

и 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 «Программирование на Python»
Вариант 31

Выполнил:
Репкин Александр Павлович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной
техники и автоматизированных
систем», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., канд. техн. наук,
доцент кафедры инфокоммуникаций

(подпись)

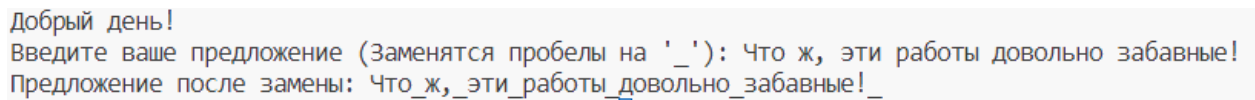
Отчет защищен с оценкой _____ Дата защиты _____
Ставрополь, 2023 г.

Тема: Работа со строками в языке Python.

Цель: приобрести навыки по работе со строками при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы:

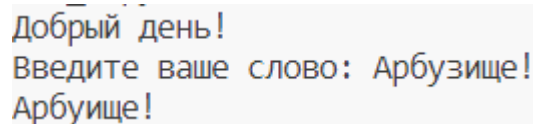
1. Выполнен первый пример. В нём производилась замена символа “ (Пробел) на символ “_” в введённом пользователем предложении.



```
Добрый день!  
Введите ваше предложение (Заменяются пробелы на '_'): Что ж, эти работы довольно забавные!  
Предложение после замены: Что_ж,_эти_работы_довольно_забавные!_
```

Рисунок 1. Полученный результат примера №1.

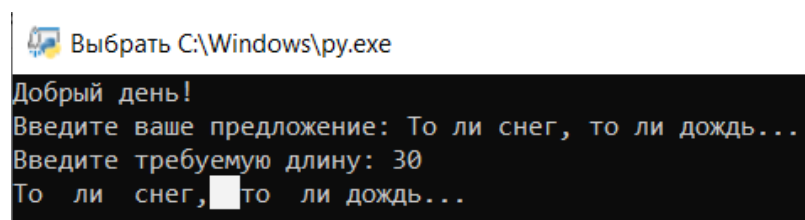
2. Выполнен второй пример. В нём требовалось: проанализировать введённое пользователем слово – если количество символов в нём чётное, то переменной *r* присваивалась сумма символов введённого слова – от 0 до середины (Не включительно) и от середины + 1 до конца.



```
Добрый день!  
Введите ваше слово: Арбузище!  
Арбузище!
```

Рисунок 2. Полученный результат примера №2.

3. Выполнен третий пример. В нём требовалось: увеличить строку до требуемого размера путём равномерного увеличения количества пробелов между словами.



```
Выбрать C:\Windows\py.exe  
Добрый день!  
Введите ваше предложение: То ли снег, то ли дождь...  
Введите требуемую длину: 30  
То ли снег, то ли дождь...
```

Рисунок 3. Полученный результат примера №3.

4. Выполнено индивидуальное задание №1: “Дан текст. Верно ли, что в нем есть пять идущих подряд одинаковых символов?”. Выполнено двумя способами – с помощью счётчика и проверкой каждых пяти символов. Согласно полученным данным, использование счётчика было быстрее, нежели постоянная проверка повторяющихся пяти элементов.

```

1  import time
2
3
4  if __name__ == '__main__':
5      words = input("Good day! Please, enter your sentence.")
6      # There are two variants - checking five letters and using a counter.
7      start_time = time.time()
8      contains = False
9      for i in range(0, len(words)-4):
10         if words[i] == words[i+1] == words[i+2] == words[i+3] == words[i+4]:
11             print("Yes, there are 5 symbols '", words[i], "'.")
12             contains = True
13             break
14     first_variant_time = time.time() - start_time
15     start_time = time.time()
16     count = 1
17     symbol = words[0]
18     for i in words[1:]:
19         if i == symbol:
20             count += 1
21             if count == 5:
22                 print("Yes, there are 5 symbols '", i, "'.")
23                 contains = True
24                 break
25         else:
26             count = 1
27             symbol = i
28     if not contains:
29         print("Sorry, there are no 5 equal symbols.")
30     second_variant_time = time.time() - start_time
31     print("First variant took ", first_variant_time,
32         " and Second variant took ", second_variant_time)
33

```

Рисунок 4. Полученный код индивидуального задания №1.

```

C:\Windows\py.exe
Good day! Please, enter your sentence. So, what should I enter?
Sorry, there are no 5 equal symbols.
First variant took 0.0 and Second variant took 0.0

C:\Windows\py.exe
Good day! Please, enter your sentence. Ok, ok! Fine, I'll enter some strange letters: rtrttttttrrrrr. Enough?
Yes, there are 5 symbols ' t '.
Yes, there are 5 symbols ' t '.
First variant took 0.0010247230529785156 and Second variant took 0.0

```

Рисунок 5. Полученный результат индивидуального задания №1.

5. Выполнено индивидуальное задание №2: “ Дан текст. Определить количество букв “и” в первом предложении. Рассмотреть два случая: известно, что буквы “и” в этом предложении есть; известно, что букв “и” в тексте может не быть.”.

```

1  if __name__ == '__main__':
2      words = input(
3          "Good day! Please, enter your text (Rememer, that only first sentence will be analised).\n")
4      count = 0
5      for i in words:
6          if i != "." and i != "!" and i != "?" and i != ";":
7              if i == "и":
8                  count += 1
9              else:
10                 break
11     if count == 0:
12         print("Sorry, there were no 'и' in the first sentence.")
13     else:
14         print("In first sentence, there are", count, "symbols 'и'.")
15

```

Рисунок 6. Полученный код индивидуального задания №2.

Good day! Please, enter your text (Remember, that only first sentence will be analised).
Что ж, в данном предложении я использовал ровно 3 буквы и. Ха-ха, на самом деле в прошлом предложении их 4, а здесь ещё 3.
In first sentence, there are 4 symbols 'и'. _

Рисунок 7. Полученный результат индивидуального задания №2.

6. Выполнено индивидуальное задание №3: “Дан текст. Определить, является ли он правильной десятичной записью целого числа.”.

```
1  if __name__ == '__main__':
2      words = input("Good day! Please, enter your text.\n")
3      for i in words:
4          if not (58 > ord(i) > 47):
5              print("Sorry, but text can't be transformed into Integer type.")
6          else:
7              print("Sure, that text can be transformed into Integer type.")
8
```

Рисунок 8. Полученный код индивидуального задания №3.

```
Good day! Please, enter your text.
Well, here is your text:12345,12345
Sorry, but text can't be transformed into Integer type.
```

Рисунок 9. Полученный отрицательный результат индивидуального задания №3.

```
Good day! Please, enter your text.
123454321
Sure, that text can be transformed into Integer type.
```

Рисунок 10. Полученный положительный результат индивидуального задания №3.

7. Выполнено задание повышенной сложности, №31: “Даны два предложения. Напечатать слова, которые встречаются в двух предложениях только один раз.”.

```
1  import re
2
3
4  if __name__ == '__main__':
5      first_sentence = input("Good day, please, enter first sentence:\n")
6      second_sentence = input("Great, now we need second sentence:\n")
7      print("\nVery well, there is a list of words, that are mentioned only once in each sentence:\n")
8      # В связи с тем, что split принимает только один разделитель, использована функция re.split, принимающая сразу
9      # несколько разделителей. \s не считает место между двумя исключённых символов за отдельное слово.
10     # Заключение в [] работает в качестве списка разграничивающих символов.
11     first_words = re.split(r"[-;,\.\s]\s*", first_sentence)
12     second_words = re.split(r"[-;,\.\s]\s*", second_sentence)
13     for i in first_words:
14         if 0 < first_words.count(i) < 2 and 0 < second_words.count(i) < 2:
15             print(i)
16
```

Рисунок 11. Полученный код усложнённого задания.

```
Good day, please, enter first sentence:
Итак,итак,что тут у нас такого интересного, м?
Great, now we need second sentence:
Итак, итак,итак, у нас тут мало чего нашего такого интересного, угу.

Very well, there is a list of words, that are mentioned only once in each sentence:

Итак
тут
у
нас
такого
интересного
```

Рисунок 12. Полученный результат усложнённого задания.

8. Ответы на вопросы.

1) Что такое строки в языке Python?

Ответ: строки - упорядоченные последовательности символов, используемые для хранения и представления текстовой информации. С помощью строк можно работать со всем, что может быть представлено в текстовом виде.

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

Ответ: для задания строковых литералов, в Python можно использовать: кавычки (“smth”); одинарные кавычки (‘smth’); тройные кавычки (‘‘smth’’). Кроме того, есть возможность экранирования специальных символов – вставка специальных символов (“\nsmth\n”). Если же нужно отключить экранирование, перед строкой необходимо добавить символ r (r“\nsmth\n”).

3) Какие операции и функции существуют для строк?

Ответ: строки можно складывать (Оператор “+” – конкатенация строк. Возвращает строку, состоящую из двух переданных ему строк) и умножать (Оператор “*”. Повторяет строку заданное количество (“строка” * “число”) раз). Также, можно проводить проверку на содержание в строке определённого фрагмента текста (Оператор принадлежности подстроки in – в случае, если в строке содержится заданная подстрока, возвращает True, иначе – False). Помимо операций, есть также функции: chr() – преобразует целое число в символ согласно таблице ASCII; ord() – обратная chr функция,

преобразующая символ в число согласно таблице ASCII; `len()` – функция, возвращающая длину строки; `str()` – функция, преобразующая данные в строковый тип.

4) Как осуществляется индексирование строк?

Ответ: индексация - отдельные элементы в упорядоченном наборе данных могут быть доступны с помощью числового индекса или ключа. Доступ к отдельным символам в строке можно получить, указав имя строки и число в квадратных скобках [“номер”]. Индексация строк начинается с 0: у первого символа индекс 0, у следующего 1, затем 2 и т.д. Индекс последнего символа в Python `len(“строка”)-1`. Индексы строк также могут быть указаны отрицательными числами. В этом случае индексирование начинается с конца строки: -1 относится к последнему символу, -2 к предпоследнему и так далее.

5) Как осуществляется работа со срезами для строк?

Ответ: Python позволяет извлечь подстроку из строки, известную как “string slice”. Выражение вида `“строка”[m:n]` возвращает часть строки, начинающуюся с индекса `m`, и до индекса `n-1`. Если пропустить первый индекс, то срез начинается с начала строки, а если пропустить второй индекс, то срез длится до конца строки. Пропуск обоих индексов возвращает ссылку на исходную строку. Если первый и второй индекс равны или первый больше второго, то возвращается пустая строка. Добавление дополнительного: и третьего индекса означает шаг, который указывает, сколько символов следует пропустить после извлечения каждого символа в срезе. Также можно указать отрицательное значение шага, в этом случае Python идет с конца строки. Начальный/первый индекс должен быть больше конечного/второго индекса

6) Почему строки Python относятся к неизменяемому типу данных?

Ответ: строки — один из типов данных, которые Python считает неизменяемыми, что означает невозможность их изменять. Однако, Python дает возможность изменять (заменять и перезаписывать) строки. Попытка заменить символы в строке по индексам приводят к ошибке `TypeError`, однако нет особой необходимости изменять строки, ведь можно сгенерировать копию

исходной строки с необходимыми изменениями (есть минимум 2 способа: использовать встроенный метод `string.replace(x, y)` или суммировать строку с необходимыми изменениями).

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

Ответ: функция `istitle()` позволяет проанализировать требуемую строку и узнать, с заглавной ли буквы начинается каждое слово. Если да – возвращает `True`, иначе – `False`.

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

Ответ: оператор `in` позволяет узнать, входит ли строка в состав другой строки. Если входит – возвращает `True`, иначе – `False`.

9) Как найти индекс первого вхождения подстроки в строку?

Ответ: функция `find()` позволяет найти подстроку внутри требуемой строки. Если подстрока найдена, то выводится индекс первой найденной подстроки, а если в строке вообще нет такой подстроки, то возвращается `-1`.

10) Как подсчитать количество символов в строке?

Ответ: функция `len()` возвращает число символов в требуемой строке.

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

Ответ: функция `count()` позволяет посчитать, сколько раз в требуемой строке встречается указанный символ.

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

Ответ: f-строки – способ форматирования строк в Python, который позволяет встраивать значения переменных и выражений непосредственно в строку с помощью префикса `'f'`. Чтобы использовать f-строки, нужно просто создать строку с префиксом `'f'` и вставить значения переменных или выражения в фигурные скобки внутри неё. Например: `extra = "wow!"`
`print(f'smth{extra}')` выведет `smth wow!`.

13) Как найти подстроку в заданной части строки?

Ответ: функция `find()` позволяет найти подстроку внутри требуемой строки. Если подстрока найдена, то выводится индекс первой найденной подстроки, а если в строке вообще нет такой подстроки, то возвращается -1.

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

Ответ: метод `format()` вставляет значения в фигурные скобки (`{...}`) внутри строки.

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

Ответ: функция `isdigit()` проверяет переданную ей строку. В случае, если в строке содержатся только цифры, функция вернёт `True`, иначе - `False`.

16) Как разделить строку по заданному символу?

Ответ: функция `split()` разделяет строку на подстроки относительно требуемого символа-разделителя.

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

Ответ: функция `islower()` позволяет проверить, состоит ли строка только из строчных букв, возвращая `True` или `False`.

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

Ответ: функция `islower()` анализирует полученную строку, если в ней все буквенные символы – строчные (маленькие), то возвращает `True`, иначе - `False`. Зная это, можно применить функцию `islower()` к первому символу требуемой строки.

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

Ответ: в Python возможно добавить число к строке, если при суммировании преобразовать число в строковый тип данных при помощи `str()`.

20) Как “перевернуть” строку?

Ответ: для переворачивания строки можно использовать срез. Требуется лишь указать отрицательный шаг = `1:[:-1]`.

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

Ответ: функция `join()` объединяет элементы списков в строки, разделяя отдельные строки с использованием переданного ей символа. Так, передав ей требуемый список строк и указав разделительный символ, `join()` создаст новую требуемую строку.

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

Ответ: функция `upper()` преобразует переданную ей строку, заменяя все символы в ней на заглавные. Функция `lower()` наоборот, делает все символы в строке строчными.

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

Ответ: функции `capitalize()` и `upper()` позволят перевести первый и последний символы строки в верхний регистр. Первая функция заменит первый символ на заглавный, в то время как передав функции `upper()` последний символ строки (При помощи строка[`len(строка)-1`] или строка[`-1`]), будет получен требуемый символ в верхнем регистре.

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

Ответ: функция `islower()` анализирует полученную строку, если в ней все буквенные символы – строчные (маленькие), то возвращает `True`, иначе - `False`.

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

Ответ: функция `splitlines()` делит строку на подстроки относительно присутствующих в строке разделителей (`\n`, `\r`, `\x1c`, прочее). Данный метод довольно полезен при считывании данных из текстовых документов, так как убирает множество ненужных символов самостоятельно.

26) Как в заданной строке заменить на что-либо все вхождения некой подстроки?

Ответ: функция `replace()` позволяет заменять в требуемой строке подстроку на переданное ей значение.

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

Ответ: чтобы проверить, начинается ли строка с требуемых символов, можно использовать функцию `startswith("строка")`. Если же нужно проверить с конца строки, то можно использовать функцию `endswith("строка")`.

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

Ответ: для проверки строки на содержание чего-либо кроме пробелов, можно использовать функцию `isspace()`, возвращающую `False`, если есть символы, не являющиеся пробелами.

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

Ответ: при умножении строки, создаётся новая строка, состоящая из повторённых 3 раза изначальных строк друг за другом.

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

Ответ: для замены всех первых символов слов в строке на заглавные, есть функция `title`.

31) Как пользоваться методом `partition()`?

Ответ: `partition("разделитель")` делит строку на основе полученного разделителя. Находя первое совпадение с разделителем, `partition` возвращает три строки – до разделителя, сам разделитель, строку после разделителя.

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

Ответ: `rfind`, в отличие от `find`, ищет подстроку в строке начиная с конца, а не с начала.

Вывод: в ходе выполнения лабораторной работы были исследованы процесс установки и базовые возможности языка Python версии 3.x. Выполнены общие, индивидуальные и усложнённые задания. Получены знания о начальных этапах работы со строками в языке программирования Python.