Наследование

Наследование позволяет создавать новый класс на основе уже существующего класса. Наряду с инкапсуляцией и полиморфизмом, наследование является одной из важных концепций объектно-ориентированного программирования.

Ключевыми понятиями наследования являются **подкласс** и **суперкласс**.

Подкласс наследует от суперкласса все публичные атрибуты и методы. **Суперкласс** еще называется **базовым** (base class) или **родительским** (parent class), а подкласс - **производным** (derived class) или **дочерним** (child class).

Синтаксис для наследования классов выглядит следующим образом:

```
class подкласс (суперкласс):
методы_подкласса
```

Наследование – механизм, позволяющий запрограммировать отношение вида **«класс B является частным случаем класса A».** B этом случае класс A также называется базовым классом, а B – производным.

Чтобы наследовать класс B от класса A, нужно написать:

```
class A:
   pass
class B(A)
   pass
```

Наследование является способом переиспользования кода между классами без необходимости нарушения инкапсуляции. Это достигается за счет того, что производный класс может пользоваться атрибутами и методами базового класса (иными словами, производный класс наследует атрибуты и методы базового класса).

Если класс наследован от другого класса, проверка существования метода (или атрибута) осуществляется так:

−Сперва метод ищется в исходном (производном) классе.

−Если его там нет, он ищется в базовом классе.

−Предыдущие шаги повторяются до тех пор, пока метод не будет найден или пока процедура не дойдет до класса, который ни от кого не наследуется.

А это означает, что производному классу доступны не только собственные методы, но и методы базового класса. В этом случае говорят, что производный класс наследует методы базового класса.

Например, ранее был создан класс **Person**, который представляет человека. Предположим, нам необходим класс работника, который работает на некотором предприятии. Мы могли бы создать с нуля новый класс, к примеру, класс **Employee**. Однако он может иметь те же атрибуты и методы, что и класс **Person**, так как сотрудник – это человек. Поэтому нет смысла определять в классе **Employee** тот же функционал, что и в классе **Person**. И в этом случае лучше применить **наследование**.

```python
class Person:
    def __init__(self, name, age):
        self.__name = name  # устанавливаем имя
        self.__age = age  # устанавливаем возраст

    @property
    def age(self):
        return self.__age

    @age.setter
    def age(self, age):
        if age in range(1, 100):
            self.__age = age
        else:
            print("Недопустимый возраст")

    @property
    def name(self):
        return self.__name

    def display_info(self):
        print("Имя:", self.__name, "\tВозраст:", self.__age)


class Employee(Person):

    def details(self, company):
        # print(self.__name, "работает в компании", company) # так нельзя, self.__name - приватный атрибут
        print(self.name, "работает в компании", company)


tom = Employee("Дмитрий", 23)
tom.details("Востокнефтепровод")
tom.age = 33
tom.display_info()
```

```
Дмитрий работает в компании Востокнефтепровод
Имя: Дмитрий    Возраст: 33
```

Класс **Employee** полностью перенимает функционал класса **Person** и в дополнении к нему добавляет метод **details**().

Стоит обратить внимание, что для **Employee** доступны через ключевое слово **self** все методы и атрибуты класса **Person**, кроме закрытых атрибутов типа **__name** или **__age**.

При создании объекта **Employee** мы фактически используем конструктор класса **Person**. И кроме того, у этого объекта мы можем вызвать все методы класса **Person**.

Рассмотрим еще один пример наследования на примере иерархии геометрических фигур:

Для начала исследуем, как производные классы могут методами базового класса.

Реализуем в классе **Shape** метод **describe**, который будет печатать название собственного класса:

```
# -*- coding: utf-8 -*-

class Shape:
    def describe(self):
        print("Класс:{}".format(self.__class__.__name__))
#Атрибут __class__ содержит класс или тип объекта self
#Атрибут __name__ содержит строку, в которой написано название класса или типа

a=Shape()
a.describe()
Класс:Shape
```

Отнаследуем от **Shape** классы **Circle** и **Rectangle** и проверим метод **describe**:

```
# -*- coding: utf-8 -*-
from math import pi
class Shape:
    def describe(self):
        print("Класс:{}".format(self.__class__.__name__))
#Атрибут __class__ содержит класс или тип объекта self
#Атрибут __name__ содержит строку, в которой написано название класса или типа
class Circle(Shape):
    def __init__(self,radius):
        self.r=radius

    def area(self):
        return pi*pow(self.r,2)

class Rectangle(Shape):
    def __init__(self,a,b):
        self.a=a
        self.b=b

    def area(self):
        return self.a*self.b

shape=Shape()
shape.describe()

circle=Circle(2)
circle.describe()

rectangle=Rectangle(2,5)
rectangle.describe()

Класс:Shape
Класс:Circle
Класс:Rectangle
```

Теперь рассмотрим, как добавить в производный класс новый метод, которого нет в базовом классе.

Добавим в класс **Circle** метод **square**, который возвращает квадрат (в нашем случае – объект **Rectangle** с равными сторонами), который по площади равен площади исходного круга (задача о квадратуре круга).

```
# -*- coding: utf-8 -*-
from math import pi

class Shape:
    def describe(self):
        print("Класс:{}".format(self.__class__.__name__))
#Атрибут __class__ содержит класс или тип объекта self
#Атрибут __name__ содержит строку, в которой написано название класса или типа

class Circle(Shape):
    def  __init__(self,radius):
        self.r=radius

    def area(self):
        return pi*pow(self.r,2)

    def square(self):
        side=pow(pi,0.5)*self.r
        return Rectangle(side,side)

class Rectangle(Shape):
    def __init__(self,a,b):
        self.a=a
        self.b=b

    def area(self):
        return self.a*self.b


circle=Circle(1)
square=circle.square()
print ("Площадь круга: {}".format(circle.area()))
print ("Площадь квадрата: {}".format(square.area()))
print ("Радиус круга: {}".format(circle.r))
print ("Длина стороны квадрата: {}".format(square.a))

Площадь круга: 3.141592653589793
Площадь квадрата: 3.1415926535897927
Радиус круга: 1
Длина стороны квадрата: 1.7724538509055159
```

Задания для самостоятельной работы

Задание 1. **Курсы по программированию.**

На курсах по программированию все пользователи разбиты на три группы: администраторы, преподаватели и обучающиеся. Все пользователи могут решать задачи. Также администраторы могут редактировать задачи, а преподаватели могут проверять решения обучающихся. Также среди администраторов есть особенные «супер-администраторы», которые могут сделать администратором любого пользователя курсов по программированию. На основании предложенной предметной области спроектировать классы, используя механизм наследования. Продемонстрировать работу всех объявленных методов. Продемонстрировать вызов конструктора родительских классов при наследовании.

Задание 2. **Наследование и полиморфизм**

1. На основании предложенной предметной области спроектировать 3-4 класса, используя механизм наследования. Для каждого класса использовать отдельный модуль.
2. Предусмотреть у класса наличие полей, методов и свойств. Названия членов класса должны быть осмысленны и снабжены комментариями.
3. Продемонстрировать работу всех объявленных методов.
4. Продемонстрировать вызов конструктора родительского класса при наследовании.

Варианты заданий приведены в таблице 1.

Таблица 1.

Вариант 1	Написать программу, в которой описана иерархия классов: средство передвижения (велосипед, автомобиль, грузовик). Базовый класс должен иметь поля для хранения средней скорости, названия модели, числа пассажиров, а также методы получения потребления топлива для данного расстояния и вычисления времени движения на заданное расстояние. Продемонстрировать работу всех методов классов, предоставив пользователю выбор типа объекта для демонстрации.
Вариант 2	Написать программу, в которой описана иерархия классов: человек («дошкольник», «школьник», «студент», «работающий»). Базовый класс должен иметь поля для хранения ФИО, возраста, пола, а также методы получения среднего дохода и среднего расхода в денежном эквиваленте. Продемонстрировать работу всех методов классов, предоставив пользователю выбор типа объекта для демонстрации.
Вариант 3	Написать программу, в которой описана иерархия классов: геометрические фигуры (круг, прямоугольник, треугольник). Реализовать методы вычисления площади и периметра фигуры. Продемонстрировать работу всех методов классов, предоставив пользователю выбор типа фигуры для демонстрации.
Вариант 4	Написать программу, в которой описана иерархия классов: геометрические фигуры (эллипс, квадрат, трапеция). Реализовать методы вычисления площади и периметра фигуры. Продемонстрировать работу всех методов классов, предоставив пользователю выбор типа фигуры для демонстрации.
Вариант 5	Написать программу, в которой описана иерархия классов: геометрические фигуры (ромб, параллелепипед, эллипс). Реализовать методы вычисления площади и периметра фигуры. Продемонстрировать работу всех методов классов, предоставив пользователю выбор типа фигуры для демонстрации.
Вариант 6	Написать программу, в которой описана иерархия классов: геометрические фигуры (куб, цилиндр, тетраэдр). Реализовать методы вычисления объема и площади поверхности фигуры. Продемонстрировать работу всех методов классов, предоставив пользователю выбор типа фигуры для демонстрации.
Вариант 7	Написать программу, в которой описана иерархия классов: геометрические фигуры (куб, конус, тетраэдр). Реализовать методы вычисления объема и площади поверхности фигуры. Продемонстрировать работу всех методов классов, предоставив пользователю выбор типа фигуры для демонстрации.
Вариант 8	Написать программу, в которой описана иерархия классов: геометрические фигуры (ромб, прямоугольник, эллипс). Реализовать методы вычисления площади и периметра фигуры. Продемонстрировать работу всех методов классов, предоставив пользователю выбор типа фигуры для демонстрации.
Вариант 9	Написать программу, в которой описана иерархия классов: функция от одной переменной (синус, косинус, тангенс). Базовый класс должен иметь методы получения значения функции для данного значения переменной, а также создания экземпляра класса, представляющего собой производную текущего экземпляра. Продемонстрировать работу всех методов классов всех классов.
Вариант 10	Написать программу, в которой описана иерархия классов: функция от одной переменной (секанс, косеканс, котангенс). Базовый класс должен иметь методы получения значения функции для данного значения переменной, а также создания экземпляра класса,

	представляющего собой производную текущего экземпляра. Продемонстрировать работу всех методов классов всех классов.
Вариант 11	Написать программу, в которой описана иерархия классов: функция от одной переменной (арксинус, арккосинус, а также класс, необходимый для представления производных). Базовый класс должен иметь методы получения значения функции для данного значения переменной, а также создания экземпляра класса, представляющего собой производную текущего экземпляра. Продемонстрировать работу всех методов классов всех классов.
Вариант 12	Написать программу, в которой описана иерархия классов: функция от одной переменной (арктангенс, арккотангенс, а также класс, необходимый для представления производных). Базовый класс должен иметь методы получения значения функции для данного значения переменной, а также создания экземпляра класса, представляющего собой производную текущего экземпляра. Продемонстрировать работу всех методов классов всех классов.
Вариант 13	Написать программу, в которой описана иерархия классов: функция от одной переменной (логарифм, натуральный логарифм, а также класс, необходимый для представления производных). Базовый класс должен иметь методы получения значения функции для данного значения переменной, а также создания экземпляра класса, представляющего собой производную текущего экземпляра. Продемонстрировать работу всех методов классов всех классов.
Вариант 14	Написать программу, в которой описана иерархия классов: функция от одной переменной (экспонента, гиперболический синус, гиперболический косинус). Базовый класс должен иметь методы получения значения функции для данного значения переменной, а также создания экземпляра класса, представляющего собой производную текущего экземпляра. Продемонстрировать работу всех методов классов всех классов.