Мини-отчёт

Рудаков Максим, M3137

11 октября 2023 г.

1 Вступление

Лабараторная написана на Питоне (Python 3.11.5) с попытками оформить ООП. Каждый вид числа соответсвует своему классу: FixedPoint, Single and Half. Была идея сделать надкласс AbstractFloat, но до исполнения не дошло. Каждый класс вынесен в свой файлик (было сильно неудобно, решил разделить). В файле main.py происходит считываение чисел, минимальная логика того, что с числами нужно сделать, а также обрабатываются некоторые ошибки ввода (не тот тип окрглления или формат).

В файле test.py я сделал тестирование. Числа с фикс. точкой подвергаются полному тестированию, сравниваясь с Питоновскими float. Проверяется вывод и все операции. Числа с плавающей точкой так проверить не удалось, т.к. Питоновские float'ы сложно привести к требуемому формату. Тесты проверяют лишь то, что программа не падает на всевозможных тестах.

Используются 5 основных методов (перегруженные операторы):

__str__ - возвращает строковое представление числа таким, каким оно должно быть выведено в консоль.

__add__, __sub__, __mul__, __truediv__ - соотвутсвуеют операциям +, -, * и /.

2 Числа с фиксированной точкой

В полях класса хранятся числа A и B - количество знаков до и после запятой. Числа хранятся в строках. Длина строки равна A + B. Операции выполняются в интах.

__str__ переводит число в десятичную запись и округляет. Перевод реализован по частям: сначала после точки, потом перед точкой. Эти числа суммируются и получается итоговое челое число без точки. Потом функцией my_round я округляю его так, чтобы после точки вместо B символов осталось только 3. В возвращенную строку втсавляю точку на нужное место.

Операция + реализует сложение чисел и переполнение.

Операция - возвращает X + -Y.

Операция * реализует умножение чисел, округление и переполнение.

Операция / реализует деление чисел, округление и переполнение.

Также для удобства я перегрузил и другие методы, например, сравнение, т.к. при делении нужно сравнивать число с 0 и __invert__, который инфертирует все биты числа (X).

3 Single и Half

В полях класса хранятся константы для соответсвующего класса. Собсвтенно, только этим Single и Half различаются. Half является подклассcом класса Single. Числа хранятся по частям: знак, экспонента и мантисса. При получении числа в формате hex или bin оно парсится на части или заменяется на константу типа nan и inf.

__str__ тщательно проверяет на то, является ли число нулём, inf или nan. Если нет, просто переводит мантиссу в 16-ричную запись и форматирует число. Учтены денормализованные числа.

В начале каждой операции я проверяю число на nan, inf и ноль, учитываю специальные случаи.

Операция + переводит числа в огромные инты (слава длинной арифметике), где 1 равен 0x00000001. В таком виде числа складываются, запоминается знак и берётся модуль. Полученное число округляется и нормализуется, вычисляется экспонента.

Операция - возвращает X + -Y.

Операция * вычисляет знак, перемножает мантиссы как числа с фикс. точкой, рассчитывает экспоненту (складывает и прибавляет переход через бит мантиссы). В конце дедлайна я заметил, что не учёл некоторые случаи при умножении и делении. Это связано с переходом через бит при округлении. Пофиксить это я не успел, но сделал так, чтобы программа из-за этого не падала.

Операция / вычисляет знак, делит мантиссы как числа с фикс. точкой, округляет (тут та же проблема с переходом). Вычитаением вычисляется экспонента.

Как и с фиксированной точкой, для удобства я перегрузил методы __neg__, который определяет поведение -Y, и некоторые другие.