4.5 Полиморфизм и абстрактные методы 3 out of 11 steps passed 2 out of 25 points received

Подвиг 5. Ранее вы уже создавали классы валидации в виде иерархии базового класса Validator и дочерних:

StringValidator
IntegerValidator
FloatValidator

для валидации (проверки) корректности данных. Повторим этот функционал с некоторыми изменениями.

Итак, вначале нужно объявить базовый класс **Validator**, в котором должен отсутствовать инициализатор (магический метод __init__) и объявлен метод со следующей сигнатурой:

def _is_valid(self, data): ...

По идее, этот метод возвращает булево значение True, если данные (data) корректны с точки зрения валидатора, и False - в противном случае. Но в базовом классе Validator он должен генерировать исключение командой:

```
raise NotImplementedError('в классе не переопределен метод _is_valid')
```

Затем, нужно объявить дочерний класс **FloatValidator** для валидации вещественных чисел. Объекты этого класса создаются командой:

```
float_validator = FloatValidator(min_value, max_value)
```

где min_value - минимально допустимое значение; max_value - максимально допустимое значение.

Пользоваться объектами класса FloatValidator предполагается следующим образом:

```
res = float_validator(value)
```

где value - проверяемое значение (должно быть вещественным и находиться в диапазоне [min_value; max_value]). Данный валидатор должен возвращать True, если значение value проходит проверку, и False - в противном случае.

Пример использования классов (эти строчки писать не нужно):

```
float_validator = FloatValidator(0, 10.5)
res_1 = float_validator(1)  # False (целое число, а не вещественное)
res_2 = float_validator(1.0)  # True
res_3 = float_validator(-1.0)  # False (выход за диапазон [0; 10.5])
```

P.S. В программе требуется объявить только классы. На экран выводить ничего не нужно.

To solve this problem please visit
https://stepik.org/lesson/701999/step/6