

Python Базовый

Полиморфизм


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.me/bond.kirill.alexandrovich)



Python Базовый

Тема урока

Полиморфизм

Python Базовый

План урока

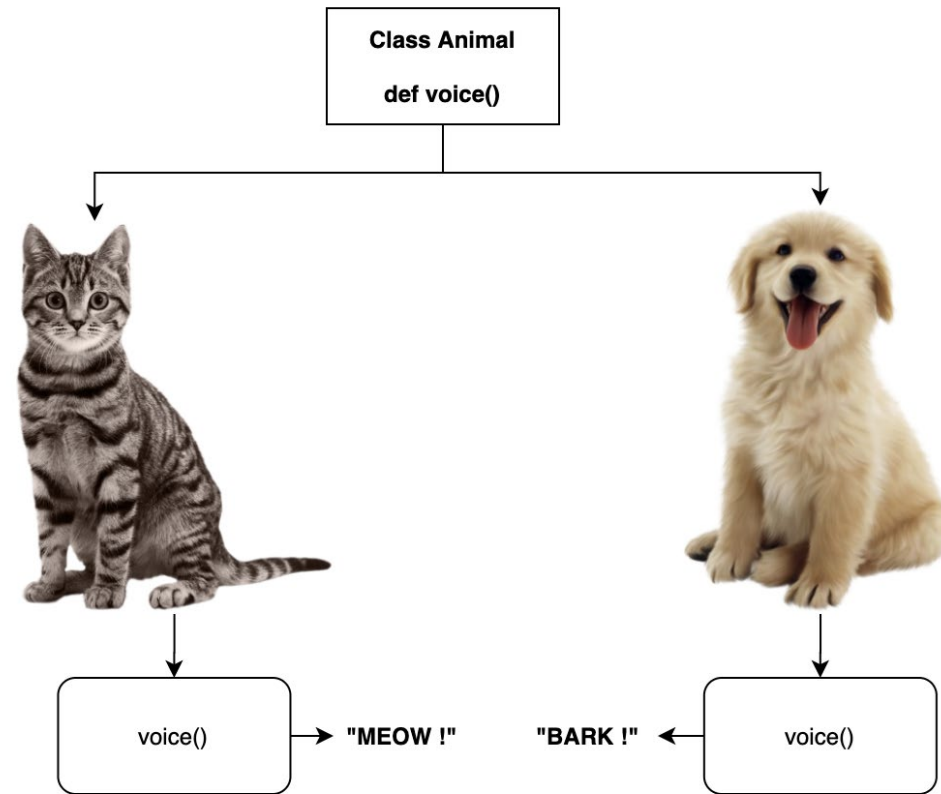
1. Что такое полиморфизм
2. Как на практике он применим
3. Решение задач

Python Базовый

Полиморфизм (Polymorphism)

Полиморфизм - разное поведение одного и того же метода в разных классах. Полиморфизм достигается путем перегрузки или переопределения метода.

Таким образом достигается дополнительная гибкость программы. А также, значительно сокращается дублирование кода, что очень важно для хорошего и поддерживаемого проекта.



Python Базовый

Полиморфизм достигается 2 методами

```
1 class Base:
2     def __init__(self, a):
3         self.a = a
4
5     def print_a(self, square=False, multiplier=None):
6         if square and not multiplier:
7             print(self.a ** 2)
8         elif not square and multiplier:
9             print(self.a * multiplier)
10        elif square and multiplier:
11            print((self.a ** 2) * multiplier)
12        else:
13            print(self.a)
14
15
16 base = Base(4)
17 base.print_a(square=True) # 16
18 base.print_a(square=False, multiplier=2) # 8
19 base.print_a(square=True, multiplier=2) # 32
20 base.print_a() # 4
21
22
```

Перегрузка

```
1 class Multiplier:
2     def __init__(self, a):
3         self.a = a
4
5     def print_a(self, x):
6         print(self.a * x)
7
8 class Exponent(Multiplier):
9     def print_a(self, x):
10        print(self.a ** x)
11
12
13 mult = Multiplier(2)
14 mult.print_a(3) # 6
15 expo = Exponent(2)
16 expo.print_a(3) # 8
17
18
```

Переопределение

Python Базовый

Полиморфизм в Python

```
1 class Car:
2     def __init__(self, name):
3         self.__name_speed_dict = {
4             "Mercedes": 250,
5             "BMW": 300
6         }
7         self._max_speed = self._define_max_speed(name)
8
9     def _define_max_speed(self, name):
10        return self.__name_speed_dict.get(name, 0)
11
12    def distance_time_on_max_speed(self, distance):
13        if not self._max_speed:
14            print("No speed param specified.")
15            return 0
16        return distance / self._max_speed
17
18 car_a = Car(name="BMW")
19 car_b = Car(name="Mercedes")
20 print(car_a.distance_time_on_max_speed(distance=167))
21 print(car_b.distance_time_on_max_speed(distance=167))
22
23
```

```
1 class Animal:
2     def __init__(self, name):
3         self.name = name
4
5     def voice(self):
6         if self.name == "dog":
7             print("Bark !")
8         elif self.name == "cat":
9             print("Meow !")
10        else:
11            print("...")
12
13 cat = Animal(name="cat")
14 dog = Animal(name="dog")
15 veloceraptor = Animal(name="veloceraptor")
16 cat.voice() # Bark !
17 dog.voice() # Meow !
18 veloceraptor.voice() # ...
19
20
```

Python Базовый

Плохой пример полиморфизма

```
1  class Messenger:
2      def __init__(self, connection):
3          self._connection = connection
4
5      def send_message(self, text, option=None):
6          self._connection.send(text)
7
8
9      class ExtendedMessenger(Messenger):
10
11         def send_message(self, text, option=None):
12             if option == "message":
13                 self._connection.send(text)
14             elif option == "clean_memory":
15                 print("Cleaning memory")
16             elif option == "math":
17                 print("2+2=4")
18
19
20
```


Python Базовый

Задачи

1. Написать класс `Parallelogram`, который имеет 2 аргумента `width` и `length` (ширина и длина) и метод `get_area`, который возвращает площадь параллелограмма с такими сторонами. Унаследовать от него класс `Square`, переопределить метод `get_area` как произведение ширины на ширину (или длины на длину).
2. Написать функцию, которая принимает 2 параметра `data` и `new_value`. Если `data` это список или множество, то добавить в него элемент `new_value`, если строка, то сконкатенировать (за исключением, если `new_value` не список, иначе вернуть не измененную `data`), а если это число, логическое значение или словарь, то вернуть не измененную `data`.

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

