ЛАБОРАТОРНАЯ РАБОТА

Тема: ***Подстроки и срезы. Форматирование вывода***

Цель работы: Продолжение знакомства с использованием в программах символьных строк.

Отчет: Отчет представляет собой документ Word, в который последовательно из окна интерпретатора копируются команды и результаты выполнения для каждого задания с необходимыми объяснениями.

Содержание

Подстрока — это часть заданной *строки*, состоящая из *одного или нескольких* её символов.

В программировании с подстроками связано две основных операции:

- выделение подстроки из строки с дальнейшем обработкой и
- поиск подстроки в строке.

Инструмент для выделения подстрок в Питоне — оператор индексации **`[]`**. Он применяется к строковым литералам или переменным, хранящим строки.

Наборы символов, составляющие подстроку, определяются числовыми параметрами, помещаемыми в квадратные скобки оператора. В простейшем случае это одно число, определяющее позицию одного искомого символа.

Замечания

1. Особенности трактовки строк в Питоне как последовательностей и использование оператора индексации для выделения отдельных символов были рассмотрены в лаб. раб. №3. Сводная информация приведена ниже в справочном разделе.

2. Задача поиска подстрок решается с помощью библиотечных функций и будет рассматриваться позже.

Подстроки из нескольких символов

Подстроки, состоящие из одного символа можно назвать *простейшими*, потому что в операции индексации они определяются *одним* числовым параметром.

Чтобы выделить подстроку, состоящую из нескольких *последовательно расположенных символов*, нужно указать два параметра: где подстрока начинается и где заканчивается.

Поэтому для получения подстроки из нескольких символов, индексное выражение имеет более сложную структуру:

```
строка[beg : end].
```

Так как подстрока — это "вырезка" какой-то части строки, то подстроки в Питоне относятся к категории *срезов* (*англ.* `slice`). Параметры `beg` и `end` являются свойствами среза.

По умолчанию оба индекса, и стартовый, и конечный, отсчитываются от `0`. Правый (конечный) индекс определяет позицию, *до которой* распространяется подстрока, т.е. соответствующий ему символ не включается в подстроку:

Индексы (один или оба) могут быть заданы отрицательными значениями. В любом случае границы подстроки определяются в порядке [`левая : правая`].

Действует правило: чтобы получить *непустой срез*, второй индекс должен определять позицию, находящуюся *правее* первой. Поэтому, если для правой границы будет задан индекс `0`, то результатом будет пустой срез.

Замечание

Формально подстрока должна содержать хотя бы один символ. Но в случаях, аналогичных последнему примеру, иногда говорят о результате как о пустой подстроке.

Выход правого индекса за пределы строки

Если при выделении простейшей подстроки из *одного символа* значение индекса больше длины строки, то возникает ошибка.

Но, если значение *конечного* индекса будет больше длины строки при задании среза, то ошибки не возникает — подстрока будет заканчиваться последним символом исходной строки.

Граничные значения индексов по умолчанию

В определении среза *обязательным является только двоеточие*, а индексы (один или оба) могут отсутствовать.

При отсутствии явно указанного значения, индексы получают значения, принятые *по умолчанию*.

Для *начального индекса* это `0`, т.е. начало *под*строки будет совпадать с началом строки.

Для *конечного индекса* значение по умолчанию равно длине строки (первая позиция после окончания строки). Поэтому конец *под*строки будет совпадать с концом строки.

Когда один или оба индекса в операторе опущены, то используются оба значения по умолчанию, т.е. срезом будет вся строка.

ЗАДАНИЕ 1 (*Выделение подстрок*)

1. Вычислить следующие выражения и объяснить полученные результаты.

```
>>> "abcdefg"[0:3]
>>> "abcdefg"[2:5]
>>> "abcdefg"[5:2]
>>> "abcdefg"[1:15]
>>> "abcdefg"[15]
```

2. Вычислить и объяснить, как выделяются подстроки в случае отрицательных значений индексов. Объяснить полученные результаты:

```
>>> "987654321"[-4:-1]
>>> "987654321"[-1:-4]
>>> "987654321"[-1:-1]
```

3. Вычислить следующие выражения и объяснить, в каком случае какие значения индексов используются по умолчанию.

```
>>> "0123456789"[:4]
>>> "0123456789"[2:]
>>> "0123456789"[:]
```

4. Вычислить следующие выражения и объяснить полученные результаты.

```
>>> "987654321"[-4:-1]
>>> "987654321"[-4:3]
>>> "987654321"[-4:7]
>>> "987654321"[-4:0]
```

5. В магазине цены на товары хранятся в виде вещественных чисел. Чтобы упростить расчеты с покупателями, после подсчета общей суммы покупок копейки отбрасываются.

Предположим, получена сумма покупок (вещественное число с двумя цифрами после запятой, задать по своему усмотрению).

Написать инструкцию, формирующую для чека сообщение:

"Сумма покупки ... руб., скидка ... коп."

Рекомендация: преобразовать сумму покупок в строку (как числа преобразовываются в строки?), выделить два среза (сумма в рублях и остаток в копейках) и "склеить" нужную строку.

Более сложные срезы

Гибкость оператора выделения подстроки может быть повышена за счет добавления третьего аргумента. Он записывает после двоеточия и определяет *шаг*, с которым из исходной строки должны выбираться символы подстроки (т.е. на сколько индекс следующего символа отстоит от предыдущего).

По умолчанию значение шага равно 1, т.е выбирать все символы по порядку.

Шаг может быть *отрицательным*. В этом случае срез строится *справа налево* и поэтому индекс начала должен быть больше конечного.

Модуль шага указывает интервал индексов между выбираемыми символами.

Подстроки как частный случай срезов

Подстроки получаются как срезы. Но при этом между подстроками и срезами есть различия.

а) Срезы можно формировать не только из символьных строк, но и из других типов последовательностей.

б) Подстроки по определению должны быть непустыми, а срезы могут быть пустыми.

в) Подстрока — это *связная* часть строки, а в срезах некоторые символы могут быть пропущены. Например, срез может быть построен из символов исходной строки, стоящих на четных позициях.

ЗАДАНИЕ 2 (*Срезы с шагом*)

1. Вычислить срезы и объяснить полученные результаты.

```
>>> "0123456789"[1:9:3]
>>> "0123456789"[::2]
```

2. Переменная хранит строку. Составить выражение, в котором из строки выделяются два среза. Первый срез состоит из символов, стоящих на четных местах (нуль считается четным), а второй — на нечетных местах. Срезы с помощью конкатенации объединяются в новую строку. Проверить на примере "1a2b3c4d5e6f".

3. Вычислить срезы с отрицательным шагом и объяснить результаты.

```
>>> "0123456789"[2:5:-1]
>>> "0123456789"[5:2:-1]
>>> "0123456789"[:2:-1]
>>> "0123456789"[::-1]
```

4. Палиндромами называются слова, читающиеся одинаково в обоих направлениях (перевертыши).

Переменная хранит строку из одного слова. Написать логическое выражение, в котором сравниваются исходное слово и слово, полученное из него перестановкой символов в обратном порядке. Если слово палиндром, то выражение возвращает `True`, иначе — `False`. Проверить правильность вычисления выражения на словах "`доход`" и "`поход`".

Форматирование вывода

Представление результатов вычислений в удобном для пользователя виде является естественным требованием к программам. Обычно требуется, чтобы сообщение содержало вспомогательный текст и данные, которые предлагаются в удобном для анализа виде. Например, для вещественного числа может потребоваться ограничить количество цифр в дробной части. Управление видом представления данных называется *форматированием*.

Так как есть правила форматирования, которые используются Питоном по умолчанию, то форматирование может выполняться неявно (как это было в предыдущих лабораторных работах).

Форматирование можно реализовать "вручную", комбинируя операции преобразования типов и конкатенации (как в задании 1.5). Однако это может приводить к достаточно громоздким и сложным действиям.

В Питоне множество шагов, необходимых для форматирования, можно объединить в одной инструкции. Для этого применяется двухместный (бинарный) оператор `%`, который *при работе со строками* имеет особый смысл.

Его левым операндом является *форматная строка*, описывающая шаблон, по которому должна быть построена результирующая строка.

А правый операнд — *заключенная в круглые скобки* совокупность *выражений* (через запятую), значения которых должны быть подставлены в шаблон. Т.о. оператор % реализует компактную запись множественной подстановки значений в шаблон результирующей строки.

Результатом вычисления выражения с `%` для строк является *строка*.

Форматная строка строится из обычного текста и *спецификаций формата*. Спецификации формата обеспечивают набор различных операций преобразования.

Каждая спецификация формата начинается *со знака процента* и содержит обязательный *спецификатор* формата — *символ*, определяющий тип форматирования.

Символы подобраны так, чтобы они были связаны с названиями конкретных типов данных. Например:

`d` или `i` — целое число в десятичной форме (**`d`**`ecimal` или **`i`**`nteger`),

`f` — вещественное число (**`f`**`loat`),

`u` — целое число без знака (**`u`**`nsigned`),

`x` или `X` — целое число в 16-ричной форме (`he`**`x`**`adecimal`),

`s` — строка (**`s`**`tring`),

`c` — символ (**`c`**`haracter`) и др.

Действие спецификатора может быть уточнено дополнительной информацией. Например, при выводе вещественных чисел можно указать количество цифр в целой и дробной частях числа: `%`**`4.1`**`f`. Если цифр в целой части числа больше, чем указано в спецификации, то ограничение *не* учитывается, цифры будут выведены все. Если в дробной части значащих цифр не хватает, то они будут дополнены нулями. А, если их больше, то после округления лишние цифры будут отброшены.

Однако, если вещественное число выводится с помощью спецификации `%d`, то дробная часть просто отбрасывается.

Пример.

Форматная строка содержит текст и спецификации формата:

```
>>> my_str="abcdefgh"
>>> print("Строка %s имеет длину %d" % (my_str, len(my_str)))
Строка abcdefgh имеет длину 8
```

Если при разборе форматной строки интерпретатор встречает спецификацию формата, то он обращается к списку в правой части, вычисляет там очередное значение, форматирует его нужным образом и подставляет результат в строку вместо спецификации формата. В приведенном примере будет установлено соответствие сначала между `%s` и `my_str`, а затем между `%d` и `len(my_str)`.

Замечание

1. В общем случае спецификации формата имеют свой достаточно развитый синтаксис и обеспечивают потребности сложного форматирования (см. справочную информацию).

2. Для форматирования также может быть применен метод `format`, который будет изучаться позже.

Так как символ процента *внутри форматных строк* является служебным, то, когда он встречается в тексте, выполняется "экранирование" — он удваивается. Например, `"Выполнение работ %d %%"`.

ЗАДАНИЕ 3 (*Форматирование вывода*)

1. Выполнить инструкции и объяснить полученные результаты:

```
>>> "%4.4f"%12.34
>>> "%4.4f"%123456.78
>>> "%4.4f"%1234.56987
>>> "%4.0f"%1234.56987
```

2. Выполнить задание 1.5, используя форматный вывод. Отбрасывание дробной части у суммы покупок должно быть предусмотрено в форматной строке, а вычисление скидки — в правой части оператора форматирования

Вопросы для самоконтроля

1. Что понимается под подстрокой? Какие основные операции связаны с подстроками?

2. Как получить подстроку, состоящую из одного символа?

3. Как получить подстроку, состоящую из нескольких символов? Как задаются границы подстроки?

4. Что происходит, когда правый граничный индекс имеет значение, большее длины строки?

5. Какие значения граничных индексов принимаются по умолчанию?

6. Для чего при задании среза указывается шаг?

7. Чем срезы похожи на подстроки и чем отличаются?

8. Что понимается под форматированием вывода?

9. Какую структуру имеет выражение форматирования с оператором %?

10. Какую структуру имеет форматная строка? Как устроена спецификация формата?

11. Какие спецификаторы используются для вывода отдельных символов, строки чисел в десятичном представлении?

Справочная информация

Основные принципы представление строк

В лаб. раб. №3 было изучено, что в Питоне

- строки рассматриваются как последовательности,
- каждый символ в строке имеет свою позицию, определяемую "координатой", которая называет индексом;
- индексы отсчитываются либо слева направо от нуля, либо справа налево от `-1`,
- можно выделить символ из строки, применяя операцию индексации [];
- результатом выделения будет *строка* из одного символа, т.е. *подстрока*.

Спецификаторы формата

В общем виде синтаксис использования спецификатора формата выглядит следующим образом:

`%[(`*`name`*`)][`*`flags`*`][`*`width`*`][.`*`precision`*`]`**`specifier`**

Спецификация формата начинается с символа процента, и завершается обязательным элементом — спецификатором формата `specifier`.

Между символом `%` и спецификатором может быть добавлена уточняющая информация: общая ширина поля, количество знаков после десятичной точки, способ выравнивания, показывать ли знак и ведущие нули у числа и многое другое.

Подробное описание элементов форматирования можно найти в документации Питона.

Основные спецификаторы формата приведены в следующей таблице.

Спецификатор формата	Назначение
`%c`	Символ
`%s`	Строка (или любой объект)
`%d` или `%i`	Целое число в десятичной форме
`%u`	Целое число без знака
`%f`	Вещественное число (с точкой)
`%o`	Целое число в восьмеричной системе
`%x` или `%X`	Шестнадцатеричное целое число (0-9**a-f**) или (0-9**A-F**)
`%e`	Число в научной нотации
`%%`	Символ %

Некоторые типы данных имеют альтернативные способы форматирования данных одного и того же типа, например, `%f` и `%e` обеспечивают разные способы представления вещественных чисел.