Python'da Dunder (Double Underscore) Metodlar

Dunder metodlar — bu Python tilida maxsus funksiyalar bo‘lib, ularning nomi ikki pastki chiziq bilan boshlanadi va tugaydi (masalan, __init__, __str__, __add__). Ular sinf ichidagi ob’ektlarning o‘zaro qanday munosabatda bo‘lishini belgilashda ishlatiladi. Ushbu qo‘llanmada eng ko‘p ishlatiladigan dunder metodlarga alohida tushuntirishlar, misollar bilan tanishasiz.

__str__

Ob’ekt haqida chiroyli, foydalanuvchiga tushunarli ma’lumotni ko‘rsatadi. Bu metod print() chaqirilganda ishga tushadi.

Misol (kod):

```
class Oquvchi:
    def __init__(self, ism, yosh):
        self.ism = ism
        self.yosh = yosh

    def __str__(self):
        return f"O'quvchi: {self.ism}, Yosh: {self.yosh} yosh"

print(Oquvchi("Ali", 14))
```

__repr__

Ob’ekt haqida ishlab chiquvchiga tushunarli formatda ma’lumot qaytaradi. Debug uchun foydali.

Misol (kod):

```
class Oquvchi:
    def __init__(self, ism, yosh):
        self.ism = ism
        self.yosh = yosh

    def __repr__(self):
        return f"Oquvchi('{self.ism}', {self.yosh})"
```

```
print(repr(Oquvchi("Ali", 14)))
```

__getitem__

Ob’ektdan indeks yordamida qiymat olish imkonini beradi. Masalan, o‘quvchilar ro‘yxatidan ism olish.

Misol (kod):

```
class Sinf:
    def __init__(self):
        self.oquvchilar = ["Ali", "Laylo", "Sardor"]

    def __getitem__(self, index):
        return self.oquvchilar[index]

sinf = Sinf()
print(sinf[1])
```

__setitem__

Ob’ekt ichidagi elementni indeks orqali yangilash uchun ishlatiladi.

Misol (kod):

```
class Sinf:
    def __init__(self):
        self.oquvchilar = ["Ali", "Laylo", "Sardor"]

    def __setitem__(self, index, qiymat):
        self.oquvchilar[index] = qiymat

sinf = Sinf()
sinf[0] = "Miraziz"
print(sinf.oquvchilar)
```

__call__

Ob’ektni funksiya kabi chaqirish imkonini beradi.

Misol (kod):

```
class Oqituvchi:
    def __init__(self, ism):
        self.ism = ism

    def __call__(self, oquvchi_ismi):
        return f"{self.ism} {oquvchi_ismi}ni ro‘yxatga oldi."

oqituvchi = Oqituvchi("Xasan aka")
print(oqituvchi("Dilnoza"))
```

__add__

Ikki ob’ektni qo‘shish operatsiyasini belgilaydi.

Misol (kod):

```
class Oquvchi:
    def __init__(self, ism):
        self.ism = ism

    def __add__(self, other):
        return f"{self.ism} va {other.ism} guruh bo‘ldi."

o1 = Oquvchi("Ali")
o2 = Oquvchi("Sardor")
print(o1 + o2)
```

__sub__

Ob’ektlar orasida ayirish amali uchun ishlatiladi.

Misol (kod):

```
class Sinf:
    def __init__(self):
        self.oquvchilar = ["Ali", "Laylo", "Sardor"]

    def __sub__(self, ism):
        self.oquvchilar.remove(ism)
        return self.oquvchilar

sinf = Sinf()
print(sinf - "Laylo")
```

__eq__, __lt__, __gt__

Ob’ektlarni solishtirish: tengmi (==), kichikmi (<), kattami (>) kabi amallarni o‘zlashtiradi.

Misol (kod):

```
class Oquvchi:
    def __init__(self, ism, baho):
        self.ism = ism
        self.baho = baho

    def __eq__(self, other):
        return self.baho == other.baho

    def __lt__(self, other):
        return self.baho < other.baho

    def __gt__(self, other):
        return self.baho > other.baho

o1 = Oquvchi("Ali", 4)
o2 = Oquvchi("Sardor", 5)

print(o1 == o2)
print(o1 < o2)
print(o1 > o2)
```

Dunder metodlar nima uchun kerak?

Dunder metodlar yordamida Python'da yozilgan sinflarimizni yanada qulay, kuchli va moslashuvchan qilishimiz mumkin. Ular orqali oddiy operatorlar va funktsiyalar bilan ob’ektlarni boshqarish, solishtirish, chop qilish, indekslash kabi amallarni bajarish mumkin. Bu esa kodni yanada tabiiyroq va o‘qilishga qulay qiladi.

Barcha dunder metodlar ikkita pastki chiziq bilan boshlanib va tugab, Python'ning maxsus imkoniyatlaridan hisoblanadi.

`__init__`	Konstruktor, obyekt yaratilganda chaqiriladi
`__str__`	print() orqali ko‘rinadigan matn
`__repr__`	Rasmiy ko‘rinish, debug uchun
`__len__`	len(obj) chaqirilganda ishlaydi
`__getitem__`	obj[index] orqali qiymat olish
`__setitem__`	obj[index] = value qilish
`__delitem__`	del obj[index] qilish
`__call__`	Obyektni funksiya kabi chaqirish
`__add__`	obj1 + obj2
`__sub__`	obj1 - obj2
`__mul__`	obj1 * obj2
`__eq__`	obj1 == obj2
`__lt__`	obj1 < obj2
`__gt__`	obj1 > obj2
`__contains__`	`in` operatori uchun