13.4 Задания для самостоятельной работы

Домашнее задание

1. Реализовать класс Context, который будет принимать обязательный аргумент expression - это строка с математическим выражением. У класса также будет метод evaluate, в котором нужно рекурсивно разобрать expression на составляющие и вернуть переформатированное выражение, где числа будут обернуты в `[x]`, а операторы с операндами в `(x*y)`:

```
ctx = Context("3*2 + 7^3")
print(ctx.evaluate())
# (([3] * [2]) + ([7] ^ [3]))
```

2. Расширить функционал, добавив класс Number, с обязательным аргументом - value, который будет представлять любое число в выражениях, а также классы Add(сумма), Sub(разность), Mul(умножение), Div(деление), Pow(степень) - которые представляют все возможные операции, и принимают два обязательных аргумента left - левый операнд, right - правый операнд, операндом может быть любой объект выше перечисленных классов. У всех упомянутых классов должен быть метод interpret(), который будет возвращать вычисленное значение типа float(к примеру "4".interpret() -> 4, "3*2".interpret() -> 6). Все исключительные ситуации отловить try/except блоками. Также нужно изменить реализацию метода Context.evaluate() так, чтобы он возвращал не строку, а объект одного из выше указанных классов. Обязательно указывать аннотации и пользоваться только описанными классами.

```
ctx = Context("3*2 + 7^3")
print(ctx.evaluate())
# 349
```

3. Ввести уровень абстракции в текущую реализацию добавив классы TerminalExpression с обязательным аргументом value: float, представляющий значения, которые нельзя разбить на части - это числа, класс Number должен быть унаследован от этого класса, NonTerminalExpression с двумя обязательными аргументами left - левый операнд, right - правый операнд, который будет представлять любое комплексное выражение, т.е все классы операций Add, ..., Pow должны быть унаследованы от этого класса. И последним мы добавим класс AbstractExpression, который будет представлять любое выражение в программе, и который будет иметь только один абстрактный метод(смотреть @abc.abstractmethod) - interpret() возвращающий вычисленное значение float. Классы TerminalExpression и NonTerminalExpression должны быть унаследованы от него.

4. *Добавить реализацию последовательного вывода операций.

```
python calc.py
? 10 * 2 + 3 ** 2
= 10 * 2 + 9
= 20 + 9
= 29
```

```
? 2.34 ** 2 - 6 / 0
= 5.4756 - 6 / 0
= Нельзя делить на 0
```