Урок 8. Принципы ООП. Полиморфизм и абстракция. Магические методы

ФИО_____

Тест:

1. *Как сделать свойство только для чтения?*

   _____

2. *Какой вариант описывает полиморфизм в ООП?*
   A) Один метод с десятью if B) Функция вызывает .area() у разных фигур C) Глобальная переменная D) Копирование кода
3. *Что ближе всего описывает абстракцию в ООП?*
   A) Спрятать детали и оставить важные методы B) Всюду print C) Делать класс из списка D) Глобальные переменные
4. *Что лучше отражает инкапсуляцию в Python?*
   A) @staticmethod B) _внутренний_атрибут и @property C) глобальные переменные D) print()

5. *Что выведет данная программа?*

```
1  class A:   1 usage
2      def f(self):   1 usage
3          return "A"
4  class B(A):  1 usage
5      def f(self):   1 usage
6          return "B"
7      def g(self):   1 usage
8          return super().f() + self.f()
9
10 print(B().g())
```

_____

Теория:

Принципы ООП:

- **Полиморфизм:** применение одних и тех же команды к объектам разных классов.

Полиморфизм в Python можно продемонстрировать с помощью методов и операторов, которые ведут себя по-разному в зависимости от типа объекта.

```
class Circle:
    def __init__(self, r):
        self.r = r
    def area(self):
        return 3.14159 * self.r * self.r

class Rectangle:
    def __init__(self, w, h):
        self.w, self.h = w, h
    def area(self):
        return self.w * self.h

def total_area(shapes) -> float:
    return sum(s.area() for s in shapes)     # «утиная типизация»: есть .area()
— значит годится

print(total_area([Circle(2), Rectangle(3,4)]))   # работает для обоих
```

Утиная типизация (англ. duck typing) — концепция в программировании, при которой корректность использования объекта определяется набором его методов и свойств, а не типом.

- **Абстракция:** применение класса-шаблона упрощенного до тех атрибутов и методов, которые нужны именно в этом коде.

Абстрактные классы выполняют функцию шаблона при создании более специализированных классов, определяя перечень методов, которые необходимо реализовать в этих классах. При определении подклассов от абстрактного класса, вы берете на себя обязанность реализации всех обязательных методов.

Определить **абстрактный класс** в Python возможно при помощи специализированного модуля abc и декоратора **@abstractmethod**. Принцип работы основан на наследовании от класса ABC и декорировании методов, что обязывает все производные классы реализовать указанные методы.

```
from abc import ABC, abstractmethod
class AbstractShape(ABC):

    @abstractmethod
    def area(self):
        pass
```

Магические методы:

Магические методы в Python, также известные как dunder-методы (от английского "double underscore"), представляют собой специальные методы, которые начинаются и заканчиваются двумя подчеркиваниями (например, __init__, __str__). Эти методы позволяют вам определять поведение ваших объектов при использовании встроенных функций и операторов.

Примеры магических методов:

Метод __init__ используется для инициализации объекта. Он вызывается автоматически при создании нового экземпляра класса. Этот метод позволяет вам задавать начальные значения атрибутов объекта и выполнять любую необходимую инициализацию.

```
class MyClass:
    def __init__(self, value):
        self.value = value
```

Метод __len__ позволяет определить длину объекта, возвращаемую функцией len(). Этот метод полезен для классов, которые представляют собой коллекции элементов.

```
class MyClass:
    def __len__(self):
        return len(self.value)
```

Еще примеры переопределения(перегрузки) магических методов сложения и умножения:

```
class Vector:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        return Vector(self.x + other.x, self.y + other.y)

    def __sub__(self, other):
        return Vector(self.x - other.x, self.y - other.y)

    def __str__(self):
        return f'Vector({self.x}, {self.y})'
v1 = Vector(2, 3)
v2 = Vector(1, 1)
print(v1 + v2)  # Вывод: Vector(3, 4)
print(v1 - v2)  # Вывод: Vector(1, 2)
```

Больше магических методов тут:
https://pyplanet.ru/article/all-magic-methods.html

Задачи:

1. **Полиморфизм + Абстракция:**
   - Создайте абстрактный класс Shape с методом area(). Реализуйте Circle(r), Rectangle(w,h), Triangle(a,h). Напишите функцию (вне классов) total_area(shapes), которая суммирует площади любых фигур.
   - Создайте абстрактный класс Animal с методом speak(). Реализуйте подклассы Cat, Dog и Bird. Функция (вне классов)

make_animals_speak() выводит, как разговаривают все перечисленные животные.

- Создайте абстрактный класс Transport с абстрактными методами move(distance_km: float) -> str и cost(distance_km: float) -> float. Реализуйте подклассы:
  - Car (параметры: расход liters_per_100km, цена топлива fuel_price),
  - Bike (без затрат: стоимость 0),
  - Bus (параметр: тариф fare_per_km).

  Напишите функцию (вне классов) trip_report(vehicles, distance_km), которая для каждого транспорта печатает строку из move() и итоговую стоимость по cost(), а затем выбирает и выводит самый дешёвый вариант на эту дистанцию.

2. Магические методы:

- Создайте класс Money с атрибутами amount и currency. Перезагрузи два магических метода __eq__ и __str__. При попытке сравнить два значения, программа должна сопоставлять не только то, что два числа равны, но и что у них равные валюты (currency). При вызове функции print формат печати следуюший: <amount> <currency> (например, 100 RUB).
- Создайте класс Team с атрибутами name и members (список имён). Перегрузите __contains__ и __str__. Оператор "Имя" in obj должен возвращать True/False в зависимости от наличия участника. При print(obj) выводите: Team <Name>: [имя1, имя2, ...] в исходном порядке.
- Создайте класс Playlist с атрибутом tracks (список строк). Перегрузите __len__, __getitem__ и __str__. len(obj) возвращает число треков; obj[i] — трек по индексу (поддержите и отрицательные индексы по аналогии со списком); при print(obj) выводите Playlist(N): [трек1, трек2, ...].
- Напишите класс, в котором перезагружаются все математические операции. Формат перезагрузки может быть любой, в качестве атрибутов *args и **kwargs.