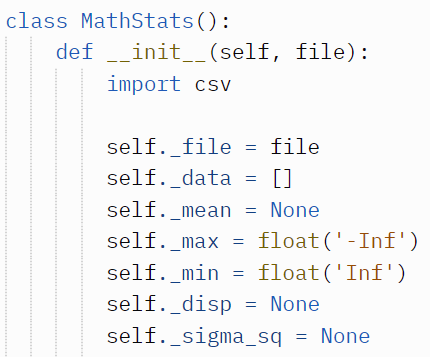
**Инвариантная самостоятельная работа**

1.1. Разработка скрипта, вычисляющего статистические показатели (среднее значение, дисперсия, среднее квадратичное отклонение) для данных, считанных из CSV-файла.

Процесс разработки скрипта

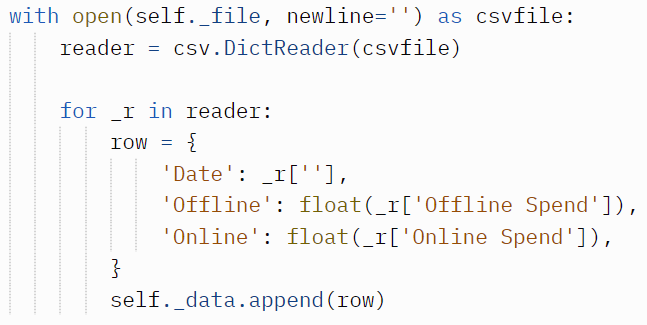
Файл mathstats.py содержит формулы для вычисления статистических показателей. Данный модуль создан в виде класса MathStats



Инициализируем класс. Сначала сделаем импорт библиотеки csv, чтобы можно было считать файл формата .csv. Затем инициализируем переменные с модификаторами доступа protected (одно подчёркивание).

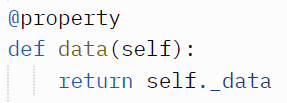
Список переменных

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Значение | Смысл | Комментарий |
| file | file | Имя файла | Задаётся при инициализации |
| data | [] | Считанные данные | Пустой список |
| mean | None | Среднее значение | Пустое значение |
| max | float('-Inf') | Максимальное значение | -бесконечность |
| min | float('Inf') | Минимальное значение | +бесконечность |
| disp | None | Дисперсия | Пустое значение |
| sigma\_sq | None | Среднее квадратичное отклонение | Пустое значение |

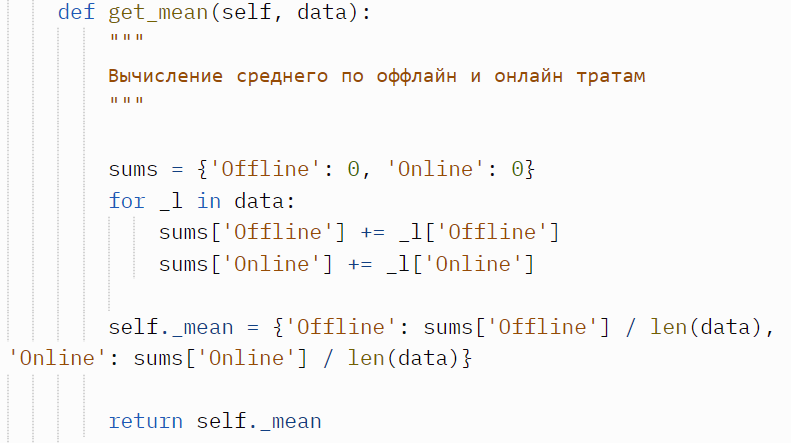


Затем считаем файл. С помощью функции open и ключевых слов “with”, “as” сохраним файловый объект в переменной csvfile. Потом с помощью функции DictReader библиотеки csv считаем файловый объект и передадим его в переменную reader.

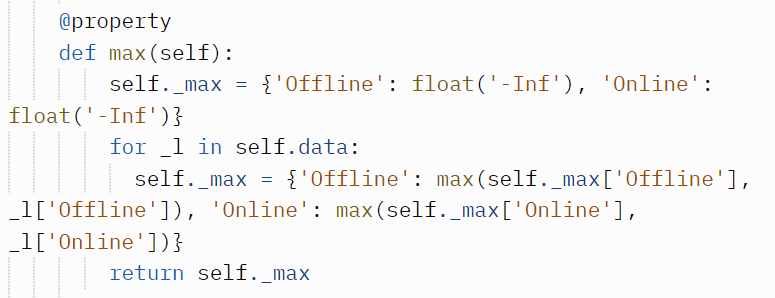
После этого нужно сохранить полученные данные в удобной форме. Для этого в цикле будем создавать словари со считанными данными и сохранять их в список data. Словари содержат три ключа – ‘Date’, ‘Offline’, ‘Online’, что совпадает со структурой файла.



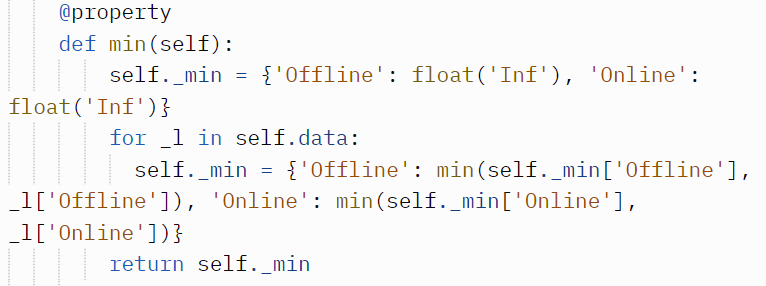
Здесь мы задали свойство класса с именем data, которое возвращает данные из считанного файла.



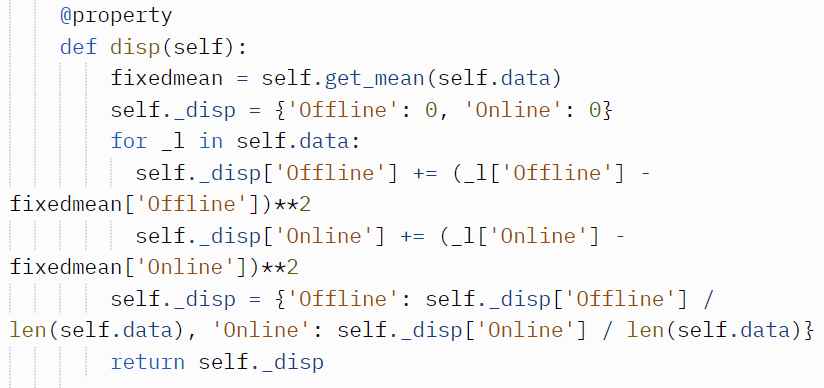
Затем мы создали метод get\_mean, позволяющий посчитать средние значения. Для этого сохраним суммы в словаре с ключами ‘Offline’ и ‘Online’. Затем значения будем суммировать с данными в цикле. Потом рассчитаем среднее значение, разделив полученные суммы на количество элементов и запишем результат в переменную mean. В конечном итоге метод возвращает эту переменную.



После этого в качестве свойства создадим метод max, который бы искал максимальные значения в данных. Запишем в переменную max словарь с самыми минимальными значениями – минус бесконечность. Потом в цикле будем сравнивать каждое значение столбцов с записанным максимальным значением с помощью метода max(). В случае нахождения максимумов будем переписывать переменную. После завершения цикла вернём переменную max с уже готовыми максимальными значениями, записанными в словарь.



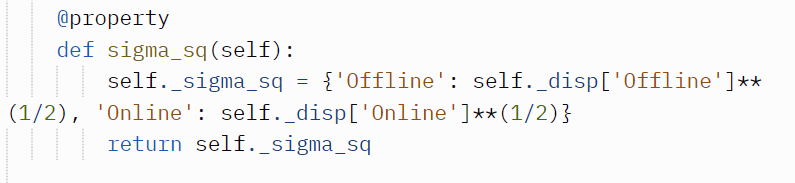
Аналогично найдём и минимальные значения с помощью метода min. Алгоритм такой же, но вместо методов max() будем использовать методы min(), а первое сравнение произойдёт с наибольшим значением – плюс бесконечность.



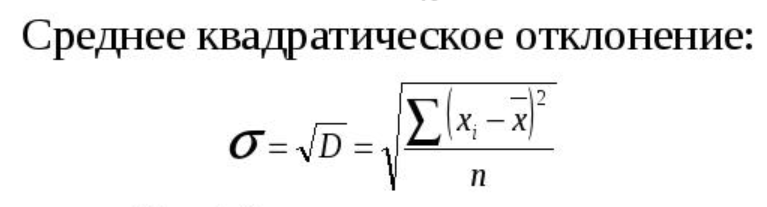
Следующий алгоритм реализует вычисление дисперсии по формуле



Запишем в переменную fixedmean словарь из полученных средних значений. Реализуем в цикле подсчёт сумм, которые нужны по формуле – квадрат разности элемента и среднего значения. В конце разделим полученные суммы на количество элементов. Таким образом с помощью алгоритма удалось посчитать дисперсии. Вернём переменную disp, которая содержит этот результат.



Здесь мы посчитаем среднее квадратичное отклонение, которое считается по формуле



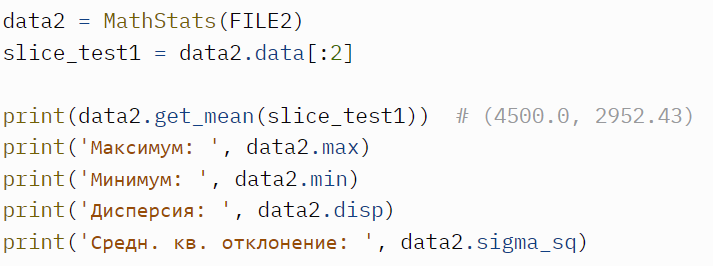
Так как дисперсии мы уже посчитали, просто найдём квадратный корень из результатов прошлого алгоритма и вернём их в переменной sigma\_sq.

Затем опробуем полученный класс в файле main.py.



Импортируем класс из файла mathstats.py.





Затем инициализируем класс с помощью переменной FILE2, которая содержит имя нужного нам файла. С помощью print выведем результаты работы методов класса.

