

Python Базовый

Введение в ООП: Наследование


Python Базовый

Introduction



Бондаренко Кирилл

Senior Data scientist, CreatorIQ

 [profile.php?id=100011447245832](https://www.facebook.com/profile.php?id=100011447245832)

 [kirill-bond/](https://www.linkedin.com/in/kirill-bond/)

 [@bond.kirill.alexandrovich](https://www.telegram.com/@bond.kirill.alexandrovich)



Python Базовый

Тема урока

Введение в ООП: Наследование

Python Базовый

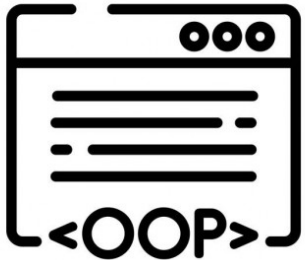
План урока

1. О чем курс
2. Что такое ООП
3. Что такое наследование
4. Примеры использования ООП и наследования на практике

Python Базовый

О чем курс

- ООП: наследование, инкапсуляция, полиморфизм и абстракция
- Рекурсия (структура данных "дерево")
- Работа с файлами (txt, json, yml, xml итд.)
- Модули Python (math, collections, itertools итд.)
- Правила "хорошего тона" в Python коде (PEP8)



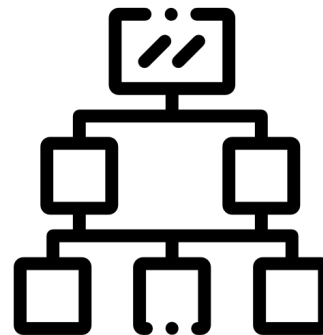
OOP



Recursion



Files



Modules

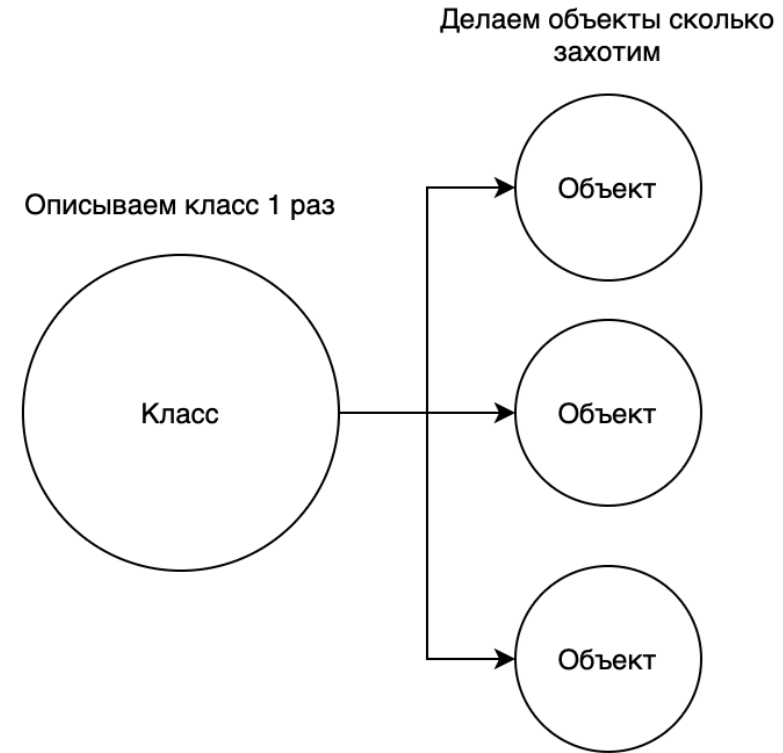


PEP8

Python Базовый

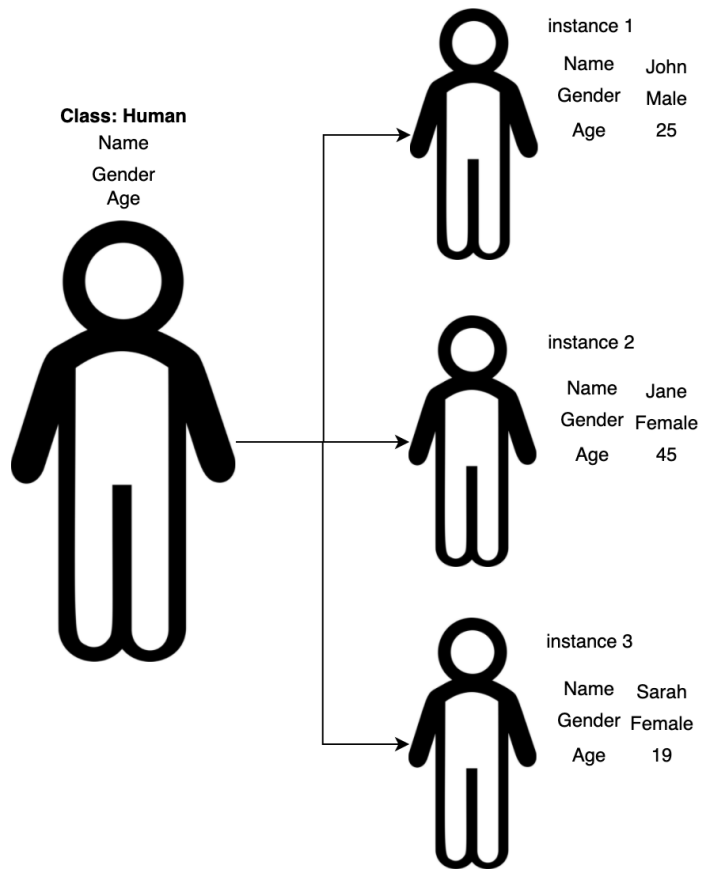
Понятие ООП

Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.



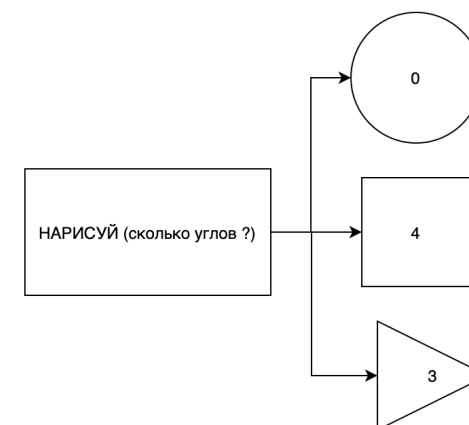
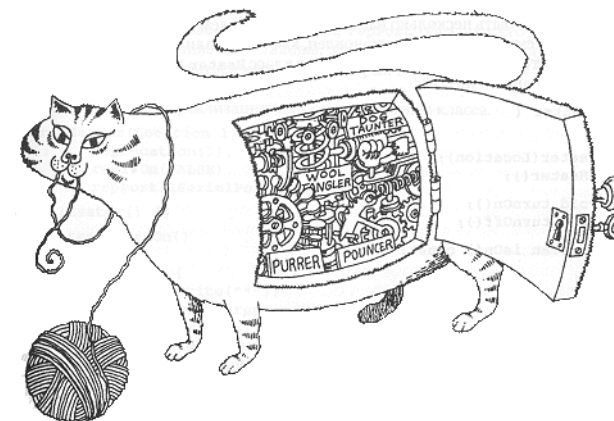
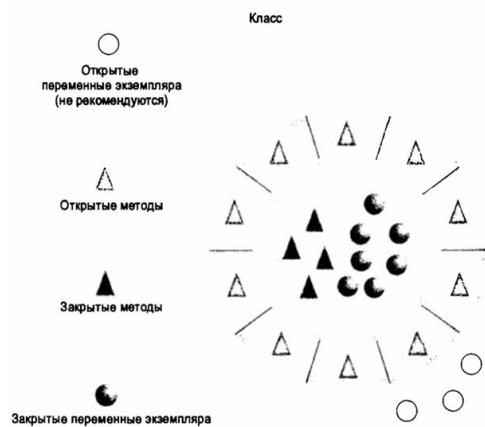
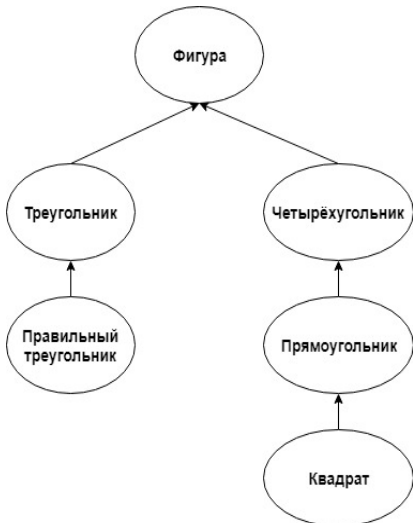
Python Базовый

Класс и объект класса



```
1 class Human:
2     def __init__(self, name, age, gender):
3         self.name = name
4         self.age = age
5         self.gender = gender
6
7     def get_name(self):
8         return self.name
9
10    def get_gender(self):
11        return self.gender
12
13    def get_age(self):
14        return self.age
15
16
17    person = Human(name='John',
18                  age=34,
19                  gender='male')
20
```

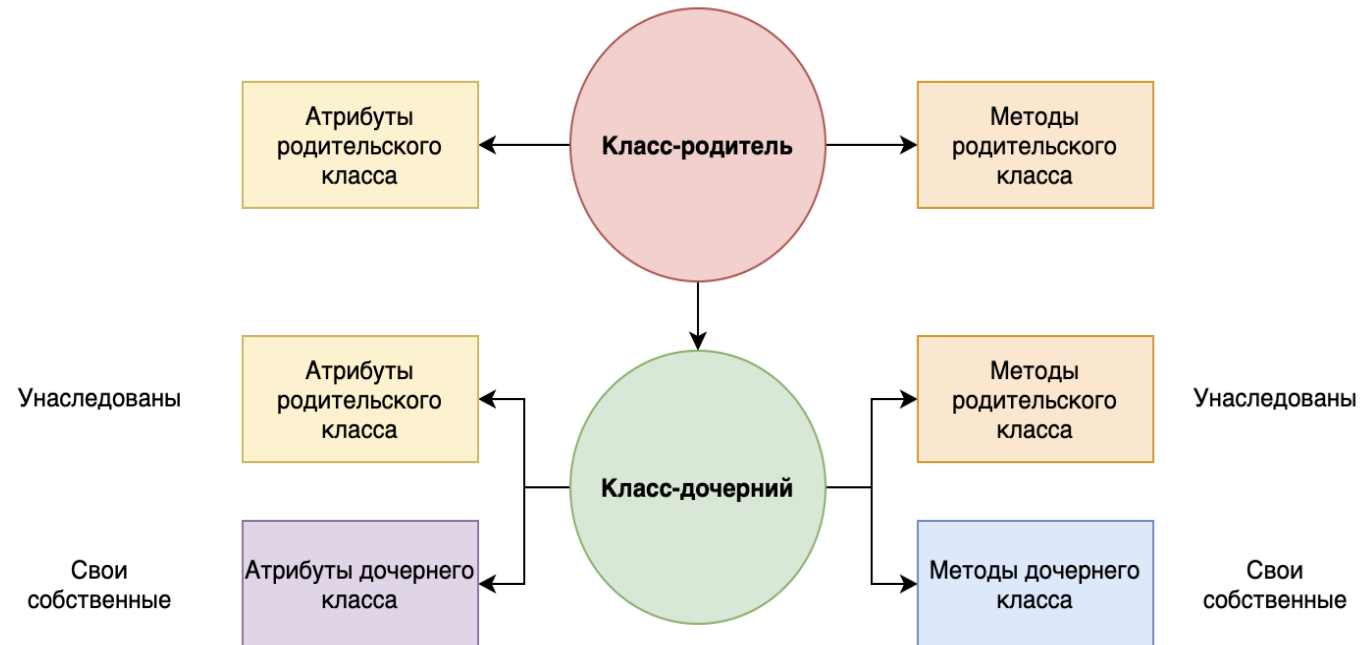
Концепции ООП



Python Базовый

Наследование

Наследование (англ. inheritance) — концепция объектно-ориентированного программирования, согласно которой абстрактный тип данных может наследовать данные и функциональность некоторого существующего типа, способствуя повторному использованию компонентов программного обеспечения.



Python Базовый

Виды наследования

Единое

```
1 class Parent:
2     def __init__(self, name):
3         self.name = name
4
5     def say_hello(self):
6         print("Hello, I am {name}".format(name=self.name))
7
8
9 parent = Parent(name="John")
10 parent.say_hello()
11
12 class Child(Parent):
13     def __init__(self, name, age):
14         super().__init__(name)
15         self.age = age
16
17     def say_hello(self):
18         print(
19             "Hello, my name is {name} and I am {age} years old".format(
20                 name=self.name, age=self.age
21             )
22         )
23
24 child = Child(name="Mark", age=25)
25 child.say_hello()
26
27
```

Множественное

```
1 class A:
2     def __init__(self):
3         self.a = 10
4
5
6 class B:
7     def __init__(self):
8         self.b = 25
9
10
11 class C(A, B):
12     def __init__(self):
13         A.__init__(self)
14         B.__init__(self)
15
16
17 c = C()
18 print("C class has a={a_value} and b={b_value}".format(a_value=c.a, b_value=c.b))
19
```

Python Базовый

Задачи

1. Написать класс автомобиля с атрибутами марки, цвета и объема двигателя и методами: ехать вперед и ехать назад.
2. Написать класс автомобиля, унаследованного от первого класса в пункте 1. Добавить методы поворота налево и направо.
3. Написать класс самолета, имеющего метод взлетать и атрибут модель самолета.
4. Написать класс, унаследованный от машины (2 пункт) и от самолета (3 пункт). Посмотреть что будет.

P.S. Все методы - это просто команда печати, например `print("Drive forward")` и т.д.

Python Базовый

Решение

```
1 class Car:
2     def __init__(self, brand, color, vol):
3         self.brand = brand
4         self.color = color
5         self.vol = vol
6
7     def drive_forward(self):
8         print('Drive forward')
9
10    def drive_backward(self):
11        print('Drive backward')
12
13
14 class Car2(Car):
15     def __init__(self, brand, color, vol):
16         super().__init__(brand, color, vol)
17
18     def turn_right(self):
19         print("Turn right")
20
21     def turn_left(self):
22         print("Turn left")
23
24
25 class Airplane:
26     def __init__(self, model):
27         self.model = model
28
29     def fly(self):
30         print("Start the flight")
31
```

```
32
33 class FlyingCar(Car2, Airplane):
34     def __init__(self, brand, color, vol, model):
35         Car2.__init__(self, brand, color, vol)
36         Airplane.__init__(self, model)
37
38     flying_car = FlyingCar(
39         brand='Tesla',
40         color='black',
41         vol=4.5,
42         model='F'
43     )
44     flying_car.fly()
45
```

Start the flight

Process finished with exit code 0

Информационный видеосервис для разработчиков программного обеспечения



Проверка знаний

TestProvider.com



Проверьте как Вы усвоили данный материал на TestProvider.com

TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

Python Базовый

Спасибо за внимание! До новых встреч!



Бондаренко Кирилл
Senior Data scientist, CreatorIQ

