**Введение в работу с файлами.**

**Шаг 1. Чтение данных из текстовых файлов**

**ЧТЕНИЕ ДАННЫХ ИЗ ФАЙЛОВ**

В реальных задачах данные мы получаем из внешних источников. Представим, что вам надо прочитать некий файл data.txt, , в котором собрана аналитика по некой рекламной кампании.

В этой работе мы научимся читать содержимое файлов и записывать их, например, в список. В течение занятия мы будем работать с файлом data.txt (и его аналогом в Excel data.xlsx).

**Начнем с малого!** Для удобства в различных задачах мы разбили исходный файл на небольшие производные файлы с ограниченным объемом данных.

Для начала рассмотрим файл, в котором содержатся только один столбец файла data.txt с ID пользователей (столбец user\_id).

Файл – user\_ids.txt

**Шаг 1.**

Для работы с этим файлом скачайте его себе.

Так выглядит столбец с ID  пользователей:

**1010**

**1036**

**1041**

**1041**

**1042**

**Шаг 2. Прочитаем файл**

Для чтения файлов используется следующая конструкция:

**with** open('user\_ids.txt', 'r') **as** f:

**for** line **in** f:

В ней указываем имя файла (user\_ids.txt), параметр чтения ('r') и обозначаем файл как f.

Далее в цикле "for line in f" проходим по каждой строке файла, записывая ее содержимое в переменную line. Выведем содержимое файла на экран:

**with** open('user\_ids.txt', 'r') **as** f:

**for** line **in** f:

**print**(line)

Какое значение будет выведено последним на экран?



**Шаг 2. Чтение содержимого файла в лист**

 Значения user\_id почему-то выводятся на экран с лишними «пробелами» между строк. Это происходит из-за того, что в текстовых файлах в конце строки всегда стоит служебный символ переноса строки (обозначается как \n и скрывается от пользователя в текстовых редакторах). Чтобы этого избежать, можно воспользоваться методом strip(). Он убирает с начала и конца строки переносы строк.

Продолжаем работать с файлом user\_ids.txt

**with** open('user\_ids.txt', 'r') **as** f:

**for** line **in** f:

**print**(line.strip())

Чтобы проводить операции с набором ID пользователей, давайте записывать каждый очередной ID в лист user\_ids:

user\_ids = []

**with** open( 'user\_ids.txt', 'r' ) **as** f:

**for** line **in** f:

user\_id = line.strip()

user\_ids.append(user\_id)

Упражнение

Выведите на экран первые 5 элементов листа user\_ids. Какой ID будет в этом наборе последним?



**Шаг 3. Подсчет уникального числа элементов**

Упражнение

На прошлом шаге мы получили лист список ID пользователей из файла.

Сколько всего элементов в этом листе? Напоминание: длину листа можно посчитать с помощью функции len



**ПОДСЧЕТ ЧИСЛА УНИКАЛЬНЫХ ЭЛЕМЕНТОВ**

В листе user\_ids некоторые ID пользователей повторяются, т. к. эти пользователи делали по несколько покупок. Представим, что нам надо понять общее число уникальных пользователей.

Для подсчета уникальных ID этого листа достаточно воспользоваться функцией set. Эта функция преобразует список в так называемый тип set, содержащий уникальный набор элементов листа. Рассмотрим простой пример.

Пусть дан лист из чисел (из нулей и единиц):

list\_of\_zeros\_and\_ones = [**0**, **1**, **0**, **1**, **0**, **1**]

Тогда функция set вернет нам список уникальных элементов листа, т. е. 0 и 1:

**print**( set(list\_of\_zeros\_and\_ones) )

Применим этот функцию к нашему набору ID пользователей. Т. е. получим набор уникальных ID пользователей из листа user\_ids:

unique\_ids = set(user\_ids)

Теперь можем посчитать количество уникальных пользователей в файле:

**print**( 'Уникальных ID в файле {}'.format( len(unique\_ids) ) )

Упражнение

Какое количество уникальных ID в файле user\_ids.txt?

**Файлы с заголовками и нескольким столбцами**

**Шаг 1. Чтение файлов с заголовками**

Часто в файле присутствует заголовок, который нам не надо учитывать в расчетах:

with open( 'user\_ids\_headers.txt', 'r' ) as f:  
      for line in f:  
            print( line.strip( ) )

На выходе:  
user\_id  
1010  
1036  
1041  
1041  
1042  
1042  
1042  
1042  
1042  
1042

Чем может помешать заголовок?

Если мы будем учитывать заголовок вместе с остальными ID пользователей, то это внесет ошибку во многие наши подсчеты: количество уникальных пользователей в файле, среднее количество заказов на пользователя и другие. Давайте рассмотрим способ, который позволит не учитывать заголовок в расчетах необходимых метрик.

Для этого упражнения используем файл с заголовком:

Файл – user\_ids\_header.txt

Раньше мы всегда читали данные с самой первой строки файла:

**with** open('user\_ids\_headers.txt', 'r') **as** f:

**for** line **in** f:

**print**(line.strip())

Упражнение

Какая строчка появится первой при чтении файла user\_ids\_headers.txt описанным выше способом?



**Шаг 2. Как избежать первой строки**

Для того, чтобы не учитывать первую строчку, можно использовать нехитрый прием. Он пригодится вам во многих рабочих задачах:

* 1. Заведем переменную Started, которая перед началом прохождения файла равна True.
  2. В первом шаге цикла переводим ее в значение False и более ничего не делаем. Тем самым мы пропускаем первую строчку файла (т. е. заголовок).
  3. Во втором и следующих шагах цикла значение Started уже всегда будет False. И в этих шагах мы уже выполняем необходимые нам операции со строками:

Started = True

**with** open('user\_ids\_headers.txt', 'r') **as** f:

**for** line **in** f:

# запись "if Started" эквивалентна "if Started == True"

**if** Started:

# эта строчка выполняется один раз только на первом шаге цикла

Started = False

**else**:

# эту строчку выполняем со второго и во всех следующих шагах цикла

**print**(line.strip())

**Шаг 3. Чтение данных из файлов с несколькими столбцами**

Сейчас мы работали только с одним столбцом, но как читать данные из файла, в котором несколько столбцов, т. е. целая таблица?

Рассмотрим работу с файлом data\_3\_columns.txt, в котором целых три столбца! В нем у нас есть некоторая статистика по источникам и стоимости заказа.

**with** open('data\_3\_columns.txt', 'r') **as** f:

**for** line **in** f:

**print**(line.strip())

Результат:

seo google **20**,**20**

sem yandex **15**,**60**

email promo **13**,**20**

sem yandex **9**,**80**

sem google **14**,**80**

Для разделения строки line на столбцы используется метод split. В качестве аргумента в нее подставляется разделитель столбцов. Например, в нашем файле разделителем является знак табуляции (красным выделено изменение в коде):

**with** open( 'data\_3\_columns.txt', 'r' ) **as** f:

**for** line **in** f:

**line = line.strip().split('\t')**

**print**( line )

['seo', 'google', '20,20']

['sem', 'yandex', '15,60']

['email', 'promo', '13,20']

['sem', 'yandex', '9,80']

['sem', 'google', '14,80']

Теперь в переменной line в каждом шаге цикла будет записан лист с элементами столбцов.

**Шаг 4. Разделяем столбцы**

В предыдущем примере видна проблема: третий столбец должен быть числом (стоимость заказа), но он выводится в кавычках, еще и через запятую (следствие копирования данных из Excel).

**Запомните: во всех основных языках программирования разделителем в числе всегда является точка!**

Хорошо, что Python позволяет нам привести к единому виду большие объемы данных. Чтобы исправить запятые на точки, можем использовать метод replace. Первым аргументом ставим запятую (элемент, который ищем и заменяем в строке). Вторым — точку (элемент, на который заменяем).

Мы уже поняли по данным, что первая и вторая колонка показывают нам промежуточный и основной источник, с которого пришел пользователь, а третий обозначает стоимость заказа.

Давайте заодно заменим элементы столбца понятными названиями переменных (medium, source, amount\_paid), которые отражают, какие данные у нас в каком столбце.

**with** open('data\_3\_columns.txt', 'r') **as** f:

**for** line **in** f:

line = line.strip().split('**\t**')

medium = line[**0**]

source = line[**1**]

amount\_paid = line[**2**].replace(',', '.')

**print**(line)

**print**(source, medium, amount\_paid)

Результат:

['seo', 'google', '20,20']

google seo **20.2**

['sem', 'yandex', '15,60']

yandex sem **15.6**

['email', 'promo', '13,20']

promo email **13.2**

['sem', 'yandex', '9,80']

yandex sem **9.8**

['sem', 'google', '14,80']

google sem **14.8**

**Шаг 5. Получаем число из строки**

Мы заменили запятые на точки, но переменная amount\_paid все еще типа string:

type(amount\_paid)

Результат:

str

Это не даст нам производить с ней вычисления, как с числом. Например, мы не сможем подсчитать сумму заказов или найти самый дорогой. Переведем стоимость заказа из строк в тип float с помощью одноименной функции float:

**with** open('data\_3\_columns.txt', 'r') **as** f:

**for** line **in** f:

line = line.strip().split('**\t**')

medium = line[**0**]

source = line[**1**]

**amount\_paid = float( line[2].replace(',', '.') )**

**print**( source, medium, amount\_paid )

Упражнение

Какое значение amount\_paid будет в последней строчке файла? Не забывайте, что разделителем должна быть точка.



**Шаг 6. Подсчет суммы в файлах**

В прошлом блоке нам удалось привести третий столбец в числовой вид. Давайте посчитаем сумму этого столбца, добавляя в переменную total\_sum в цикле стоимость каждого следующего заказа. Приступим?

При работе используем все тот же файл data\_3\_columns.txt.

Для понимания процесса на каждом шаге будем выводить на экран сумму total\_sum накопленным итогом. Берем код из прошлого шага и добавляем необходимые строки:

total\_sum = **0**

**with** open('data\_3\_columns.txt', 'r') **as** f:

**for** line **in** f:

line = line.strip().split('**\t**')

amount\_paid = float(line[**2**].replace(',', '.'))

**total\_sum += amount\_paid**

**print('Текущая сумма расходов: {:.2f}'.format( total\_sum ))**

Результат:

Текущая сумма расходов: **20.20**

Текущая сумма расходов: **35.80**

Текущая сумма расходов: **49.00**

Текущая сумма расходов: **58.80**

Текущая сумма расходов: **73.60**

...

Упражнение

Какое значение суммы получаем в последнем примере?

**Шаг 7. Подсчет суммы с фильтром**

Давайте посчитаем сумму не для всех строк, а при следующем условии: нам необходимо брать только те строки, у которых источник source равен google. Для этого в цикл надо добавить всего одну строчку:

total\_sum = **0**

**with** open( 'data\_3\_columns.txt', 'r' ) **as** f:

**for** line **in** f:

line = line.strip().split('**\t**')

medium = line[**0**]

source = line[**1**]

amount\_paid = float( line[**2**].replace(',', '.') )

**if source == 'google':**

total\_sum += amount\_paid

**print**( 'Текущая сумма расходов google: {:.2f}'.format( total\_sum ) )

Результат:

Текущая сумма расходов google: **20.20**

Текущая сумма расходов google: **35.00**

Текущая сумма расходов google: **49.40**

Текущая сумма расходов google: **63.40**

Текущая сумма расходов google: **86.00**

...

Текущая сумма расходов google: **1318.80**

Упражнение

Какова сумма amount\_paid для строк, у которых source == 'yandex' и medium == 'seo'? Напоминание: одновременные условия в операторе if перечисляются с помощью and.