**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**12-mashg‘ulot.** Pythonda OOP asoslari.

**O‘quv savollari:**

1. OOP asoslari. Klasslarni e’lon qilish va nusxasini yaratish.
2. Sinf va obyekt. Sinf konstruktori.
3. \_\_init\_\_() va \_\_del\_\_() metodlari.

**1. OOP asoslari. Klasslarni e’lon qilish va nusxasini yaratish.**

Pythonda dasturda foydalanish mumkin bo‘lgan int, str va shunga o‘xshash ko‘plab o‘rnatilgan turlar mavjud. Ammo Python, sinflar yordamida o‘z tiplarimizni aniqlashga imkon beradi.

Quyidagi o‘xshashlikni ham o‘tkazish mumkin. Har birimiz ismi-sharifi, yoshi, va shu kabi boshqa xususiyatlarga ega bo‘lgan inson haqida qandaydir tasavvurga egamiz. Inson muayyan harakatlarni bajarishi mumkin: yurish, yugurish, o‘ylash va hokazo. O‘ziga xos xususiyatlar va harakatlar majmuasini o‘z ichiga olgan bu ko‘rinishni ***class*** deb atash mumkin. Aniq bir xususiyatlarni o‘zida jamlagan bu class obyektlar uchun shablon xisoblanadi va turli o‘yektlar bir-biri bilan farq qiladi. masalan: bir odamning ism bunaqa bo‘lsa, boshqalari boshqa ismga ega.

**Sinf** class kalit so‘zi yordamida aniqlanadi:

**class** sinf\_nomi:

    sinf\_atributlari

    class\_methods

Sinf ichida uning atributlari aniqlanadi, ular sinfning turli xususiyatlarini saqlaydi va metodlar sinfning funktsiyalari vazifasini bajarishadi.

**Keling, oddiy sinf yarataylik** :

class Person:

pass

Bunda shartli ravishda shaxsni ifodalovchi Person sinfi aniqlanadi. Bunday holda, sinfda hech qanday metodlar yoki atributlar aniqlanmaydi. Biroq, unda biror narsa aniqlanishi kerakligi sababli, pass iborasi sinf funksionalligi o‘rnini bosuvchi sifatida ishlatiladi. Ushbu bayonot sintaktik jihatdan ba’zi kodlarni aniqlash uchun zarur bo‘lganda ishlatiladi, lekin biz buni xohlamaymiz va ma’lum bir kod o‘rniga biz pass bayonotini kiritamiz.

 Sinf yaratilgandan so‘ng, ushbu sinf ob’yektlarini aniqlash mumkin. Masalan :

**class** Person:

**pass**

tomson = Person() # tomson ob’yektini aninqlash

bobbi = Person() # bobbi ob’yektini aninqlash

Person sinfi aniqlangandan so‘ng, Person sinfining ikkita - tomson va bobbi ob’yektlari yaratildi. Klass ob’yektini yaratish uchun maxsus funktsiyadan - konstruktordan foydalaniladi. U class nomi bilan chaqiriladi va class ob’yektini qaytaradi. Ya’ni, bu holda, Person() chaqiruvi konstruktor chaqiruvini ifodalaydi. Har bir yaratilgan classda parametrlarsiz standart konstruktor mavjud bo‘ladi:

tomson = Person() *# Person() - bu Person sinfining ob’yektini qaytaruvchi konstruktor chaqiruvi*

**2. Sinf va obyekt. Sinf konstruktori**

Class metodlari aslida class ichida belgilangan va uning xususiyatlarini belgilaydigan funktsiyalarni ifodalaydi. Masalan, bitta metod bilan Person sinfini aniqlaylik:

**class** Person: # Person klasining aniqlanishi

**def** say\_hello(self):

**print**("Salom")

tom = Person()

tom.say\_hello() # Salom

Bu yerda **say\_hello()** metodi aniqlandi, u shartli ravishda Salom xabarini konsolga chop etadi. Classga tegishli har qanday metodlarini aniqlashda shuni yodda tutish kerakki, ularning barchasi birinchi parametr sifatida ***self*** kalit so‘zini qabul qilishadi. Self bu o‘zi degan ma’noni anglatadi va ushbu klasga tegishli ekanligini bildiradi. Ammo metod chaqirilganda, bu marametr hisobga olinmaydi hamda qiymat berilmaydi.

ob’yekt nomidan foydalanib, classning metodlariga murojaat qilish mumkin. Mavjud metodlarni chaqirish uchun nuqta belgisi qo‘llaniladi - ob’yekt nomidan keyin nuqta qo‘yiladi va undan keyin metod chaqiruvi keladi:

Ob’yekt.metod([metod qabul qiladigan parametrlar])

Masalan, konsolga salomlashishni chop etish uchun say\_hello() metodini chaqirish:

tomson.say\_hello()     # Salom

Natijada, ushbu dastur konsolga "Salom" qatorini chop etadi.

**Agar metod self dan tashqari boshqa parametrlarni qabul qiladigan bo‘lsa, ular self** parametridan keyin belgilanadi va bunday metodni chaqirishda, ushbu parametrlariga qiymatlar berish talab qilinadi:

class Person: *# Person klasining aniqlanishi*

def say(self, msg): *# method*

print(msg)

tom = Person()

tom.say("Hello , HOW ARE YOU?") *# Hello , HOW ARE YOU?*

**Say()** metodi bu yerda aniqlanadi. Bu metod ikkita parametr oladi: **self** va **msg**. Va ikkinchi parametr uchun - ***msg***, metodni chaqirganda ushbu parametrga qiymat berilishi kerak.

Self kalit so‘zi orqali classning atributi va funksiyalariga murojaat qilish mumkin:

**self**.attribute   *# methodning atributiga murojaat qilish*

**self**.method      *# method chaqiruvi*

**Masalan, Person** sinfida ikkita metodni aniqlaymiz :

**class** Person:

**def** say(**self**, msg):

print(msg)

**def** say\_hello(**self**):

**self**.say("Hello JOB")

tom = Person()

tom.say\_hello() *# Hello work*

Bu yerda **say\_hello()** metodi ichida turib **say()** metodiga murojaat amalga oshirilmoqda

self.say("Hello JOB")

Say() metodi o‘ziga qo‘shimcha ravishda parametrlarni (msg parametrini) qabul qilganligi sababli, metod chaqirilganda ushbu parametr uchun qiymat berish kerak.

Bundan tashqari, metod ichida turib, ob’yekt metodini chaqirganda, self kalit so‘zini ishlatishimiz kerak. Aks holda xatolik kelib chiqadi:

**def** say\_hello(**self**):

say("Hello job") *# Xatolik*

**3.\_\_init\_\_() va \_\_del\_\_() metodlari**

**Konstruktorlar.** Konstruktor sinf ob’yektini yaratish uchun ishlatiladi. Shunday qilib, yuqorida, Person sinfining ob’yektlarini yaratganda, hech qanday parametrlarni qabul qilmaydigan va barcha sinflar bilvosita ega bo‘lgan standart konstruktordan foydalanildi:

tom = Person()

Biroq**, \_\_init\_\_()** deb nomlangan maxsus metod yordamida sinflarda konstruktorni aniq belgilash mumkin (har bir tomonda ikkita past chiziqcha). Masalan, keling, Person sinfini unga konstruktor qo‘shish orqali o‘zgartiramiz:

**class** Person:

*# konstruktor*

**def** \_\_init\_\_(**self**):

print("Person obyektining yaratildi")

**def** say\_hello(**self**):

print("Hello")

tom = Person() *# Person obyekti yaratildi*

tom.say\_hello() *# Hello*

Demak, bu yerda Person sinfining kodida konstruktor va say\_hello () metodi aniqlangan. Birinchi parametr sifatida konstruktor ham metodlar kabi – ***self*** ni qabul qiladi. Odatda konstruktorlar obyekt yaratilganda bajariladigan amallarni aniqlash uchun ishlatiladi:

tom = Person()

Person sinfidagi **\_\_init\_\_()** konstruktori chaqiriladi, u konsolga "Person obyekti yaratildi " matnini chop etadi.

**Ob’yekt atributlari.** Atributlar ob’yekt holatini saqlaydi. Class ichidagi atributlarni aniqlash va o‘rnatish uchun ***self*** so‘zidan foydalanish mumkin. Masalan, quyidagi Person sinfini aniqlaymiz:

**class** Person:

**def** \_\_init\_\_(**self**, nom):

**self**.nom = nom *# shaxning nomi*

**self**.yosh = 1 *# shaxsning yoshi*

tomson = Person("Tomson")

*# Atributga murojaat*

*# qiymatni qabul qilish*

print(tomson.nom) *# Tom*

print(tomson.yosh) *# 1*

*# Qiymatni o‘zgartirish*

tomson.yosh = 37

print(tomson.yosh) *# 37*

Endi Person klassi konstruktori yana bitta parametr - **nom** ni qabul qiladi. Ushbu parametr orqali yaratilgan person ob’yektining nomi konstruktorga uzatiladi.

Konstruktor ichida ikkita atribut o‘rnatilgan – **nom** va **yosh** (shartli ravishda, shaxsning ismi va yoshi):

**def** \_\_init\_\_(**self**, nom):

**self**.nom = nom

**self**.yosh = 1

**self.nom** atributi **nom** o‘zgaruvchisiga tenglashtirib qo‘yildi. ***yosh*** atributining boshlang‘ich qiymati 1 ga teng deb qabul qilindi.

Agar sinfda **\_\_init\_\_** konstruktoridan foydalanilgan bo‘lsa, endi standart konstruktorni chaqira olmaymiz. Endi biz aniq belgilangan **\_\_init\_\_** konstruktorimizni chaqirishimiz kerak, unga **nom** parametri uchun qiymat berilishi kerak:

tomson = Person("Tomson")

Keyinchalik ushbu nom orqali ob’yektning atributlariga murojaat qilib, uni olish va o‘zgartirish mumkin:

print(tomson.nom)*# nom atributining qiymatini olish*

tomson.yosh = 37 *# yosh atributining qiymatini o‘zgartiring*

Asosan, class ichida atributlarni aniqlamasdan uni ob’yektning o‘ziga biriktirib qo‘yish mumkin. Python bu atributni dinamik ravishda ob’yektga qo‘shishga imkon beradi:

**class** Person:

**def** \_\_init\_\_(**self**, nom):

**self**.nom = nom *# shaxsning nomi*

**self**.yosh = 1 *# shaxsning yoshi*

tomson = Person("Tomson")

tomson.company = "Microsoft"

print(tomson.company) *# Microsoft*

Bu yerda kompaniya atributi dinamik ravishda o‘rnatildi. Bu atribut person ob’yektining ish joyini o‘zida saqlaydi. Va o‘rnatgandan so‘ng, uning qiymatiga murojaat qilib, qiymatini olish mumkin. Shuni esda turish kerakki, agar atributni aniqlashdan oldin unga murojjat qilishga harakat qilinsa, dastur xatolik qaytaradi:

tomson = Person("Tomson")

print(tomson.company)

*# ! Xatolik - AttributeError:*

*# Person object has no attribute company*

Sinf ichidagi ob’yektning atributlariga murojaat qilish uchun uning metodlarida self so‘zi ham qo‘llaniladi:

**class** Person:

**def** \_\_init\_\_(**self**, nom):

**self**.nom = nom *# shaxsning nomi*

**self**.yosh = 1 *# shaxsning yoshi*

**def** display\_info(**self**):

print(f"Nomi: {**self**.nom} Yoshi: {**self**.yosh}")

tomson = Person("Tomson")

tomson.display\_info() *# Nomi: Tomson Yoshi: 1*

Bu konsolga ma’lumotlarni chop etadigan ***display\_info()*** metodini belgilaydi**.** Metoddagi ob’yekt atributlariga murojaat qilish uchun esa ***self*** so‘zi ishlatiladi: ***self****.nom* va ***self****.yosh*

**Ob’yektlarni yaratish.** Yuqorida bitta ob’yekt yaratilgan. Ammo shunga o‘xshash tarzda class ning boshqa ob’yektlarni yaratish mumkin:

**class** Person:

**def** \_\_init\_\_(**self**, nom):

**self**.nom = nom *# shaxsning nomi*

**self**.yosh = 1 *# shaxsning yoshi*

def display\_info(**self**):

print(f"Nomi: {**self**.nom} Yoshi: {**self**.yosh}")

tomson = Person("Tomson")

tomson.yosh = 37

tomson.display\_info() *# Nomi: Tomson yoshi: 37*

bobbi = Person("Bobbi")

bobbi.yosh = 41

bobbi.display\_info() *# Nomi: Bobbi Yosh: 41*

Ushbu dasturda Person sinfining ikkita obyekti: tomson va bobbi yaratilgan. Ushbu ob’yektlar Person sinfga tegishli bo‘lib, ular bir xil atributlar va metodlar to‘plamiga egalar. Biroq ular bir-biridan xususiyatlari bilan farq qilishadi.

Dastur bajarilganda, Python self parametri dinamik ravishda aniqlanadi - bu chaqiriladigan ob’yektni ifodalaydi. Masalan, satrda:

tomson.display\_info() *# Nomi: Tomson yoshi: 37*

bobbi.display\_info() *# Nomi: Bobbi Yosh: 41*

**Nazorat savollari:**

1. Pythonda ob’yekt tushunchasiga ta’rif bering.
2. Pythonda class tushunchasiga tarif bering.
3. Classning methodlari qanday yaratiladi?
4. Classga murojaat qanday amalga oshiriladi?
5. Class ob’yekti qanday hosil qilinadi?
6. Metodga qanday murojaat qilinadi?