Лабораторна робота 5

Метакласи та метапрограмування

Виконала:
студентка групи МІТ-31
Півторак Каріна
Варіант -6

Мета: розглянути концепцію метапрограмування у Пайтон. Ознайомитися з поняттям класу в Пайтон, навчитися створювати та використовувати метакласи

Завдання:

6. **ЗАВДАННЯ 1 (6 балів у разі повного і правильного виконання і захисту роботи). Написати метаклас згідно свого варіанту завдання (таблиця 5.1) та продемонструвати його використання.**

6	Метаклас, який записує створені ним класи та кількість наявних у них методів

Репозиторій: https://github.com/KariSpace/python-oop-labs.git

Хід виконання лабораторної роботи

Створюємо клас під назвою `MyMetaclass.` Цей клас є метакласом, оскільки, ми передаємо у нього функцію `type.`
Метаклас - конструкція, що може створювати класи.
`type -` метаклас, що у пайтон використовується для створення функцій.
можна сказати, що клас `MyMetaclass` наслідує метаклас `type`

Найчастіше метакласи використовуються в ролі віртуального конструктора. Щоб створити екземпляр класу, потрібно спочатку викликати цей самий клас. Точно так чинить і Python: для створення нового класу викликає метаклас.

```
clases = {}

class MyMetaclass(type):
    def __new__(cls, clsname, superclasses, attributedict):
      #   print("clsname: ", clsname)
      #   print("superclasses: ", superclasses)
      #   print("attributedict: ", attributedict)
         clases[clsname] = {
           "methods" : [*attributedict],
           "methods_amount" : len([*attributedict])
         }
```

```
        return type.__new__(cls, clsname, superclasses, attributedict)
```

Метаклас `MyMetaclass` створює клас за допомогою `__new__` у який ми передаємо `(cls, clsname, superclasses, attributedict)`

Згідно із здавданням, я модифікувала стандартну логіку створення класу, додавши конструкцію

```
        clases[clsname] = {
          "methods" : [*attributedict],
          "methods_amount" : len([*attributedict])
        }
```

яка при створенні класу додає у словник clases, що був ініціалізований вище, додає ключ `clsname` значенням якого є словник із значеннями `methods` (список методів) та `methods_amount` (кількість методів класу).

Подивимось як це працює:
додамо два класи, що були створені у попередніх роботах:

```
class Phone(metaclass=MyMetaclass):

    basic_color = "Grey"
    def __init__(self, company, model, price, hue):
        self.company = company
        self._model = model       # 'Please' don't access directly
        self.price = price
        self.color = hue + self.__class__.basic_color

    def __str__(self):
        """Return a descriptive string for this instance, invoked by print()
and str()"""
        return f'\nThis is a {self.color} {self.company} {self._model} phone'

    def discount_price(self, discount):
        return self.price*(1-discount)

    def set_model(self, _model):
        """Setter for instance variable model"""
        match = re.search(r'[^0-9]', _model)
        if(match):
            raise ValueError('Shall be non-numeric')
        else:
            self._model = _model

    def get_model(self):
        """Getter for instance variable model"""
        return self._model
```

```
class ATM(metaclass=MyMetaclass):

    def blockATM(self):
        self.blocked = True
        print("ATM was blocked")

    def unblockATM(self):
        self.blocked = False
        print("ATM unblocked")

    def callTheIncasator(self):
        self.money = self.volume
        self.unblockATM()
```

У цих класів є метаклас **MyMetaclass**, що створений вище.

Дописуємо вивід словника `clases.` Для виводу я використовую pp, це імпортована функція, що виводить словник у зручному форматуванні(схожому на jscon)

```
from beeprint import pp
...
... //тут код, описаний вище
...
pp(clases)
```

Запускаєм код, дивимось що вийшло

```
{
  'ATM': {
    'methods': ['__module__', '__qualname__', 'blockATM', 'unblockATM',
'callTheIncasator'],
    'methods_amount': 5,
  },
  'Phone': {
    'methods': ['__module__', '__qualname__', 'basic_color', '__init__',
'__str__', 'discount_price', 'set_model', 'get_model'],
    'methods_amount': 8,
  },
}
```

Висновок: Був створений кастомний метаклас, що створює класи, та записує їх методи