Python

12

Переопределение метода.

```
class A:
    def do_something(self):
        print('AA')
class B(A):
    def do_something(self):
        print('BB')
```

Переопределение метода заключается в изменении поведения дочернего метода при сохранении сигнатуры (имени и параметров) родительского

Задание 12.01

Переопределить методы change_weight, change_height в классе Parrot. В случае не передачи параметра - вес изменяется на 0.05

Вызов родительского метода

```
class A:
    def do_something(self):
        print('AA')
class B(A):
    def do_something(self):
        super().do_something()
        print('BB')
```

Задание 12.02

Добавить метод jump, принимающий высоту прыжка. Метод выводит сообщение “Jump X meters”

Переопределить метод jump в дочерних классах. Если передать методу jump класса dog значение больше 0.5, выводить сообщение “Dogs cannot jump so high, аналогично для кошек(2), для попугаев(0.05)

Добавление атрибутов в дочерний класс

```
class A:
    def __init__(self, a):
        self.a = a
class B(A):
    def __init__(self, a, b):
        super().__init__(a)
        self.b = b
```

Задание 12.03

Добавить в класс Perrot новый атрибут - species

Задание 12.04

Добавить в класс Pet пустой метод voice. Заменить имена методов bark, meow на voice. Добавить voice для класса Parrot.

Создать функцию, принимающая список животных и вызывающая у каждого животного метод voice.

Функция dir(object)

Магические методы

Магические методы - методы которые начинаются и заканчиваются с двойного подчеркивания. Магические методы переопределяют действия тех или иных операторов

http://sheregeda.github.io/blog/2015/01/18/maghichieskiie-mietody-python/

Задание 12.05

Создать класс MyTime. Атрибуты: hours, minutes, seconds.

Переопределить магические методы сравнения(равно, не равно), сложения, вычитания, вывод на экран.

Перегрузить конструктор на обработку входных параметров вида: одна строка, три числа, другой объект класса MyTime. В остальных случаях создавать объект по-умолчанию(0-0-0)

Атрибуты класса vs Атрибуты объекта(экземпляра)

```
class Car:
    last_model = None

    def __init__(self, model):
        self.model = model
        Car.last_model = model

car1 = Car('A')
print(Car.last_model)
car2 = Car('B')
print(Car.last_model)
print(car1.last_model)
```

Атрибуты класса - атрибуты, которые делят между собой все объекты данного класса.

last_model - атрибут класса, значение которого доступно из любого объекта и также самого класса

model - атрибут экземпляра, значение которого доступно только через сам объект (атрибут экземпляра - собственность экземпляра)

Задание 12.06

Добавить в класс Pet атрибут counter = 0, значение которого увеличивается при создании любого объекта.

Сделать атрибут counter приватным.