репозитория, проработал примеры ЛР.

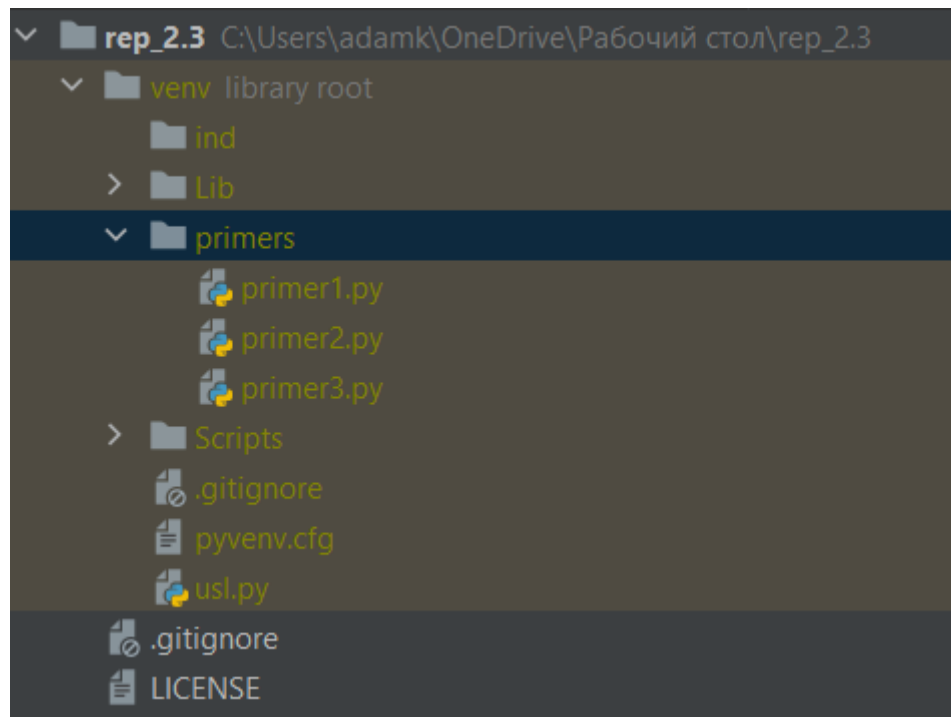


Рисунок 2.1 Создание проекта в PyCharm

```
Введите предложение: asdf asdf asdfnjk   asd s sasdf
Предложение после замены: asdf_asdf_asdfnjk____asd_s_sasdf
```

Рисунок 2.2 Рез-т выполнения программы

```
Введите слово: gashkdf
gaskdf
```

Рисунок 2.3 Рез-т выполнения программы

```
Введите предложение: qwe qweq qqw   er w a asdfda rt
Введите длину: 39
qwe   qweq   qqw   er   w   a   asdfda   rt
```

Рисунок 2.4 Рез-т выполнения программы

3. (22 вариант). Выполнил 3 индивидуальных задания и задание повышенной сложности.

**Задание 1.** Дано предложение. В нем слова разделены одним пробелом (начальные и конечные пробелы и символ «-» в предложении отсутствуют). Определить количество слов в предложении.

```

1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      """
5      Дано предложение. В нем слова разделены одним пробелом
6      (начальные и конечные пробелы и символ «-» в предложении отсутствуют).
7      Определить количество слов в предложении.
8      """
9
10  ▶  if __name__ == '__main__':
11      s = str(input('Введите предложение:\n'))
12      am_words = 0
13      for i in range(0, len(s)):
14          if s[i] == ' ':
15              am_words += 1
16      if s[-1] != ' ':
17          am_words += 1
18      print(am_words)

```

Рисунок 3.1 Листинг программы

```

Введите предложение:
kłj hkj gh hhfgh
4

```

Рисунок 3.2 Выполнение программы

**Задание 2.** Дано слово. Поменять местами его третью и последнюю буквы.

```

1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      """
5      Дано слово. Поменять местами его третью и последнюю буквы.
6      """
7
8  ▶  if __name__ == '__main__':
9      w = str(input('Введите слово: '))
10     tmp = list(w)
11     s = tmp[2]
12     tmp[2] = tmp[-1]
13     tmp[-1] = s
14     w = ''.join(tmp)
15     print(w)
16

```

Рисунок 4.1 Листинг программы

```
Введите слово: nj;hlj
njjhl;
```

Рисунок 4.2 Выполнение программы

**Задание 3.** Дано слово. Переставить его последнюю букву на место k-й. При этом k-ю, (k + 1)-ю, ..., предпоследнюю буквы сдвинуть вправо на одну позицию.

```
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      """
5      Дано слово. Переставить его последнюю букву на
6      место k-й. При этом k-ю, (k + 1)-ю, ..., предпоследнюю
7      буквы сдвинуть вправо на одну позицию.
8      """
9
10  ▶  if __name__ == '__main__':
11      w = str(input('Введите слово: '))
12      k = int(input('Введите k: '))
13      tmp = list(w)
14      s = tmp[-1]
15      for i in range(len(w) - 1, k - 1, -1):
16          tmp[i] = tmp[i-1]
17      tmp[k-1] = s
18      w = ''.join(tmp)
19      print(w)
20
```

Рисунок 4.3 Листинг программы

```
Введите слово: ethfglxj
Введите k: 2
ejthfglx
```

Рисунок 4.4 Выполнение программы

**Усложненное задание.** Дано предложение. Напечатать все его слова в порядке убывания их длин.

```
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import re
5
6
7 ▶ if __name__ == '__main__':
8     s = input('Введите предложение\n')
9     s1 = re.split(' |,|.', s)
10    i = 0
11    while i < len(s1):...
16    for i in range(len(s1)):
17        for j in range(len(s1)-1):
18            if len(s1[j]) < len(s1[j+1]):
19                s1[j], s1[j + 1] = s1[j + 1], s1[j]
20    s = ''
21    for i in range(0, len(s1)):
22        s += s1[i] + ' '
23    print(s)
24
```

Рисунок 4.5 Листинг программы

```
Введите предложение
ыапвапр апрапроа ароап апл о лу
апрапроа ыапвапр ароап апл лу о
Process finished with exit code 0
```

Рисунок 4.6 Выполнение программы

4. Сделал коммит, выполнил слияние с веткой main, и запустил изменения в уд. репозиторий.

```
C:\Users\adamk\OneDrive\Рабочий стол\rep_2.3>git add .
C:\Users\adamk\OneDrive\Рабочий стол\rep_2.3>git commit -m "formatted according to pep8"
[develop 3cb8760] formatted according to pep8
4 files changed, 8 insertions(+), 4 deletions(-)
```

Рисунок 4.1 Фиксация и коммит файлов

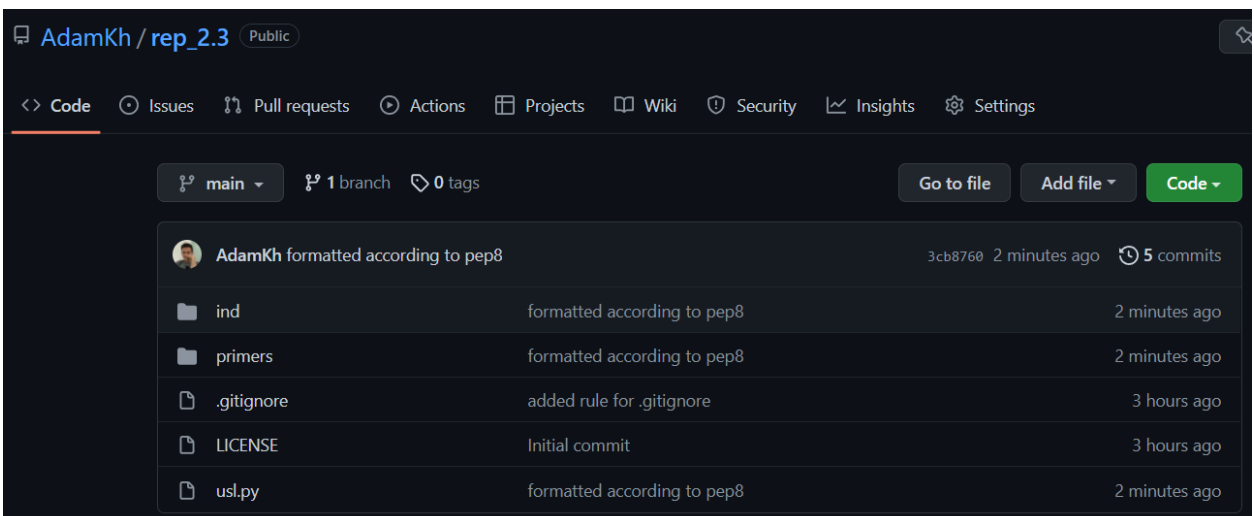
```
C:\Users\adamk\OneDrive\Рабочий стол\rep_2.3>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\adamk\OneDrive\Рабочий стол\rep_2.3>git merge develop
Updating 649aece..3cb8760
Fast-forward
 .gitignore          | 2 +-
 ind/ind1.py         | 12 ++++++++
 ind/ind2.py         | 13 ++++++++
 ind/ind3.py         | 15 ++++++++
 primers/primer1.py  | 7 +++++
 primers/primer2.py  | 13 ++++++++
 primers/primer3.py  | 57 ++++++++
 usl.py              | 23 ++++++++
 8 files changed, 141 insertions(+), 1 deletion(-)
 create mode 100644 ind/ind1.py
 create mode 100644 ind/ind2.py
 create mode 100644 ind/ind3.py
 create mode 100644 primers/primer1.py
 create mode 100644 primers/primer2.py
 create mode 100644 primers/primer3.py
 create mode 100644 usl.py
```

Рисунок 4.2 Слияние ветки develop с main

```
C:\Users\adamk\OneDrive\Рабочий стол\rep_2.3>git push
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
Delta compression using up to 8 threads
Compressing objects: 100% (22/22), done.
Writing objects: 100% (22/22), 3.20 KiB | 818.00 KiB/s, done.
Total 22 (delta 9), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (9/9), completed with 1 local object.
To https://github.com/AdamKh/rep_2.3.git
 649aece..3cb8760 main -> main
```

Рисунок 4.3 Пуш коммитов



AdamKh / rep\_2.3 Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

Commit	Message	Time
AdamKh	formatted according to pep8	2 minutes ago
ind	formatted according to pep8	2 minutes ago
primers	formatted according to pep8	2 minutes ago
.gitignore	added rule for .gitignore	3 hours ago
LICENSE	Initial commit	3 hours ago
usl.py	formatted according to pep8	2 minutes ago

Рисунок 4.4 Изменения на уд. сервере



## **Контр. вопросы и ответы на них:**

### **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, "сырые" строки, строки в тройных апострофах или кавычках

### **3. Какие операции и функции существуют для строк?**

Сложение, дублирование, длина строки, извлечение среза и т. д.

### **4. Как осуществляется индексирование строк?**

Доступ к символам в строках основан на операции индексирования – после строки или имени переменной, ссылающейся на строку, в квадратных скобках указываются номера позиций необходимых символов.

### **5. Как осуществляется работа со срезами для строк?**

Есть три формы срезов. Самая простая форма среза: взятие одного символа строки, а именно,  $S[i]$  — это срез, состоящий из одного символа, который имеет номер  $i$ , при этом считая, что нумерация начинается с числа 0. То есть если  $S = \text{'Hello'}$ , то  $S[0] == \text{'H'}$ ,  $S[1] == \text{'e'}$ ,  $S[2] == \text{'l'}$ ,  $S[3] == \text{'l'}$ ,  $S[4] == \text{'o'}$ .

Если указать отрицательное значение индекса, то номер будет отсчитываться с конца, начиная с номера -1.

Срез с двумя параметрами:  $S[a:b]$  возвращает подстроку из  $b$ -а символов, начиная с символа с индексом  $a$ , то есть до символа с индексом  $b$ , не включая его.

## **6. Почему строки Python относятся к неизменяемому типу данных?**

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

## **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?**

```
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?**

Эти строки улучшают читаемость кода, а также работают быстрее чем другие способы форматирования. F-строки задаются с помощью литерала «f» перед кавычками. Пример: `print(f"Меня зовут {name} Мне {age} лет.")`

## **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?**

`s.isalpha()`

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

`s.istitle()`

**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 символ 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()` ?**

`s.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?**

`Asd*3 = AsdAsdAsd`

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

`s.title()`

**31. Как пользоваться методом `partition()`?**

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