# OBYEKTGA YO‘NALTIRILGAN DASTURLASHDA BAJARILISH VAQTIDAGI XATOLIKLARNI TRY/EXCEPT MEXANIZMI ORQALI BOSHQARISH

***Gulzoda Meliqo‘ziyeva Komiljon qizi***
*Farg‘ona davlat universiteti talabasi*
*gulzodameliqoziyeva705@gmail.com*
***Mirsaid Yusupov Abdulaziz o‘g‘li***
*Farg‘ona davlat universiteti amaliy matematika va informatika kafedrasi o‘qituvchisi*
*mirsaidbeky@gmail.com*

## ANNOTATSIYA

Obyektga yo‘naltirilgan dasturlash jarayonida bajarilish vaqtida yuzaga keladigan xatoliklar dastur barqarorligiga jiddiy ta’sir ko‘rsatadi. Ushbu ishda xatoliklarning kelib chiqish sabablari va ularni boshqarishdagi asosiy muammolar yoritiladi. try/except mexanizmi asosida xatoliklarni qayta ishlash usullari tushuntirilib, ularning dastur oqimini nazorat qilishdagi ahamiyati ochib beriladi. Natijada foydalanuvchi bilan ishlashda ishonchli va barqaror dastur yaratish imkoniyatlari asoslab beriladi.

**Kalit so‘zlar:** Obyektga yo‘naltirilgan dasturlash, xatoliklarni qayta ishlash, exception, try/except, dastur barqarorligi, bajarilish vaqtidagi xatoliklar.

## ANNOTATION

In object-oriented programming, runtime errors can seriously impact program stability. This work discusses the causes of errors and the main problems in their management. Error handling methods based on the try/except mechanism are explained, and their importance in controlling the program flow is revealed. As a result, the possibilities of creating reliable and stable programs when working with the user are substantiated.

**Keywords:** Object-oriented programming, error handling, exception, try/except, program stability, runtime errors.

## АННОТАЦИЯ

В объектно-ориентированном программировании ошибки, возникающие во время выполнения, оказывают серьезное влияние на стабильность программы. В данной работе рассматриваются причины ошибок и основные проблемы в их управлении. Объясняются методы обработки ошибок, основанные на механизме try/except, и раскрывается их важность в управлении потоком выполнения программы. В результате обосновываются возможности создания надежных и стабильных программ при работе с пользователем.

**Ключевые слова:** Объектно-ориентированное программирование, обработка ошибок, исключение, try/except, стабильность программы, ошибки времени выполнения.

## Kirish

Dasturlash jarayoni faqatgina kod yozishdan iborat emas, balki doimiy ravishda yuzaga keladigan muammolarni aniqlash va ularni to‘g‘ri hal qilish jarayonidir. Har bir dasturchi tajribasidan qat’i nazar, xatoliklarga duch keladi va bu holat dastur yaratishning ajralmas qismi hisoblanadi. Ba’zan xatoliklar oddiy bo‘lib, tezda tuzatiladi, ba’zan esa dastur ishining to‘xtab qolishiga, foydalanuvchi noroziligiga yoki tizim barqarorligining buzilishiga olib kelishi mumkin. Shu sababli dasturlashda xatoliklarni tushunish va ularni ongli ravishda boshqarish muhim ahamiyat kasb etadi.

Xatoliklar turli bosqichlarda paydo bo‘lishi mumkin. Dastlabki yozish jarayonida sintaksis xatoliklari uchrashi ehtimoli yuqori bo‘lsa, dastur ishga tushirilganda mantiqiy yoki bajarilish vaqtidagi xatoliklar yuzaga keladi. Ayniqsa, foydalanuvchi kiritadigan ma’lumotlar bilan ishlashda, fayllar yoki tashqi manbalar bilan bog‘liq jarayonlarda xatoliklar ko‘p uchraydi. Agar bunday holatlar oldindan hisobga olinmasa, dastur kutilmagan vaziyatlarda to‘xtab qoladi yoki noto‘g‘ri natija qaytaradi. Bu esa foydalanuvchi tajribasiga salbiy ta’sir ko‘rsatadi.

Obyektga yo‘naltirilgan dasturlash yondashuvida dastur mantiqi sinflar va obyektlar orqali tashkil etiladi. Har bir sinf o‘ziga xos vazifani bajaradi va ular o‘zaro bog‘langan holda ishlaydi. Bunday tuzilma dastur arxitekturasini aniq va tushunarli qiladi, biroq shu bilan birga xatoliklar ta’sir doirasini kengaytirishi mumkin. Masalan, bitta metodda yuzaga kelgan xatolik boshqa obyektlar ishiga ham ta’sir ko‘rsatishi ehtimoli mavjud. Shu sababli obyektga yo‘naltirilgan dasturlashda xatoliklarni nazorat qilish alohida e’tibor talab qiladi.

Ko‘pincha boshlovchi dasturchilar xatolik yuzaga kelganda dasturni to‘xtatib qo‘yuvchi standart xabarlar bilan cheklanib qoladi. Bunday yondashuv texnik jihatdan to‘g‘ri bo‘lishi mumkin, ammo foydalanuvchi uchun tushunarsiz va noqulay hisoblanadi. Dasturdan foydalanayotgan shaxs texnik tafsilotlarni emas, balki sodda va aniq tushuntirishni kutadi. Shu nuqtai nazardan, xatoliklarni qayta ishlash mexanizmlari nafaqat texnik muammoni hal qiladi, balki foydalanuvchi bilan samarali muloqotni ham ta’minlaydi.

Zamonaviy dasturlash tillarida xatoliklarni boshqarish uchun maxsus mexanizmlar mavjud bo‘lib, ular dastur oqimini nazorat ostida saqlash imkonini beradi. Python dasturlash tilida keng qo‘llaniladigan try/except mexanizmi ana shunday yechimlardan biridir. Ushbu yondashuv xatolik yuz berishi mumkin bo‘lgan kod qismini ajratib ko‘rsatadi va muammo kelib chiqqan taqdirda dastur qanday

harakat qilishi kerakligini aniq belgilaydi. Natijada dastur to‘xtab qolmaydi, balki moslashuvchan tarzda ishlashda davom etadi.

try/except mexanizmi dasturchiga xatoliklarni oldindan taxmin qilish va ularga tayyor turish imkonini beradi. Bu esa kodni yanada puxta, o‘qilishi oson va kengaytirishga qulay holga keltiradi. Xatoliklar aniq joyda ushlanadi, ularning sababi tushunarli bo‘ladi va muqobil yechimlar taklif etiladi. Ayniqsa, obyektga yo‘naltirilgan dasturlashda har bir metod o‘z ichida xatoliklarni boshqara olishi katta afzallik hisoblanadi.

Shu bilan birga, xatoliklarni to‘g‘ri qayta ishlash dastur xavfsizligini ham oshiradi. Nazoratsiz xatoliklar tizim zaifliklariga olib kelishi mumkin bo‘lsa, ongli boshqaruv bunday xavflarning oldini oladi. Barqaror va ishonchli dastur yaratish uchun xatoliklarni inkor etish emas, balki ularni to‘g‘ri qabul qilish va samarali boshqarish muhimdir. Aynan shu yondashuv dasturchining professional darajasini belgilab beradi.

Dasturlash jarayonida yuzaga keladigan asosiy muammolardan biri — kutilmagan xatoliklar sababli dastur ishining to‘xtab qolishidir. Ayniqsa, foydalanuvchi kiritadigan ma’lumotlar bilan ishlashda yoki tashqi resurslarga murojaat qilinganda bu holat tez-tez uchraydi. Agar bunday vaziyatlar oldindan hisobga olinmasa, dastur xato haqida tushunarsiz xabar chiqarishi yoki umuman javob bermay qolishi mumkin. Bu esa foydalanuvchi tajribasiga salbiy ta’sir ko‘rsatadi va dasturga bo‘lgan ishonchni kamaytiradi.

Obyektga yo‘naltirilgan dasturlashda sinflar va metodlar o‘zaro bog‘liq holda ishlaydi. Bitta joyda yuzaga kelgan xatolik butun tizim ishlashiga ta’sir qilishi mumkin. Masalan, metodga noto‘g‘ri parametr uzatilsa yoki obyekt holati kutilganidan farq qilsa, bajarilish vaqtida xatolik yuz beradi. Xatoliklar nazoratsiz qoldirilganda, ularni aniqlash va tuzatish ancha murakkablashadi, chunki muammo manbai dastur oqimi bo‘ylab yashirinib qoladi.

Shu sababli asosiy muammo faqat xatolikning o‘zi emas, balki uni qanday boshqarish masalasidir. Xatoliklarni e’tiborsiz qoldirish dastur barqarorligini pasaytiradi, ularni to‘liq to‘xtatib qo‘yish esa moslashuvchanlikni yo‘qqa chiqaradi. To‘g‘ri yondashuv xatolik yuz bergan paytda dasturni nazorat ostida ushlab turish, foydalanuvchiga tushunarli javob berish va dastur ishini imkon qadar davom ettirishdan iborat. Aynan shu nuqtada maxsus xatoliklarni qayta ishlash mexanizmlariga ehtiyoj tug‘iladi.

Dastur ishini to‘xtatib qo‘yadigan holatlarning oldini olish uchun xatoliklarni oldindan nazorat qilish zarur. Shu maqsadda ko‘plab obyektga yo‘naltirilgan tillarda, xususan Python’da, try/except mexanizmi qo‘llaniladi. Bu yondashuv xatolik yuz berishi ehtimoli bo‘lgan kod qismini alohida ajratib, muammo sodir bo‘lganda dastur qanday harakat qilishini aniq belgilash imkonini beradi. Natijada dastur keskin to‘xtab qolmaydi, balki nazorat ostida ishlashda davom etadi.

try bloki ichida bajarilishi xavfli bo‘lgan amallar yoziladi. Agar ushbu qismda xatolik yuz bermasa, kod odatdagidek ishlaydi. Agar bajarilish vaqtida muammo kelib chiqsa, dastur avtomatik ravishda except blokiga o‘tadi. Bu yerda xatolikni ushlash, unga mos xabar chiqarish yoki muqobil yechimni ishga tushirish mumkin bo‘ladi. Bunday yondashuv dastur oqimini aniq va boshqariladigan holatda saqlab turadi.

Amaliyotda try/except mexanizmi faqat xatolikni “yashirish” uchun emas, balki uni ongli ravishda boshqarish uchun ishlatiladi. Masalan, foydalanuvchi noto‘g‘ri qiymat kiritsa, dastur tushunarsiz texnik xabar o‘rniga sodda va aniq izoh berishi mumkin. Bu obyektga yo‘naltirilgan dasturlashda sinflar va metodlar o‘rtasidagi bog‘liqlikni yanada mustahkamlaydi, chunki har bir obyekt kutilmagan holatlarga tayyor bo‘ladi.

Shuningdek, try/except bilan birga else va finally bloklaridan foydalanish mumkin. else qismi xatolik yuz bermagan holatda bajariladi, finally esa har qanday vaziyatda ishga tushadi. Bu imkoniyatlar resurslarni to‘g‘ri yopish, fayllar bilan ishlash yoki ulanishlarni nazorat qilishda muhim ahamiyat kasb etadi. Natijada dastur nafaqat barqaror, balki o‘qilishi va kengaytirilishi oson bo‘lgan tuzilishga ega bo‘ladi.

Nazariy tushunchalar amaliy misollar bilan mustahkamlanganda yanada tushunarli bo‘ladi. Quyidagi holatni tasavvur qilaylik: foydalanuvchi kiritgan qiymat asosida hisob-kitob bajarilishi kerak. Agar foydalanuvchi kutilgan formatdan chetga chiqsa, dastur oddiy holatda xatolik bilan to‘xtab qoladi. try/except mexanizmi esa bu vaziyatni nazorat ostiga olib, muammoni yumshoq tarzda hal qilish imkonini beradi.

Masalan, foydalanuvchidan son kiritish talab qilinadi va ushbu qiymat ustida arifmetik amal bajariladi. try qismi ichida foydalanuvchi kiritgan ma’lumot son turiga o‘tkaziladi va hisoblash amalga oshiriladi. Agar kiritilgan qiymat son bo‘lmasa, bajarilish vaqtida xatolik yuz beradi. Shu paytda except qismi ishga tushib, dasturni to‘xtatmasdan foydalanuvchiga mos va tushunarli xabar beradi. Natijada dastur mantiqiy oqimini saqlab qoladi va qayta ishlash imkoniyati paydo bo‘ladi.

Obyektga yo‘naltirilgan yondashuvda bu mexanizm sinf metodlari ichida ayniqsa foydali hisoblanadi. Har bir metod o‘ziga tegishli xatoliklarni mustaqil boshqarishi mumkin bo‘ladi. Bu esa kodni modullashtiradi, uni o‘qish va qo‘llab-quvvatlashni osonlashtiradi. Xatoliklar aniq joyda ushlanib, umumiy tizimga salbiy ta’sir ko‘rsatmaydi.

finally blokidan foydalanish orqali esa har qanday holatda bajarilishi zarur bo‘lgan amallar kafolatlanadi. Masalan, fayl yopish, ulanishni uzish yoki vaqtinchalik resurslarni tozalash kabi jarayonlar xatolik bo‘lishidan qat’i nazar bajariladi. Bu yondashuv dastur barqarorligini yanada oshiradi va kutilmagan holatlarning oldini oladi.

## Xulosa

Dastur yaratish jarayonida xatoliklar muqarrar holat bo‘lib, ular dasturchining tajribasidan qat’i nazar yuzaga keladi. Muhim jihat xatoliklarning mavjudligi emas, balki ularga qanday munosabat bildirilishidir. Agar xatoliklar nazoratsiz qoldirilsa, dastur ishining to‘xtab qolishi, noto‘g‘ri natijalar paydo bo‘lishi yoki foydalanuvchi bilan muloqotning buzilishi kabi salbiy holatlar yuzaga keladi. Shu sababli xatoliklarni ongli ravishda boshqarish dastur barqarorligini ta’minlovchi asosiy omillardan biri hisoblanadi.

Obyektga yo‘naltirilgan dasturlash muhitida dastur tarkibi bir-biri bilan bog‘langan sinflar va obyektlardan iborat bo‘ladi. Bunday tuzilma kodni tartibli va mantiqan izchil qiladi, ammo ayni paytda xatoliklarning ta’sir doirasini kengaytirishi mumkin. Bitta metod ichida yuzaga kelgan xatolik boshqa obyektlar faoliyatiga ham ta’sir o‘tkazishi ehtimoli mavjud. Shu nuqtai nazardan, xatoliklarni aniq joyda ushlash va boshqarish katta ahamiyat kasb etadi. try/except mexanizmi aynan shu muammoni samarali hal qilish imkonini beradi.

Ushbu mexanizm yordamida bajarilish vaqtida yuzaga keladigan xatoliklar dastur oqimini buzmasdan nazorat ostida ushlanadi. Dastur kutilmagan vaziyatlarda to‘xtab qolish o‘rniga, oldindan belgilangan mantiq asosida ishlashda davom etadi. Bu esa foydalanuvchi uchun qulaylik yaratadi va dasturga bo‘lgan ishonchni oshiradi. Texnik jihatdan murakkab xatoliklar o‘rniga sodda va tushunarli xabarlar berilishi foydalanuvchi tajribasini sezilarli darajada yaxshilaydi.

try/except yondashuvi nafaqat xatoliklarni ushlash, balki kod sifatini oshirishga ham xizmat qiladi. Xatoliklar bilan bog‘liq holatlar aniq ajratib ko‘rsatilgani sababli kod o‘qilishi osonlashadi, mantiqiy tuzilma ravshanlashadi va kelajakda kengaytirish yoki o‘zgartirish jarayonlari soddalashadi. Bundan tashqari, else va finally bloklaridan foydalanish orqali resurslar bilan ishlash, fayllarni yopish va ulanishlarni to‘g‘ri boshqarish imkoniyati paydo bo‘ladi. Bu esa dastur xavfsizligi va barqarorligini yanada oshiradi.

Zamonaviy dasturiy ta’minot foydalanuvchiga yo‘naltirilgan bo‘lishi talab etiladi. Bunday sharoitda xatoliklarning ochiq va tushunarli tarzda qayta ishlanishi muhim ahamiyatga ega. Foydalanuvchi noto‘g‘ri amal bajargan taqdirda ham dastur mantiqan to‘g‘ri javob berishi va yo‘l-yo‘riq ko‘rsatishi kerak. try/except mexanizmi aynan shu vazifani bajarishga xizmat qiladi va dastur bilan foydalanuvchi o‘rtasidagi muloqotni mustahkamlaydi.

Xatoliklarni qayta ishlashga nisbatan bunday yondashuv dasturchining professional darajasini namoyon etadi. Xatoliklardan qochish emas, balki ularni oldindan hisobga olish va samarali boshqarish barqaror va ishonchli dastur yaratishning asosiy mezonlaridan biridir. Natijada obyektga yo‘naltirilgan dasturlashda try/except mexanizmidan ongli va tizimli foydalanish yuqori sifatli dasturiy yechimlar yaratishga mustahkam zamin hozirlaydi.

**Foydalanilgan adabiyotlar**

1. Лутц М. Изучаем Python. — 5-е изд. — СПб.: Питер, 2019. — 1280 с.
2. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приёмы объектно-ориентированного проектирования. Паттерны проектирования. — М.: Вильямс, 2020. — 366 с.
3. Хорстманн К. С. Объектно-ориентированное программирование на Java. — М.: Бином, 2018. — 816 с.
4. Макконнелл С. Совершенный код. — 2-е изд. — М.: Русская редакция, 2017. — 896 с.