PYTHONDA OBYEKTGA YO'NALTIRILGAN DASTURLASH(OOP)

PYTHONDA OOP TUSHUNCHALARI

Boshqa umumiy maqsadli tillar singari, python ham boshidan beri ob'ektga yo'naltirilgan til hisoblanadi. **Python** - ob'ektga yo'naltirilgan dasturlash tili. Bu bizga ob'ektga yo'naltirilgan yondashuv yordamida dasturlarni ishlab chiqishga imkon beradi. Python-da biz osongina sinflar va obyektlarni yaratishimiz va ulardan foydalanishimiz mumkin.

Ob'ektga yo'naltirilgan dasturlash tizimining asosiy printsiplari quyida keltirilgan:

- **Object** (Ob'ekt)
- **Class** (Sinf)
- **Method** (metod, usul)
- **Inheritance** (Meros olish)
- **Polymorphism** (Polimorfizm)
- **Data Abstraction** (Ma'lumotlarni olish)
- **Encapsulation** (Inkapsulyatsiya)

Object (Ob'ekt)

Ob'ekt - bu holat va xulq-atvor, xususiyatlarga ega bo'lgan shaxs. Bu sichqoncha, klaviatura, stul, stol, ruchka va boshqa turdagi har qanday haqiqiy ob'ekt bo'lishi mumkin.

Python-dagi hamma narsa ob'ekti bo'lib, deyarli hamma narsada atributlar va metodlar mavjud. Barcha funksiyalar funksiya manba kodida belgilangan**_doc_**qatorini qatorini qaytaradigan o'rnatilgan **doc** atributiga ega.

Class (Sinf)

Sinf ob'ektlar to'plami sifatida aniqlanishi mumkin. Bu ba'zi bir o'ziga xos atributlar va usullarga ega bo'lgan mantiqiy shaxs. Masalan: agar sizda ishchilar sinfingiz bo'lsa, unda u atribut va usulni, ya'ni elektron pochta identifikatori, ism, yosh, ish haqi va boshqalarni o'z ichiga olishi kerak.

Sintaksis

```
class ClassName:
    <Bayonot-1>
    .
    .
    <Bayonot-N>
```

Method (metod, usul)

Metod - bu ob'ekt bilan bog'liq bo'lgan funksiya. Python-da metod faqat sinf misollari uchun xos emas. Har qanday ob'ekt turi metodlariga ega bo'lishi mumkin.

Inheritance (Meros olish)

Merosxo'rlik - bu haqiqiy dunyo meros tushunchasini simulyatsiya qiladigan ob'ektga yo'naltirilgan dasturlashning eng muhim jihati. Bola ob'ekti ota-onaning barcha xususiyatlarini va xatti-harakatlarini egallashini belgilaydi.

Merosdan foydalanib, biz boshqa sinfning barcha xususiyatlari va xatti-harakatlaridan foydalanadigan sinfni yaratishimiz mumkin. Yangi sinf hosil bo'lgan sinf yoki bola klassi, xossalari olingan sinf esa asosiy sinf yoki ota-ona sinfi sifatida tanilgan.

Bu kodning qayta ishlatilishini ta'minlaydi.

Polymorphism (Polimorfizm)

Polimorfizm tarkibida ikkita "poli" va "morflar" so'zlari mavjud. Poli ko'p, morflar esa shakllar degan ma'noni anglatadi. Polimorfizm bilan biz bitta vazifani har xil usulda bajarish mumkinligini tushunamiz. Masalan, sizda sinf hayvonlari bor, va barcha hayvonlar gapirishadi. Ammo ular boshqacha gapirishadi. Bu erda "gapirish" harakati ma'noda polimorf va hayvonga bog'liq. Shunday qilib, mavhum "hayvon" tushunchasi aslida "gapirmaydi", lekin aniq hayvonlar (it va mushuklar kabi) "gapirish" harakatini aniq amalga oshiradilar.

Encapsulation (Inkapsulyatsiya)

Inkapsulyatsiya – obyektga yo’naltirilgan dasturlashning muhim jihati hisoblanadi. U metodlar va o’zgaruvchilarga kirishni cheklash uchun ishlatiladi. Inkapsulyatsiya kod va ma’lumotlar tasodifan o’zgartirilishidan bir bir ichida birlashtiriladi.

Data abstraction (Ma'lumotlarni abstraktsiya qilish)

Ma'lumotlarni ajralish va inkapsulyatsiya qilish ikkalasi ham ko'pincha sinonim sifatida ishlatiladi. Ikkalasi ham deyarli sinonimdir, chunki ma'lumotlar abstraktsiyasiga inkapsulyatsiya orqali erishiladi.

Abstraktsiya ichki tafsilotlarni yashirish va faqat funksionallikni ko'rsatish uchun ishlatiladi. Biron bir narsani mavhumlashtirish, bu narsa funktsiyalar yoki butun dastur bajaradigan ishlarning mohiyatini o'z ichiga olishi uchun narsalarga nom berishni anglatadi.

Ob'ektga yo'naltirilgan va protseduraga yo'naltirilgan dasturlash tillari

Index	Object-based Programming	Procedural Programming
1	Ob'ektga yo'naltirilgan dasturlash - bu muammolarni yechishga qaratilgan yondashuv va hisoblash ob'ektlar yordamida amalga oshiriladigan joyda qo'llaniladi.	Protsedurali dasturlash hisoblashlarni bosqichma-bosqich bajarish bo'yicha ko'rsatmalar ro'yxatidan foydalanadi
2	Bu rivojlanish va texnik xizmat ko'rsatishni osonlashtiradi.	Protsessual dasturlashda loyiha uzoq davom etganda kodlarni saqlash oson emas.
3	Shunday qilib, haqiqiy muammolarni osongina hal qilish mumkin.	Bu haqiqiy dunyoni taqlid qilmaydi. U funktsiyalar deb nomlangan kichik qismlarga bo'linib, bosqichma-bosqich ko'rsatmalar bilan ishlaydi
4	Ma'lumotlarni yashirishni ta'minlaydi. Shunday qilib, protsessual tillardan ko'ra xavfsizroq. Siz shaxsiy ma'lumotlarga biron bir joydan kira olmaysiz.	Protsedurali til ma'lumotlarni bog'lashning to'g'ri usulini taqdim etmaydi, shuning uchun u xavfsiz emas.
5	Ob'ektga yo'naltirilgan dasturlash tillarining misoli C ++, Java, .Net, Python, C # va boshqalar.	Protsessual tillarning namunalari: C, Fortran, Paskal, VB va boshqalar.

PYTHONDA SINF VA OBYEKTLAR

Biz allaqachon muhokama qilganimizdek, sinf virtual ob'ekt bo'lib, uni ob'ektning rejasi sifatida ko'rish mumkin. Sinf paydo bo'lganida obyekt paydo bo'ldi. Keling, buni bir misol orqali tushunaylik.

Aytaylik, sinf binoning prototipidir. Bino polga, eshiklarga, derazalarga va hokazolarga oid barcha ma'lumotlarni o'z ichiga oladi, biz ushbu detallarga asoslanib, xohlagancha bino yasay olamiz. Demak, binoni sinf sifatida ko'rish mumkin va biz shu sinfning shuncha ob'ektini yaratishimiz mumkin.

Boshqa tomondan, ob'ekt sinfning misoli. Ob'ektni yaratish jarayonini **instantatsiya** deb atash mumkin.

O'quv qo'llanmasining ushbu qismida biz python-da sinflar va ob'ektlarni yaratishni muhokama qilamiz. Atributga sinf ob'ekti yordamida qanday erishish mumkinligi haqida ham gaplashamiz.

Python – obyektga yo’naltirilgan dasturlash tili. Pythonda deyarli barcha narsa **obyekt** hisoblandi. Ularning o’z xususiyatlari va funksiyalari bor.

Sinflar esa *obyekt konstruktorlari* hisoblanadi. Ular bilan obyektlar tuziladi.

Sinf hosil qilish

Sinf hosil qilish uchun **class** kalit so’zi ishlatiladi. Hozir biz **Son** degan sinf hosil qilamiz. Shu sinf nomini **print** so’zi bilan ekranga chiqarish buyrug’ini bersak, shu sinf mavjudligi haqida ma’lumot chiqadi:

```
class Son:
    x = 5

print(Son)
```

```
<class '__main__.Son'>
```

Obyekt hosil qilish

Sinflar *obyekt konstruktorlari* ekanligini aytgan edik. Hozir yuqorida hosil qilgan sinfimiz orqali yangi **obyekt** hosil qilamiz. Uning nomi **s1** bo’ladi.

```
class Son:
    x = 5

s1 = Son()
print(s1.x)
```

```
5
```

__init__() funksiyasi

Yuqoridagi misollarimizdagi **sinf** va **obyektlar** bilan shunchaki sodda ko'rinishda tanishib chiqdik. Ammo ular haqiqiy dasturlar tuzishga yaroqsiz. Sinflarning mohiyatini tushunish uchun **__init__()** ichki funksiyasini bilishimiz lozim.

Har bir sinf tuzilgan paytda **__init__()** funksiyasi mavjud bo'ladi. **__init__()** funksiyasi obyektlartuzilayotgan paytda ularning xususiyatlariga qiymatlarni yoki bajarilishi kerak bo'lgan operatsiyalarni biriktiradi.

Hozir **Ishchi** degan sinf hosil qilamiz va unda **ism** va **yosh** ko'rsatkichlariga qiymatlar o'zlashtirish uchun**__init__()** funksiyasidan foydalanamiz.

Keyin**__init__()** funksiyasi har safar yangi obyekt tuzilganda avtomatik tarzda ishlaydi.

Eslatib o'tamiz, **__init__()** funksiyasini yozayotganda har ikkala tarafdan ham ikkitadan (_) tag chiziq yoziladi.

```
class Ishchi:
    def__init__(self, ism, yosh):
        self.ism = ism
        self.yosh = yosh

p1 = Ishchi ("Abbosbek", 20)

print(p1.ism)
print(p1.yosh)
```

```
Abbosbek
20
```

```
class Ishchi:
    def_init_(self, ism, yosh):
        self.ism = ism
        self.yosh = yosh

    def tanish(self):
        print("Mening ismim "+ self.ism)

p1 = Ishchi ("Abbosbek", 20)
p1.tanish()
```

```
Mening ismim Abbosbek
```

Obyekt funksiyalari

Obyektlar ham funksiyaga ega bo'lishi mumkin. Bu funksiyalar sinf ichida tuziladi va obyektlar tomonida ishlatiladi. Masalan, obyekt o'zini tanishtirish funksiyasini tuzamiz:

self parametri

self parametri sinfga tegishli o'zgaruvchilarga murojaat qila olish uchun ishlatiladi. U o'ziga xos yo'llovchi vositadir. U aynan **self** deb nomlanishi shart emas, boshqa nomlarni ishlatish ham mumkin. Faqat u sinfdagi istalgan funksiyaning ilk parametri sifatida yozilishi shart.

Hozir yuqoridagi misolimizdagi **self** parametrlarini **abc** deb o'zgartiramiz va natija o'zgarmaydi.

```
class Ishchi:
    def_init_(abc, ism, yosh):
        abc.ism = ism
        abc.yosh = yosh

    def tanish(abc):
        print("Mening ismim "+ abc.ism)

p1 = Ishchi ("Abbosbek", 20)
p1.tanish()
```

```
Mening ismim Abbosbek
```

Obyekt xususiyatini o'zgartirish

Biror obyektning xususiyatlarini osongina o'zgartirishimiz mumkin. Masalan, dastlab tuzgan obyektimiz 22 yosh bo'lsa, so'ng uni 25 yoshga o'zgartiramiz:

```
class Ishchi:
    def_init_(abc, ism, yosh):
        abc.ism = ism
        abc.yosh = yosh

p1 = Ishchi ("Abbosbek", 20)

p1.yosh = 25

print(p1.yosh)
```

```
25
```

Obyekt xususiyatini o’chirish

Obyekt xususiyatlarini o’chirish ham mumkin. Hozir obyektimizdagi **yosh** xususiyatini o’chiramiz. So’ng uni ekranga chiqarish buyrug’ini beramiz. Dastur ishga tushgach xatolik haqida xabar beriladi.

```
class Ishchi:
    def_init_(abc, ism, yosh):
        abc.ism = ism
        abc.yosh = yosh

p1 = Ishchi ("Abbosbek", 20)

del p1.yosh

print(p1.yosh)
```

```
AttributeError: 'Ishchi' object has no attribute 'yosh'
```

Obyektni o’chirish

Obyektni o’chirish uchun **del** kalit so’zini obyekt nomi bilan qo’llaymiz. Natijada obyekt butkul o’chib ketadi.

Quyidagi kodimizda ham xatolik haqida xabar beriladi. Sababi, biz o’chib ketgan obyektni ekranga chiqarmoqchi bo’lyabmiz:

```
class Ishchi:
    def_init_(abc, ism, yosh):
        abc.ism = ism
        abc.yosh = yosh

p1 = Ishchi ("Abbosbek", 20)

del p1

print(p1)
```

```
NameError: name 'p1' is not defined
```

SINFLARDA KONSTRUKTOR TUSHUNCHASI

Konstruktor - bu sinfning instansiya a'zolarini initsializatsiya qilish uchun ishlatiladigan maxsus metod (funktsiya) turi.

Konstruktorlar ikki xil bo'lishi mumkin:

- Parametrlangan konstruktor
- Parametrlanmagan konstruktor

Ushbu sinf ob'ektini yaratganimizda konstruktor ta'rifi bajariladi. Shuningdek, konstruktorlar ob'ekt uchun biron bir ishga tushirish vazifasini bajarish uchun yetarli resurslar mavjudligini tasdiqlaydilar.

Python-da konstruktor yaratish

Pythonda____init_metodi sinf konstruktorini simulyatsiya qiladi. Ushbu usul sinfni qo'zg'atganda chaqiriladi. Biz ___init_ta'rifiga qarab, sinf ob'ektini yaratishda istalgan sonli argumentlarni berishimiz mumkin. Bu asosan sinf atributlarini ishga tushirish uchun ishlatiladi. Har bir sinf konstruktorga ega bo'lishi kerak, hatto u oddiygina konstruktorga tayansa ham.

Employee sinfining atributlarini ishga tushirish uchun quyidagi misolni ko'rib chiqing.

Example:

```
class Employee:
    def __init__(self,name,id):
        self.id = id; self.name = name;

    def display (self):
        print("ID: %d \nName: %s"%(self.id,self.name))

emp1 = Employee("John",101)
emp2 = Employee("David",102)

#accessing display() method to print employee 1 information
emp1.display();
#accessing display() method to print employee 2 information
emp2.display();
```

```
ID: 101
Name: John
ID: 102
Name: David
```

Misol: Sinf ob'ektlari sonini hisoblash

```
class Student:
    count = 0
    def __init__(self):
        Student.count = Student.count + 1

s1=Student()
s2=Student()
s3=Student()
print("The number of students:",Student.count)
```

```
The number of students: 3
```

Pythonning parametrlanmagan konstruktorga misoli

```
class Student:
    # Constructor - parametrlanmagan
    def __init__(self):
        print("This is non parametrized constructor")

    def show(self,name):
        print("Salom",name)

student = Student()
student.show("Abbosbek")
```

```
Salom Abbosbek
```

Pythonning parametrlangan konstruktorga misol

```
class Student:
    # Constructor - parameterized
    def __init__(self, name):
        print("This is parametrized constructor")
        self.name = name

    def show(self):
        print("Hello",self.name)

student = Student("John")
student.show()
```

```
Hello John
```

Python ichki sinf vazifalari

Sinfda aniqlangan ichki funktsiyalar quyidagi jadvalda tavsiflangan.

SN	Funksiya	Vazifasi
1	**getattr** (obj, name, default)	Ob'ektning atributiga kirish uchun ishlatiladi.
2	**setattr** (obj, name, value)	U ob'ektning o'ziga xos atributiga ma'lum bir qiymatni belgilash uchun ishlatiladi.
3	**delattr** (obj, name)	U ma'lum bir atributni o'chirish uchun ishlatiladi.
4	**hasattr** (obj, name)	Ob'ektda o'ziga xos atribut bo'lsa, u haqiqiy qiymatni qaytaradi.

Misol:

```
class Student:
    def_init_(self,name,id,age):
        self.name = name;
        self.id = id;
        self.age = age

#creates the object of the class Student
s = Student("John",101,22)

#prints the attribute name of the object s
print(getattr(s,'name'))

# reset the value of attribute age to 23
setattr(s,"age",23)

# prints the modified value of age
print(getattr(s,'age'))

# prints true if the student contains the attribute with name id

print(hasattr(s,'id'))

# deletes the attribute age
delattr(s,'age')

# this will give an error since the attribute age has been deleted
print(s.age)
```

```
John
23
True
AttributeError: 'Student' object has no attribute 'age'
```

O'rnatilgan sinf atributlari

Boshqa atributlar bilan bir qatorda, python klassida sinf haqida ma'lumot beradigan ba'zi bir o'rnatilgan sinf atributlari mavjud.

O'rnatilgan sinf atributlari quyidagi jadvalda keltirilgan:

- **__dict__** - Bu sinf nomlari maydoni haqidagi ma'lumotlarni o'z ichiga olgan lug'atni taqdim etadi.
- **__doc__** - U sinf hujjatiga ega bo'lgan qatorni o'z ichiga oladi
- **__name__** - U sinf nomiga kirish uchun ishlatiladi.
- **__module__** - Ushbu sinf aniqlangan modulga kirish uchun foydalaniladi.
- **__bases__** - Unda barcha asosiy sinflarni o'z ichiga olgan korniş mavjud.

Misol:

```
class Student:
    def__init__(self,name,id,age):
        self.name = name;
        self.id = id;
        self.age = age

    def display_details(self):
        print("Name:%s, ID:%d, age:%d"%(self.name,self.id))

s = Student("John",101,22)
print(s.__doc__)
print(s.__dict__)
print(s.__module__)
```

```
{'name': 'John', 'id': 101, 'age': 22}
__main__
```

SINFLARDA VORISLIK TUSHUNCHASI

Vorislik - bu atama sinflarga xosdir. **Vorislik** deb bir sinfdagi barcha funksiya va xususiyatlarni boshqa bir sinf o'ziga o'zlashtirishiga aytiladi.

Funksiyalari meros qilib olinadigan sinf **ona sinf** deyiladi.

Meros qilib olingan funksiyalarni o'ziga o'zlashtiradigan sinf **voris sinf** deyiladi.

Ona sinf hosil qilish

Istalgan sinf **ona sinf** bo'lishi mumkin. Shu sababli ona sinfni hosil qilish xuddi oddiy sinfni hosil qilish kabidir.

Hozir **Odam** degan sinf hosil qilamiz. Unda **ism** va **familiya** parametrlari va tanish degan funksiyasi bo'ladi. So'ngra shu sinf orqali **x** obyekt hosil qilamiz:

```
class Odam:
    def_init_(self, ism, familiya):
        self.ism = ism
        self.familiya = familiya

    def tanish(self):
        print(self.ism, self.familiya)

x = Odam ("Abbosbek", "Ibragimov")
x.tanish()
```

```
Abbosbek Ibragimov
```

Voris sinf hosil qilish

Voris sinf hosil qilish uchun yangi sinf tuzilayotganda ona sinfni paramet sifatida kiritamiz. Shunda voris sinf ona sinfdan barcha xususiyatlarni o'zlashtiradi.

Hozir **Talaba** degan sinf hosil qilamiz. **Odam** sinfi uning onam sinfi bo'ladi. Qavslar ichida ona sinfni kirittamiz va uning barcha xususiyatlarini voris sinf o'zlashtiradi. Qo'shimcha parametr qo'shish shart emas, ammo sinf hosil qilayotganda ichi bo'sh bo'lishi ham mumkin emas.

Agar hechnarsa yozishni istamasak xatolik yuz bermasligi uchun **pass** kalit so’zini qo’shib qo’yamiz:

```
class Odam:
    def_init_(self, ism, familiya):self.ism = ism
        self.familiya = familiya
```

```
def tanish(self):
    print(self.ism, self.familiya)
```

```
# Endi voris sinf ya'ni bola sinfni hosil qilamiz

class Talaba (Odam):
    pass

x = Talaba ("Asadbek", "Suvonov")
x.tanish()
```

```
Asadbek Suvonov
```

__init__() funksiyasini qo’shish

Avvalgi misolimizda **voris sinf** hosil qilganimizda **pass** kalit so’zi bilan cheklanib qo’ya qoldik. Shu sababli voris sinf barcha funksiyalarni avtomatik tarzda o’zlashtirgan edi. Endi voris sinfga **___init___()** funksiyasi bilan parametrlarini joylashtiramiz. Bunda voris sinf ona sinfdagi **___init_()** funksiyasidan emas o’zidagidan foydalanadi.

```
class Odam:
    def_init_(self, ism, familiya):
        self.ism = ism
        self.familiya = familiya

    def tanish(self):
        print(self.ism, self.familiya)

# Endi voris sinf ya'ni bola sinfni hosil qilamiz

class Talaba (Odam):
    def __init__(self, ism, familiya):
        self.ism = ism
        self.familiya = familiya

x = Talaba ("Asadbek", "Suvonov")
x.tanish()
```

```
Asadbek Suvonov
```

Ammo ona sinfdagi **___init___()** funksiyasidan foydalanmoqchi bo’lsak, voris sinfdagi **___init ()**funksiyasi ichiga ona sinfning shu funksiyasini yozamiz:

```
# Endi voris sinf ya'ni bola sinfni hosil qilamiz

class  Talaba (Odam):
    def __init_(self, ism, familiya): Odam.
        init__(self, ism, familiya)

x = Talaba ("Asadbek", "Suvonov")
x.tanish()
```

```
Asadbek Suvonov
```

super() funksiyasi

Sinflar bilan ishlash uchun maxsus **super()** funksiyasi ham mumkin. Bu funksiya ona sinfdagibarcha funksiya va parametrlarni voris sinfga o’zlashtiradi:

```
class Odam:
    def_init_(self, ism, familiya):
        self.ism = ism
        self.familiya = familiya

    def tanish(self):
        print(self.ism, self.familiya)

# Endi voris sinf ya'ni bola sinfni hosil qilamiz

class  Talaba (Odam):
    def_init_(self, ism, familiya):
        super().__init__(ism, familiya)

x = Talaba ("Asadbek", "Suvonov")
```

```
Asadbek Suvonov
```