

# Магические методы

в Python OOP

FullCode Academy

Преподаватель: Ислам Дуйшобаев

# Цели мастер-класса: Что вы узнаете

## Понимание Основ

Что такое магические методы и почему они важны в Python.

## Практическое Применение

Научимся использовать ключевые методы: `__init__`, `__str__`, `__add__` и другие.

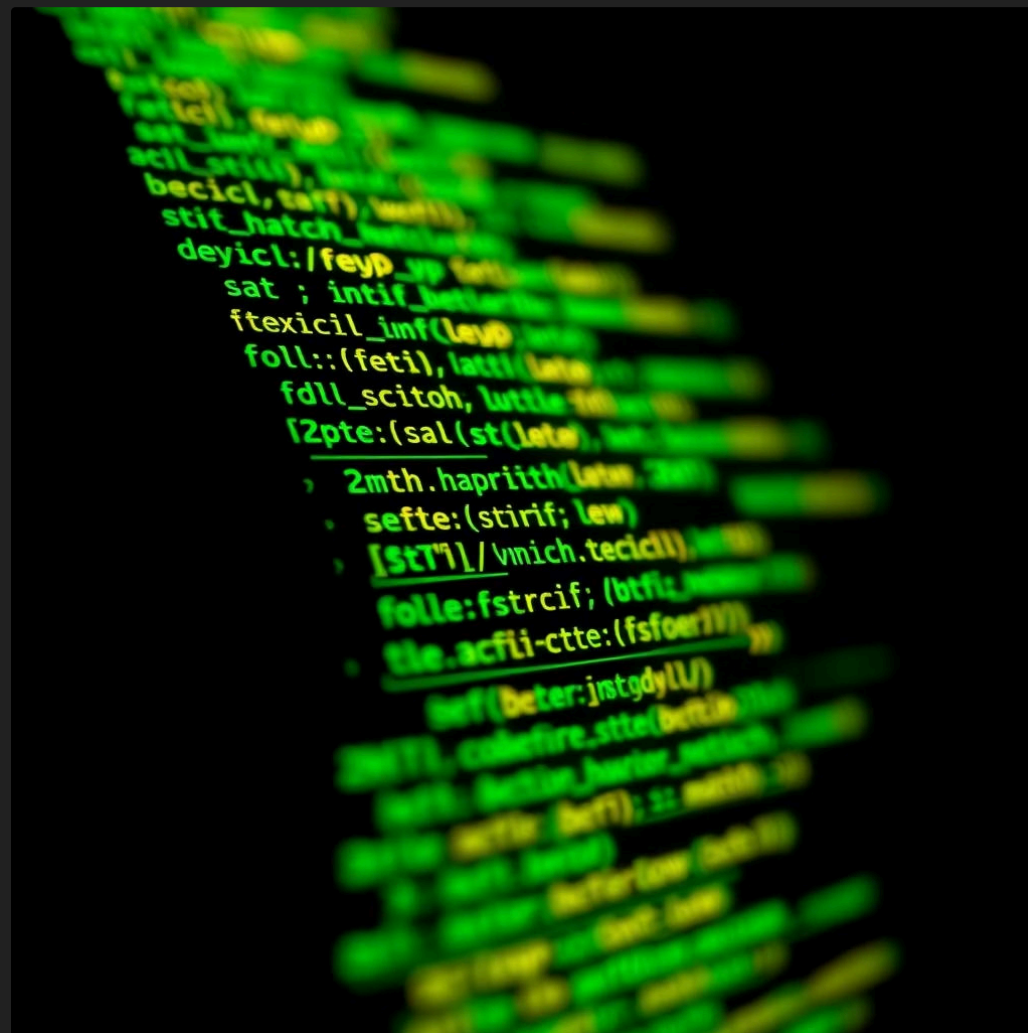
## Улучшение Кода

Как применять магические методы для создания более выразительного и "питонического" кода.

# Введение в магические методы

Магические методы — это специальные методы в Python, которые позволяют классам взаимодействовать со встроенными функциями и операторами языка. Их легко узнать по двойным подчеркиваниям в начале и в конце имени, например, `__len__` или `__add__`.

Их основное назначение — **переопределение стандартного поведения объектов**, делая их более интуитивно понятными и удобными в использовании.



# Ключевые магические методы: Примеры



`__init__`

Конструктор класса, вызывается при создании нового объекта. Инициализирует его атрибуты.



`__str__`

Определяет строковое представление объекта, когда он выводится с помощью `print()` или `str()`.



`__add__`

Позволяет объектам участвовать в операции сложения с помощью оператора `+`.

# Пример кода: Car Class

```
class Car:
    def __init__(self, brand):
        self.brand = brand

    def __str__(self):
        return f"Car: {self.brand}"

my_car = Car("Toyota")
print(my_car) # Выведет: Car: Toyota
```

В этом примере `__init__` присваивает имя бренда, а `__str__` делает вывод объекта Car легко читаемым.

# Практическая демонстрация: Класс Point

Давайте создадим класс `Point` и используем `__add__` для сложения координат точек.


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

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

    def __str__(self): # Добавляем для удобного вывода
        return f"Point({self.x}, {self.y})"

p1 = Point(1, 2)
p2 = Point(3, 4)
p3 = p1 + p2
print(p3) # Выведет: Point(4, 6)
```

Без `__add__` операция `p1 + p2` вызвала бы ошибку, но благодаря магическому методу объекты `Point` теперь "понимают" оператор сложения.



# Расширенные магические методы

1

## Контейнеры

`__len__`, `__getitem__`, `__setitem__`: Для имитации поведения списков или словарей.

2

## Сравнение

`__eq__`, `__lt__`, `__gt__`: Для определения логики сравнения объектов (`==`, `<`, `>`).

3

## Вызов

`__call__`: Позволяет объекту вести себя как функция, делая его вызываемым.

# Итоги: Сила магических методов

## 1 Расширение возможностей

Магические методы позволяют расширять функциональность ваших классов далеко за пределы базового поведения.

## 2 Гибкое поведение

Вы можете переопределять стандартное поведение операторов и встроенных функций, чтобы они работали с вашими пользовательскими объектами.

## 3 Питонический код

Применение магических методов делает ваш код более интуитивным, читаемым и соответствующим духу Python.



# Спасибо за внимание!

Удачи в освоении Python!

