Введение

Слова и предложения имеют основополагающее значение для того, как мы общаемся, поэтому необходимо, чтобы компьютеры также имели возможность работать со словами и предложениями.

В Python мы храним что-то вроде слова, предложения или даже целого абзаца в виде строки. Строка — это последовательность символов. Последовательность может иметь любую длину и содержать любые буквы, цифры, символы и пробелы.

В этом уроке вы узнаете больше о строках и о том, как они обрабатываются в Python. Вы узнаете, как делать срезы строки, выбирать определенные символы из строк, искать в строках символы, выполнять итерацию по строкам и использовать строки в условных операторах.

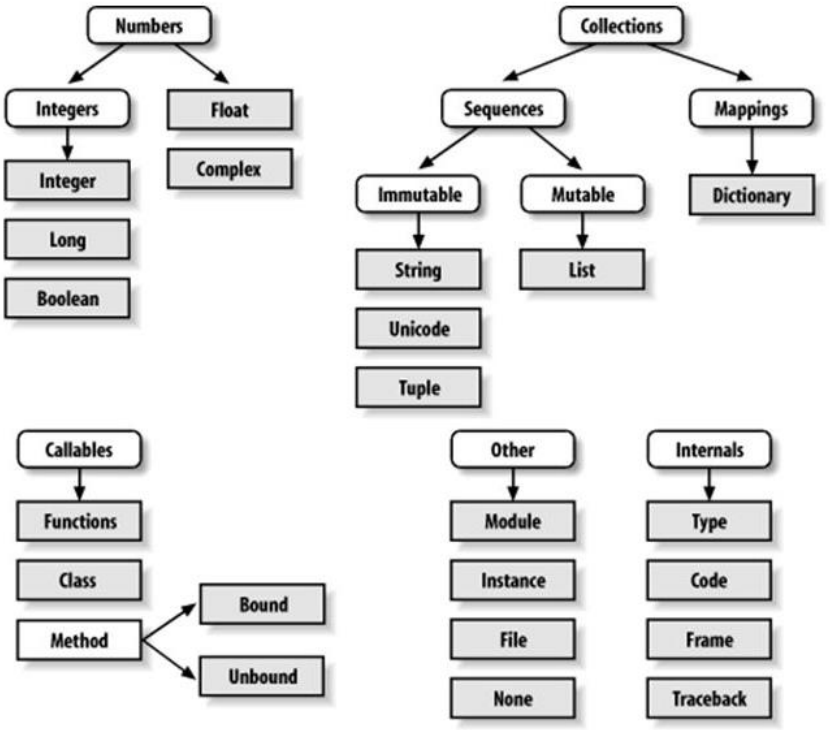
Задание

1. Сохраните ваше любимое слово в виде строки в переменной favour_word.

2. Выведите favour_word.

Последовательные типы данных

Рассмотрим существующие в python типы данных (рис):



В python, как и во всех популярных языках программирования, существуют последовательные типы данных, представленных на рисунке как sequences. Такие последовательные типы данных бывают изменяемые (можем вносить изменения прямо в созданный объект) и неизменяемые (при изменении необходимо каждый раз создавать новый объект). Последовательными типы являются частью типа коллекций.

Коллекциями в языке Python принято называть типы данных, в том или ином виде хранящие ссылки на ряд других объектов (не обязательно

однотипных). Последовательности отличаются тем, что обязательно имеют определенный порядок следования элементов, а так же набор индексов.

В рамках данного занятия будет рассмотрен такой последовательный тип, как строка.

Это все последовательность символов!

Строку можно рассматривать как последовательность символов.

Как и любая другая последовательность, каждый символ в строке имеет индекс. Рассмотрим строку

>>favorite_fruit = "яблоко"

Мы можем выбрать конкретные буквы из этой строки с помощью индекса. Давайте посмотрим на первую букву строки.

```
>>> favorite_fruit[1]
'б'
```

Обратите внимание, что буква в индексе 1 слова яблоко — это не я, а б. Это связано с тем, что индексы строки начинаются с 0. b находится в нулевом индексе, и мы могли бы выбрать его, используя:

```
>>> favorite_fruit[0]
'я'
```

Важно отметить, что индексы строк должны быть целыми числами. Если бы вы попытались выбрать нецелочисленный индекс, мы бы получили TypeError:

```
>>> favorite_fruit[1.5]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: string indices must be integers, not float
```

Как получить срез строки

Мы можем не только выбрать один символ из строки, мы можем выбрать фрагменты символов из строки. Это возможно сделать это с помощью следующего синтаксиса:

string_name[first_index:last_index]

Это называется срезом строки. Когда мы делаем срез, мы создаем новую строку, которая начинается (и включает) first_index и заканчивается (но исключает) last_index. Давайте посмотрим на несколько примеров. Вспомните вашу favorite_fruit:

```
>>> favorite_fruit
```

```
'яблоко'
```

Допустим, нам нужна новая строка, содержащая буквы ок. Мы могли бы сделать срез favorite_fruit следующим образом:

```
>>> favorite_fruit[3:5]
'ок'
```

Обратите внимание, что символ в первом индексе, я, ВКЛЮЧЕН, но символ в последнем индексе, о, ИСКЛЮЧЕН.

У нас также могут быть не указаны индексы. Если мы удалим первый индекс, срез начнется с начала строки, а если мы удалим второй индекс, срез продолжится до конца строки.

```
>>> favorite_fruit[:4]
'ябло'
>>> favorite_fruit[4:]
'ко'
```

Опять же, обратите внимание, как к из строки исключен из первого примера и включен во второй пример.

Задание

1. Вы программист, работаете в большой компании. В этой компании имя пользователя каждого сотрудника создается путем взятия первых пяти букв их фамилии.

Новый сотрудник, Виталий Красилов, начинает работу сегодня, и вам необходимо создать его учетную запись. Запишите его first_name и last_name.

Создайте переменную new_account, срезав первые пять букв его last_name.

2. Временные пароли для новых сотрудников также генерируются из их фамилий.

Создайте переменную с именем temp_password, создав фрагмент с третьей по шестую буквы его last_name.

Конкатенация (объединение строк)

Вы также можете объединить две существующие строки в новую строку. Рассмотрим следующие две строки.

```
>>fruit_prefix = "blue"
>>fruit_suffix = "berries"
```

Мы можем создать новую строку, объединив их вместе следующим образом:

```
>>>favorite_fruit = fruit_prefix + fruit_suffix
>>> favorite_fruit
'blueberries'
```

Обратите внимание, что здесь не добавлены пробелы. Вы должны вручную добавить пробелы при объединении строк, если хотите их включить.

Задание

1. Руководство Компании, в которой вы работаете, осознало, что их политика использования первых пяти букв фамилии сотрудника в качестве имени пользователя не идеальна, если у них несколько сотрудников с одинаковой фамилией.

   Напишите функцию с именем account_generator, которая принимает два входа, first_name и last_name, объединяет первые три буквы каждого из них, а затем возвращает новое имя учетной записи.

2. Проверьте свою функцию и сохраните результат ее выполнения в переменную new_account.

Измерение длины строки

Python имеет несколько встроенных функций для работы со строками. Одна из наиболее часто используемых этих функций - len (). len () возвращает количество символов в строке

```
>>> favorite_fruit = "blueberry"
>>> len(favorite_fruit)
9
```

Если вы измеряете длину предложения, то учитываются и пробелы.

```
>>> fruit_sentence = "I love blueberries"
>>> len(fruit_sentence)
18
```

Теперь рассмотрим следующий код:

```
>>> length = len(favorite_fruit)
>>> favorite_fruit[length]
```

Но этот код сгенерирует IndexError, потому что, помните, индексы начинаются с 0, поэтому последний символ в строке имеет индекс len (string_name) - 1.

```
>>> favorite_fruit[length-1]
'y'
```

Вы также можете вывести последние несколько символов строки с помощью len ():

```
>>> favorite_fruit[length-4:]
'erry'
```

Задание

1. Руководство компании снова хочет обновить способ генерации временных паролей для новых сотрудников.

Напишите функцию с именем password_generator, которая принимает два входа, first_name и last_name, а затем объединяет последние три буквы каждого из них и возвращает их в виде строки.

2. Протестируйте свою функцию и сохраните их в переменной temp_password.

Отрицательные индексы

В предыдущем упражнении мы использовали len (), чтобы получить фрагмент символов в конце строки.

Есть более простой способ это сделать - использовать отрицательные индексы! Отрицательные индексы отсчитываются в обратном порядке от конца строки, поэтому string_name [-1] — это последний символ строки, string_name [-2] - второй последний символ строки и т. д.

Вот некоторые примеры:

```
>>> favorite_fruit = 'blueberry`
>>> favorite_fruit[-1]
'y'
>>> favorite_fruit[-2]
'r'
>>> favorite_fruit[-3:]
'rry'
```

Обратите внимание, что мы можем вывести последние три символа слова blueberry, имея начальный индекс -3 и опуская конечный индекс.

Задание

1. Используйте отрицательные индексы, чтобы найти предпоследний символ в company_motto. Сохраните это в переменной second_to_last.

company_motto= «Мечты сбываются»

2. Используйте отрицательные индексы, чтобы создать фрагмент из последних 4 символов в company_motto. Сохраните это в переменной final_word.

Строки неизменяемы (иммутабельны)

До сих пор в этом уроке мы выбирали символы из строк, формировали срезы строки и объединяли строки. Каждый раз, когда мы выполняем одну из этих операций, мы создаем совершенно новую строку.

Это потому, что строки неизменяемы. Это означает, что мы не можем изменить строку после ее создания. Мы можем использовать строку для создания других строк, но мы не можем изменить саму строку.

Это свойство обычно называется неизменяемостью (иммутабельностью). Типы данных, которые являются изменяемыми, могут быть изменены, а типы данных, такие как строки, которые являются неизменяемыми, не могут быть изменены.

Задание

1. Последним сотрудником нашей компании стал парень по имени Роб Дейли. К сожалению, отдел кадров, похоже, допустил небольшую опечатку и прислал неправильное имя first_name.

Попробуйте изменить первый символ first_name, запустив

first_name[0] = "P"

2. Строки неизменяемы, поэтому мы не можем изменить отдельный символ. Хорошо, это не проблема, мы все еще можем это исправить!

Объедините строку «P» с фрагментом first_name, который включает все, кроме первого символа, и сохраните его в новой строке fixed_first_name.

Escape-последовательности

Иногда при работе со строками необходимо включить символы, которые уже имеют особое значение в Python. Например, скажем, создаем строку

>>favorite_fruit_conversation = "He said, "blueberries are my favorite!""

Мы случайно закончили строку раньше, чем захотим, включив символ ". Это можно сделать, введя escape-символы. Добавив обратную косую черту перед специальным символом, который мы хотим экранировать, \", мы можем включить его в строку.

>>favorite_fruit_conversation = "He said, \"blueberries are my favorite!\""

Задание

1. Когда Роб Дейли настраивал свою учетную запись, он установил свой пароль:

```
theycallme"crazy"91
```

Его пароль вызывал некоторые ошибки в системе из-за отметок ". Перепишите его пароль, используя escape-символы, и сохраните его с переменным паролем.

Методы форматирования строк

Вступление

Python поставляется со встроенными строковыми методами, которые позволяют очень быстро и эффективно выполнять сложные задачи со строками. Эти строковые методы позволяют изменять регистр строки, разбивать строку на множество меньших строк, объединять множество маленьких строк в большую строку и позволяют аккуратно комбинировать изменяющиеся переменные со строковыми выводами.

На предыдущем уроке вы работали с функцией len (), которая определяла количество символов в строке. Это, хотя и похоже, НЕ был строковым методом. Все строковые методы имеют одинаковый синтаксис:

```
py string_name.string_method(arguments)
```

Методы форматирования

Есть три строковых метода, которые могут изменить регистр строки. Это .lower (), .upper () и .title ().

• .lower () возвращает строку, содержащую все символы нижнего регистра.

• .upper () возвращает строку, содержащую все символы верхнего регистра.

• .title () возвращает строку в регистре заголовка, что означает, что первая буква каждого слова пишется с заглавной буквы.

Вот пример использования .lower () в действии:

```
>>> favorite_song = 'SmOoTH'
>>> favorite_song_lowercase = favorite_song.lower()
>>> favorite_song_lowercase
'smooth'
```

Каждый символ был изменен на нижний регистр! Важно помнить, что строковые методы могут создавать только новые строки, они не изменяют исходную строку.

```
>>> favorite_song
'SmOoTH'
```

Видите, все то же самое! Эти строковые методы отлично подходят для очистки пользовательского ввода и стандартизации форматирования ваших строк.

Задание

1. Вы программист, работающий в организации, которая пытается оцифровать и хранить стихи под названием «Preserve the Verse».

Вам дали две строки, название стихотворения и его автора, и попросили их немного переформатировать, чтобы они соответствовали правилам базы данных организации.

Сделайте poem_title заголовком и сохраните его в poem_title_fixed.

```
poem_title = "spring storm"
poem_author = "William Carlos Williams"
```

2. Выведите poem_title и poem_title_fixed.

Заключение

Из этого урока вы узнали:

• Строка — это список символов (Последовательный тип данных).

• Символ может быть выбран из строки с помощью его индекса имя_строки [index]. Эти индексы начинаются с 0.

• «Срез» можно выбрать из строки. Они могут находиться между двумя индексами или могут быть открытыми, выбирая всю строку из точки.

• Строки можно объединять в строки большего размера.

• len () может использоваться для определения количества символов в строке.