ЛАБОРАТОРНАЯ РАБОТА

Тема: ***Использование списков и строк при обработке данных***

Цель работы: Изучение приемов обработки данных, сочетающих использование списков и строк

Содержание

Расщепление строк на подстроки. Метод `split`

Часто символьные строки строятся из набора информационных элементов, которые друг от друга отделяются определенным символом или символами. Например, в строке "`1,2,3,4`" для отделения чисел используется *разделитель* "запятая". Чтобы работать с отдельными элементами информации, такую строку нужно "разобрать на части" — представить строку как совокупность подстрок.

Строки в Питоне обладают методом строковых объектов `.split(разделитель)`, который возвращает *список* подстрок из исходной строки. В общем случае `разделитель` — это некоторая строка.

Если разделитель в строке не найден, то метод возвращает список из одного элемента — исходной строки. Разделитель в найденные подстроки не включается.

```
>>> 'Ростов-->Москва'.split('-->')
['Ростов', 'Москва']
>>> 'ab,,,cd'.split(',')
['ab', '', '', 'cd']
>>> 'ab,,,cd'.split('-')
['ab,,,cd']
```

ЗАДАНИЕ 1 (*Использование списков для представления данных 2*)

По-прежнему рассматривается задача, сформулированная в лаб. раб. 8.

Имеются результаты ЕГЭ абитуриента, поступающего на специальности информационного профиля: Математика — 78, Информатика — 75, Русский язык — 62.

Необходимо выполнить простейшую программную обработку данных — вывести результаты ЕГЭ по дисциплинам и напечатать сумму баллов.

Данные о ЕГЭ представлены списком строк, в котором каждая строка содержит информацию об одной дисциплине:

```
ege2 = ["Математика - 78", "Информатика - 75", "Русский язык - 62"]
```

Для решения задачи предлагается получить два списка — один `subjects` с названиями дисциплин, т.е. `["Математика", "Информатика", "Русский язык"]`, а второй — `marks` с баллами ЕГЭ, т.е. `[78, 75, 62]`. Программа пишется в несколько этапов.

а) Вспомогательная функция — преобразование данных по предмету в список.

Описывается вспомогательная функция `get_info(строка)`, которой передается строка с информацией по предмету (например, `"Математика - 78"`). Функция возвращает данные в виде списка (т.е. `["Математика", 78]`).

Выполнить необходимое преобразование можно с помощью метода строковых объектов `.split(разделитель)`.

В рассматриваемых данных два интересующих нас информационных элемента (предмет и баллы) разделены строкой из трех символов `" - "`.

Метод `.split()` возвращает список *строк*, но баллы для суммирования требуются нам как числа.

Поэтому в функции `get_info()` нужно

– с помощью вызова метода `.split()` получить список из названия предмета и набранных баллов,

– преобразовать прямо *в этом списке* значение баллов в числовой формат, и

– сделать полученный список возвращаемым значением.

Написать и протестировать функцию `get_info()`.

б) Применение* `get_info()` *для преобразования формата исходных данных.

В качестве примера применения оператора цикла выполнить преобразование исходных данных из формата `ege2` в форматы `ege3` и `ege4`

```
ege3 = [ "Математика", 78, "Информатика", 75, "Русский язык", 62 ]
ege4 = [ ["Математика",78],  ["Информатика", 75], ["Русский язык", 62]]
```

Для этого на глобальном уровне с помощью конструктора `list()` или пустых квадратных скобок создаются два пустых списка: `lst3=list()` и `lst4=[]`, в которых будет накапливаться информация.

Затем в цикле `for-in` по очереди извлекаются строки с информацией. На каждом шаге цикла эти данные передаются `get_info()`. Возвращаемое ей значение добавляется в списки `lst3` и `lst4` так, чтобы одном случае строился общий список (как в `ege3`), а в другом — список списков (как в `ege4`).

После завершения оператора цикла добавить в программу вывод на экран значений `lst3` и `lst4`.

в) Получение результатов

Используя опыт пункта б) построить два списка: `subjects` с названиями дисциплин, т.е. `["Математика", "Информатика", "Русский язык"]`, и `marks` с баллами ЕГЭ, т.е. `[78, 75, 62]`.

Для этого определить функции `subjects(ege)` и `marks(ege)`, формирующие из `ege2` и затем возвращающие, соответственно, список дисциплин и список оценок.

Определить функцию `print_results(subjects,marks)`, которая построчно выводит результаты ЕГЭ в формате "`предмет — баллы`" и сумму баллов. Для вычисления суммы баллов использовать функцию `sum` (см. справочную информацию).

Поиск подстрок в строке

Поиск подстроки в строке — одна из самых распространенных и простых задач поиска информации. Поэтому в библиотеках языков программирования всегда присутствуют функции, выполняющие такой поиск. При описании этих функций искомую подстроку часто обозначают как *needle* (*англ.* иголка), а строку, в которой ведётся поиск, как *haystack* (*англ.* стог сена).

В Питоне для поиска подстроки `needle` в строке `haystack` используется метод `.find()`:

```
haystack.find(needle)
```

При успешном поиске метод возвращает позицию, с которой начинается подстрока.

Если подстрока не найдена, то метод возвращает `-1`.

```
>>> "abcabc".find("cab")
2
>>> "abcabc".find("cba")
-1
```

У метода `.find()` есть два необязательных аргумента, задающих начальную и конечную позицию в строке, ограничивающих поиск только определенной частью строки.

```
>>> "abcabc".find("c",3)
5
>>> "abcabc".find("a",1,5)
3
```

Замечание

У строк в Питоне есть второй метод для поиска подстрок — `.index()`. Он имеет одно важное отличие от `.find()` — если подстрока не найдена, то генерируется ошибка (исключительная ситуация).

ЗАДАНИЕ 2 (*Обработка данных с помощью поиска подстрок*)

Данные о ЕГЭ представлены *одной* строкой, которая содержит всю информацию о результатах:

```
ege1 = "Математика – 78, Информатика – 75, Русский язык – 62".
```

а) Проверка наличия результатов ЕГЭ по заданной дисциплине.

Абитуриенты могут сдавать ЕГЭ по разным предметам, поэтому возникает задача: выяснить, сдавал ли абитуриент ЕГЭ по конкретной дисциплине?

Для этого определить функцию `get_tested(ege, subject)`, которой передаются `ege` — строка с результатами ЕГЭ и `subject` — название учебной дисциплины (предмета).

Функция `get_tested()` играет роль предиката — возвращает `True` или `False`.

Проверка выполняется с помощью метода `find()`.

Чтобы при поиске не возникало проблем с разницей в регистрах букв, обе строки с помощью метода `.lower()` (или `.upper()`) предварительно преобразовать к нижнему (или верхнему регистру).

***б) Тестирование функции* `get_tested()`.**

Для тестирования разработанной функции использовать строку `ege1` и названия предметов "`Информатика`", "`информатика`", "`Физика`" и "`ФИЗИКА`".

Для тестирования программы создать список `subjects`, в который поместить названия четырех тестовых дисциплин.

С помощью оператора `for-in` сформировать цикл, в котором поочередно для каждого элемента списка `subjects` выполняется вызов функции `get_tested()`. По результатам проверки функция выводит на экран сообщение:

"`ЕГЭ по дисциплине .... сдан`" или "`ЕГЭ по дисциплине .... не сдан`".

Удаление ведущих и концевых пробельных символов

В процессе обработки строк часто появляются подстроки имеющие лишние бесполезные пробельные символы.

Это пробельные символы, которые находятся в начале строки до первого непробельного символа (`leading` whitespaces, ведущие пробелы) или в конце (`trailing` whitespaces, концевые пробелы) после последнего непробельного строки.

Лишние пробельные символы не только занимают память, но и могут мешать некоторым видам обработки. Поэтому в языках программирования имеются специальные методы или функции, позволяющие удалять ведущие и конечные символы.

В Питоне это методы с названием от английского слова strip ("обдирать") .

`строка.strip()` — удаление пробелов в начале и в конце строки,

`строка.lstrip()` — удаление пробелов начале строки (**l**`eft` **strip**),

`строка.rstrip()` — удаление пробелов в конце строки (**r**`ight` **strip**).

Примеры.

```
>>> sp = "  Это строка   "
>>> sp.strip()
'Это строка'
>>> sp.lstrip()
'Это строка   '
>>> sp.rstrip()
'  Это строка'
```

Замечание.

В языках программирования для функций, удаляющих ведущие и концевые пробелы, часто используется другое имя — `trim` (*англ.* подстригать).

ЗАДАНИЕ 3 (*Преобразование исходных данных*)

Написать функцию `convert_list14(ege)`, которая преобразует список `ege1` в список `ege_new`, имеющий формат `ege4`.

Для разбора строки применить метод `.split(',')`.

При этом требуется учесть, что в исходной строке `ege1` после запятых допускается разное количество пробельных символов

`"Математика – 78,  Информатика – 75, Русский язык – 62"`,

которые не должны попасть в итоговый список.

Для удаления пробельных символов использовать методы `.strip()`.

Протестировать работу функции `convert_list14(ege)`.

Вопросы для самоконтроля

1. Какие действия выполняет метод `.split()` и что он возвращает?
2. Что может выступать в роли разделителя при выполнении метода `.split()`?
3. Какими двумя способами можно в программе создать пустой список?
4. Какие действия выполняет метод `.find()` и что он возвращает?
5. Какие дополнительные необязательные параметры имеет метод `.find()` и для чего ими можно пользоваться?
6. Что понимается под ведущими и концевыми пробелами?
7. Можно ли удалить только ведущие пробелы или только концевые пробельные символы?
8. Какую основную структуру имеет оператор `for-in` и для чего он предназначен?
9. Можно ли с помощью оператор `for-in` организовать арифметический цикл на подобии того оператора `for`, который есть в языке Паскаль?
10. В чем разница в применении к спискам методов `.append()` и `.extend()`?

Справочная информация

Цикл for-in

Операторы цикла состоят из заголовка и тела.

Тело оператора цикла — это блок инструкций, вложенных в него, которые записываются с отступом по отношению к заголовку.

Оператор цикла `for-in` предназначен для поочередного извлечения и обработки элементов из последовательности объектов данных (*итерирования*) . Он имеет следующую структуру.

Заголовочная часть начинается со служебного слова `for` и *имени переменной*. Переменная вспомогательная и предназначается для хранения очередных значений, извлекаемых из последовательности. Хотя её имя может быть выбрано произвольно, имеет смысл выбрать его так, чтобы характеризовать смысл хранимых данных.

Заголовочную часть продолжает ключевое слово `in` и *последовательность*, из которой будут извлекаться объекты данных. Последовательность может быть указана явно (строка, список и др.) или с помощью переменной, ссылающейся на последовательность.

Завершается заголовок *двоеточием* и переводом строки. Далее с отступом записываются инструкции тела оператора.

```
for <переменная> in <последовательность>:
    <операторы>
```

В следующем примере в цикле поочередно выводятся символы, из которых состоит строка:

```
for symbol in "abc":
    print(symbol, end='-')            # a-b-c-
```

Оператор `for` приспособлен для работы с коллекциями. Но в программах часто требуется использовать арифметические циклы. Т.е. перебираются не элементы существующей последовательности, а значения из арифметической прогрессии (`от ... до ... с шагом ...`).

Для реализации циклов арифметического типа в Питоне имеется специальная конструкция `range`, которой можно передать три параметра — "начальное значение", "**до** какого значения" и "шаг изменения" (разность прогрессии). Её можно рассматривать как конструктор виртуальной (имитируется, реально её нет) последовательности, хранящий элементы арифметической прогрессии, описывающие изменения параметра цикла.

```
>>> for i in range(1,10,2):
    print(i, end=' ')
1 3 5 7 9
```

Некоторые методы и функции списков

1. Методы списков.

а) Метод `lst.`**`append`**`(elem)` добавляет *один* элемент данных `elem` в конец списка `lst`. Дописываемый в конец списка объект `elem` может принадлежать к любому тип данных. Длина списка `lst` увеличится на единицу.

б) Метод `lst.`**`insert`**`(position, elem)` вставляет в список `lst` один *элемент* `elem` перед элементом, который имеет индекс `position`.

в) Метод `lst.`**`extend`**`(lst1)` добавляет в конец списка `lst` элементы *списка* `lst1` (слияние списков). Длина списка `lst` увеличивается на величину длины списка `lst1`.

2. Функции списков.

`sum``(список)` — возвращает сумму членов числовой последовательности,

`max``(список)` — возвращает максимальный элемент последовательности,

`min``(список)` — возвращает минимальный элемент последовательности.