SHART OPERATORLARIDAN FOYDALANIB REAL HAYOTIY VAZIYATLARNI MODELLASHTIRISH

Yusupov Mirsaidbek
Farg‘ona davlat universiteti amaliy matematika va informatika kafedrasi o’qituvchisi
mirsaidbeky@gmail.com
Abdusattorova Mohigul Valijon qizi
Farg‘ona davlat universiteti Amaliy Matematika fakulteti 3-bosqich talabasi
Mohigulabdusattorova50@gmail.com

Annotatsiya: Ushbu maqola shartli operatorlar yordamida haqiqiy hayotiy vaziyatlarni, xususan, imtihon natijalarini baholash jarayonini dasturiy modellashtirishni o‘z ichiga oladi. Python tilidagi “if-elif-else” tuzilmasi va Ob’ektga Yo‘naltirilgan Dasturlash (OOP) tamoyillari asosida qanday qilib qaror qabul qilish mantiqini tushunish mumkinligi tushuntiriladi. Maqolada, baholashni avtomatlashtirishni ko‘rsatish uchun oddiy, ammo amaliy dastur misoli keltirilgan. Kod parchalarining soddaligi shartlarni qo‘llash usulini aniq ko‘rsatib beradi. Ushbu ma’lumotlar dasturlashni endi o‘rganayotganlar hamda amaliy mashqlar izlayotganlar uchun foydali bo‘ladi.

Kalit so‘zlar: *Shart operatorlari, Real hayotiy modellashtirish, Python dasturlash tili, Ob’ektga yo‘naltirilgan dasturlash, Imtihon natijalarini baholash, Qaror qabul qilish algoritmlari, “if-elif-else” konstruktsiyasi, Dasturiy mantiq va optimallashtirish.*

Аннотация: В статье рассматривается моделирование реальных ситуаций с использованием условных операторов, в частности процесса оценки экзаменационных результатов. Через конструкции “if-elif-else” и элементы ООП на Python объясняется логика принятия решений. Приведена небольшая модель автоматической проверки оценок как практический пример. Краткие фрагменты кода показывают применение условий на практике. Материал полезен для начинающих программистов и учебных занятий.

Ключевые слова: *Условные операторы, моделирование реальных жизненных ситуаций, язык программирования Python, объектно-ориентированное программирование, оценка экзаменационных результатов, алгоритмы принятия решений, конструкция if-elif-else, логика программы и оптимизация.*

Abstract: This article explores the use of conditional statements to model real-world situations, focusing specifically on automating exam result evaluations. It

explains the decision-making logic behind Python's "if-elif-else" structures and demonstrates an Object-Oriented Programming (OOP) approach. A simple, practical example of an automatic grading model is provided to illustrate these concepts. The straightforward code snippets clearly show how to implement conditional logic in practice. This resource is useful for beginners learning programming fundamentals and for those seeking hands-on exercises.

Key words: *Conditional statements, real-life modeling, Python programming language, object-oriented programming, assessment of exam results, decision-making algorithms, if-elif-else construction, program logic and optimization.*

Shart operatorlari va ular orqali real jarayonlarni modellashtirish

Dasturlashda shart operatorlari deganida, kompyuterga "shartli qaror qabul qilish" qobiliyatini beruvchi asosiy vosita tushuniladi. Bu konstruksiyalar orqali dasturimizga "agar bu shart bajarilsa, shu amalni bajar, aks holda boshqasini bajar" degan oddiy va tushunarli ko'rsatma berishimiz mumkin. Bu nafaqat oddiy dasturlarda, balki murakkab Obyektga Yo'naltirilgan Dasturlash (OYP)da ham keng qo'llaniladi, chunki u real hayotdagi murakkab mantiqiy jarayonlarni soddalashtirib, kompyuter tushunadigan tilga aylantiradi.

Misol uchun, kundalik hayotimizda qiladigan qarorlarimizning ko'pi shunga o'xshaydi: "Agar imtihondan 90 ball olgan bo'lsam, ota-onam sovg'a beradi", "Agar kredit tarixim yaxshi bo'lsa, bank menga pul beradi", "Agar mening yoshim 18dan katta bo'lsa, saylovda ovoz berishim mumkin". Ana shunday qarorlar zanjirini dasturlashda shart operatorlari yordamida ifodalaymiz.

Python tilida bu ish "if", "elif" va "else" kalit so‘zlari yordamida juda oson bajariladi. Bu tuzilma hayotdagi "agar...yoki...yoki...bo'lmasa" degan fikrlash tarzimizga to'g'ri keladi. Bu nafaqat yangi o'rganayotganlar uchun tushunarli, balki dastur mantiqini aniq va izchil qiladi. Bir misol keltiraylik: talaba 86 ball to'plagan deb tasavvur qilaylik. Dasturimiz shu ballni tekshirib, "A" bahosini avtomatik tarzda belgilab beradi. Bu yerda aynan shart operatori ishlaydi.

Shart operatorlarining yana bir kuchi, ular dasturchiga katta moslashuvchanlik beradi. Siz qancha shart qo'shishni, ularni qanday tartibda tekshirishni va qaysi shartga ustunlik berishni o'zingiz belgilaysiz. Aynan shu sababdan ular nafaqat dasturlashning boshlang'ich darajasi, balki matematik hisob-kitoblar, moliyaviy tahlillar, avtomatlashtirilgan baholash tizimlari va hatto tibbiy dasturlardagi tashxis jarayonlarigacha qo'llaniladi.

OYP kontekstida shart operatorlari ko'pincha obyektning ma'lum bir metodi ichida joylashadi. Masalan, "Talaba" degan klassimiz bor desak, unda "baholash" degan metod bo'lishi mumkin. Ana shu metodning ichida talabaning ballari qay bahoga mos kelishini shart operatorlari yordamida tekshiramiz. Bu usul nafaqat kodni tartibli

va o'qish uchun qulay qiladi, balki kelajakda dasturimizga yangi funksiyalar qo'shish imkoniyatini oshiradi.

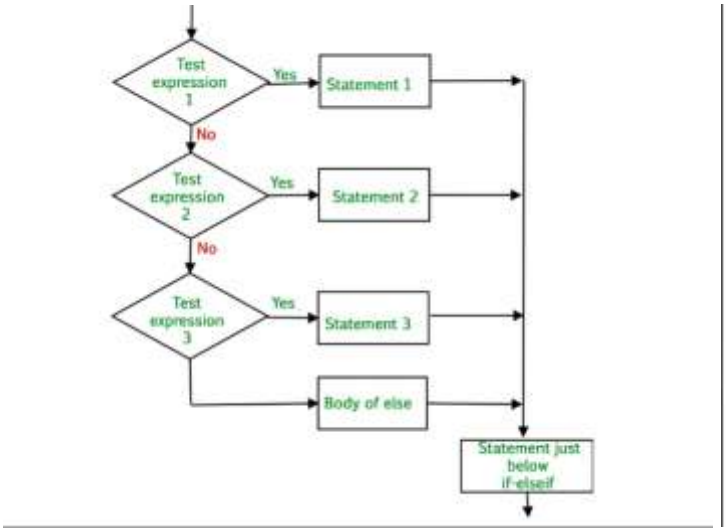
Dasturimizni tezroq va samaraliroq ishlashi uchun shart operatorlarini to'g'ri joylashtirish muhimdir. Eng ko'p uchraydigan holatni tekshirish uchun shartni eng oldinga qo'yish, keraksiz tekshiruvlarni olib tashlash yoki murakkob mantiqiy shartlarni soddaroq ko'rinishga keltirish kabi usullar dasturimizning islash tezligiga sezilarli ta'sir ko'rsatadi. Shuning uchun ham shart operatorlarini oddiy "ha" yoki "yo'q" degan javob beruvchi vosita sifatida emas, balki butun bir samarali algoritm yaratishning poydevori deb hisoblash kerak.

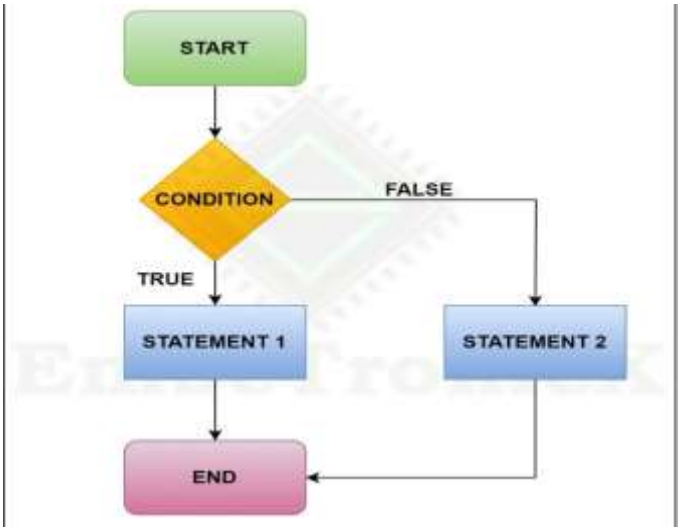
Quyidagi kod bunga misol:

- Ball = 86
- If ball >= 86:
- Baho = “A”
- Elif ball >= 71:
- Baho = “B”
- Elif ball >= 56:
- Baho = “C”
- Else:
- Baho = “Fail”
- Print(“Natija:”, baho)

Bu misolda shart operatorlari real hayotdagi baholash tizimini aniq va lo‘nda modellashtirib beradi. Dastur avval eng yuqori shartni tekshiradi, so‘ngra past darajadagi variantlarga o‘tadi — xuddi o‘qituvchining baholash mantig‘I kabi.

Shart operatorlarining kuchi shundaki, ular faqat matematik yoki aniq natijalarni emas, balki inson faoliyatidagi murakkab qaror qabul qilish jarayonlarini ham modellashtirishga imkon yaratadi. Shu bois ular Python dasturlashidagi eng muhim mantiqiy vositalardan hisoblanadi.





Real hayotiy modellashtirishda shart operatorlarining qo‘llanilishi va ularning amaliy ahamiyati

Haqiqiy hayotdagi jarayonlarni dasturga aylantirish (modellashtirish) degani – aslida bizni o’rab turgan voqelikni matematik qonunlar va mantiqiy qoidalar asosida tasvirlashdir. Python tilida shart operatorlari ana shu vazifani bajaradi: ular hayotiy qarorlarni kompyuter tushunadigan tilga tarjima qilishimizga yordam beradi.

Dasturchi biror narsani modellashtirishni boshlaganda, avval real hayotda qanday shartlar va qoidalar mavjudligini aniqlaydi. Keyin esa ularni “agar…, unda…” ko’rinishidagi ketma-ket tekshiruvlar sifatida dasturga joylashtiradi. Aynan shu sababdan “if-elif-else” tuzilmasi har qanday modellashtirishning asosiy tarkibiy qismiga aylanadi.

Buni tushunish uchun bankdagi kredit berish jarayonini misol qilaylik. Haqiqiy hayotda bank xodimi mijozning yoshi, maoshi, oldingi kredit tarixi va kafolatlari haqida ma’lumot to’playdi va shu asosda qaror qabul qiladi. Dasturda esa bu jarayon shart operatorlari zanjiriga aylanadi:

· Agar mijoz yoshi yetarli bo’lsa,
· agar uning daromadi yetarli bo’lsa,
· agar uning kredit tarixi yaxshi bo’lsa…
· unda kreditni tasdiqlash,
· aks holda rad etish.

Pythonda bu modellashtirish juda oddiy va tushunarli bo’ladi. Dastur mantiqini bosqichma-bosqich yozish mumkin, bu esa murakkab hayotiy vaziyatlarni ham soddalashtirib ko’rsatish imkonini beradi. Quyidagi oddiy misol buni yanada aniqroq qilib ko’rsatadi:

- Yosh = 25

- Daromad = 6000000
- Kredit_tarixi = True
- If yosh >= 20 and daromad >= 5000000 and kredit_tarixi:
- Natija = “Kredit tasdiqlandi”
- Else:
- Natija = “Rad etildi”
- Print(natija)

Ushbu algoritm haqiqiy hayotdagi murakkab tahlil jarayonining aniq bir aksidir va shart operatorlarining dasturda qanchalik amaliy va foydali ekanligini ko’rsatib beradi.

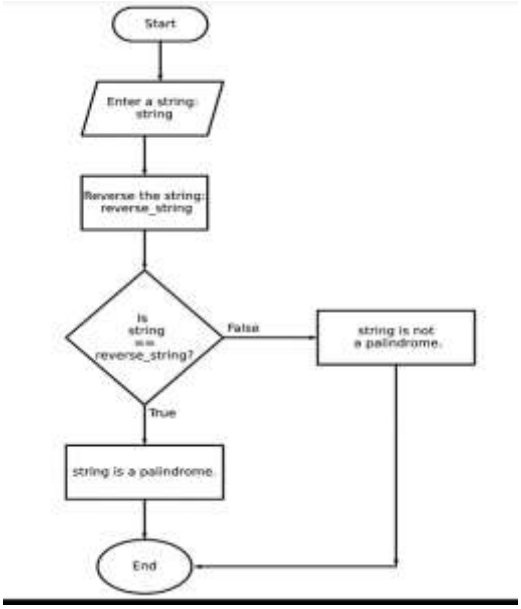
Obyektga yo’naltirilgan dasturlash (OYP) yondashuvidan foydalanganda, bu modellashtirish butunlay yangi va aniqroq ma’no kasb etadi. Tasavvur qiling, “KreditMijoz” degan bir klass yaratasiz. Bu klass ichida mijozning yoshi, maoshi va kredit tarixi kabi barcha kerakli ma’lumotlar saqlanadi. Shu klassning ichidagi kreditniTasdiqlash() degan metod esa, shart operatorlari yordamida, aynan shu mijozga kredit berish kerakmi yoki yo’qmi degan yakuniy qarorni chiqaradi. Bu usul butun mantiqni bir joyda, tartibli va kelajakda o’zgartirish oson bo’lgan shaklda tashkil qiladi.

Shart operatorlarining yana bir kuchi shundaki, ular haqiqiy hayotdagidek o’zgaruvchan sharoitlarni osongina qayta ishlash imkonini beradi. Masalan, bir universitetda imtihondan o’tish bali 60 bo’lsa, ikkinchisida 70 bo’lishi mumkin. Dasturga shunchaki shartni o’zgartirib qo’yish kifoya va u yangi qoidaga moslashadi. Xuddi shu algoritm turli muassasalar uchun ishlayveradi. Bu shart operatorlarining modellashtirishdagi ajoyib moslashuvchanligidan dalolat beradi.

Haqiqiy jarayonlarni dasturga aylantirishda shart operatorlaridan foydalanish xatolarni ham sezilarli darajada kamaytiradi. Inson bir nechta omillarni e’tiborsiz qoldirishi mumkin, lekin dastur dasturchi ko’rsatgan barcha shartlarni hech qayermasdan, aniq va ketma-ket tekshiradi. Aynan shu sababli banklar, sug’urta kompaniyalari, avtomatik baholash tizimlari va hatto dori-darmonlarni tavsiya qiluvchi dasturlar ham o’z mantiqini asosan shart operatorlari ustiga qurishadi.

Modellashtirishning muhim jihati – jarayonning aniq va tushunarli bo’lishidir. Shart operatorlari yordamida butun jarayon aniq bosqichlarga bo’linadi. Har bir bosqich mustaqil ravishda tekshiriladi va barchasi mantiqiy bir yakunga olib kelinadi. Bu – haqiqiy hayotdagi qaror qabul qilish zanjirini kompyuter tilidagi eng sodd va ishonchli ko’rinishidir.

Quyidagi sxema, dastur qanday qilib bir nechta “ha/yo’q” darvozalaridan o’tib, oxirgi natijaga kelishini soddagina ko’rsatib beradi:



Imtihon natijalarini baholash jarayonini shart operatorlari yordamida modellashtirish

Imtihon natijalarini baholash – bu haqiqiy hayotda hammimiz tushunadigan eng oddiy va keng tarqalgan misollardan biridir. O'qituvchi yoki tizim talabaga baho qo'yishda bir qancha omillarni inobatga oladi: to'plagan ballari, minimal o'tish chegarasi, qo'shimcha ballar, qayta topshirish imkoniyati va hokazo. Aslida, bu butun jarayon shartlardan iborat bo'lib, Python dasturlash tilida uni aniq va osonlikcha modellashtirish mumkin. Shart operatorlari (if-elif-else) orqali "agar ball shuncha bo'lsa, shu bahoni ber" degan oddiy qarorlar ketma-ketligini yaratishimiz mumkin.

Oddiy misol ko'rib chiqaylik. Faraz qilaylik, 100 ballik tizim asosida baholaymiz. Talaba 86 yoki undan yuqori ball olgan bo'lsa, "A" bahosi, 71-85 ball orasida bo'lsa "B", 56-70 ball orasida bo'lsa "C", undan past bo'lsa esa "Yiqildi" degan natija beriladi. Bu baholash mezonlarini dasturda quyidagicha ifodalashimiz mumkin:

- ```python
- Def baholash(ball):
- If ball >= 86:
  - Return "A"
- Elif ball >= 71:
  - Return "B"
- Elif ball >= 56:
  - Return "C"

- Else:
  - Return “Yiqildi”
- ```

Bu yerda har bir shart haqiqiy hayotdagi baholash qoidasiga to’g’ri keladi. Dastur ballni oladi, shartlarni ketma-ket tekshiradi va mos keladigan bahoni qaytaradi. Ushbu funksiya yordamida har bir talabani alohida tekshirish o’rniga, avtomatlashtirilgan va adolatli tizim yaratish mumkin.

Obyektga yo’naltirilgan dasturlash nuqtai nazaridan, imtihon baholash jarayonini “Talaba” obyektlari orqali modellashtirish ancha qulay va tushunarli. Har bir talaba uchun ismi, guruhi, balli kabi xususiyatlar, shuningdek “baho_hisobla()” kabi metodlar bo’lishi mumkin:

- ```python
- Class Talaba:
- Def __init__(self, ism, ball):
  - Self.ism = ism
  - Self.ball = ball

- Def baho_hisobla(self):
  - If self.ball >= 86:
    - Return “A”
  - Elif self.ball >= 71:
    - Return “B”
  - Elif self.ball >= 56:
    - Return “C”
  - Else:
    - Return “Yiqildi”

```

Bu modelda shart operatorlari talaba obyekti ichida joylashgan bo’lib, tashqi kod faqat metodni chaqiradi. Natijada, dastur mantiqi tartibli, modulli va o’qish uchun qulay bo’ladi. Agar keyinroq kengaytirish kerak bo’lsa, masalan, “bonus_ball” qo’shish yoki qayta topshirish natijasini hisobga olish kabi o’zgarishlarni aynan shu metod ichida amalga oshirish mumkin.

Qaror qabul qilish algoritmlarida aniqlik va izchillik juda muhimdir. Imtihon natijalarini baholashda noto’g’ri shart yozilsa, talaba nohaq ravishda “Yiqildi” deb baholanishi yoki aslida kamroq baho olishi kerak bo’lganida yuqori baho olishi mumkin. Shuning uchun shart operatorlarini yozishda qamrov oralig’ini, ya’ni >=, >,
```

<= kabi belgilarni diqqat bilan tanlash kerak. Masalan, 85 ball to’plagan talaba “A” yoki “B” bahosini olishi, shartlarning qanday yozilishiga bog’liq. Shu sababli, bunday tizimlarni ishlab chiqishda matematik intervallarni diqqat bilan tahlil qilish zarur.

Dasturiy mantiq va optimallashtirish nuqtai nazaridan, baholash algoritmi odatda juda tez ishlaydi. Biroq, shartlar ko’paygan sari ularni mantiqiy guruhlash va soddalashtirish foydali bo’ladi. Masalan, agar bakalavr va magistr talabalariga turli mezonlar qo’llanilsa, bu mezonlarni alohida funksiyalar yoki klasslarga ajratib yozish, kodni yanada tushunarli va boshqarish oson qiladi.

Xulosa qilib aytganda, imtihon natijalarini baholash tizimi shart operatorlarining amaliy qo’llanilishi uchun ajoyib misol bo’lib xizmat qiladi. Ushbu model orqali talabalar nafaqat Python sintaksisini, balki haqiqiy hayotdagi baholash jarayonining mantiqiy tuzilishini ham chuqurroq tushunishlari mumkin.

Qaror qabul qilish algoritmlarida shart operatorlarining o‘rni va optimallashtirish masalalari

Dasturiy tizimlarning asosini qaror qabul qilish algoritmlari tashkil etadi, chunki ular ma’lumotlarni tahlil qiladi, solishtiradi va oxirgi natijani belgilaydi. Shart operatorlari esa bu jarayonning eng muhim qismi bo’lib, Pythonda qaror qabul qilishni oddiy, tushunarli va aniq qiladi.

Algoritmning samaradorligi ko’pincha shartlar qanday joylashtirilganiga, qaysi shart birinchi tekshirilishiga va har bir shartning mantiqiy tuzilishiga bog’liq. Shuning uchun “if-elif-else” bloklarini to’g’ri tashkil qilish dasturni optimallashtirishning muhim jihatidir.

Qaror qabul qilish jarayonida shart operatorlari turli imkoniyatlarni tekshiradi va bir nechta natijalardan birini tanlaydi. Masalan, foydalanuvchi ma’lumot kiritganda, dastur uni shunday tahlil qiladi:

· Agar qiymat to’g’ri bo’lsa – qabul qiladi
· Noto’g’ri bo’lsa – xatolik ko’rsatadi
· Ma’lumot yetarli bo’lmasa – qo’shimcha so’raydi

Bularning hammasi qaror daraxtiga o’xshash mantiqiy tuzilma bo’lib, shart operatorlari shu daraxtning asosiy tugunlaridir.

Muammo murakkablashgan sari shart operatorlari soni ham oshadi. Bu holatda optimallashtirish juda muhim ahamiyat kasb etadi. Eng ko’p uchraydigan yoki eng muhim shartlarni yuqoriga yozish, keraksiz tekshiruvlarni olib tashlash, takrorlanuvchi mantiqlarni alohida funksiyalarga ajratish – bularning barchasi algoritmning tezligi va samaradorligini oshiradi. Python interpretatori shartlarni ketma-ket tekshiradi, shuning uchun shartlarning tartibi dastur ishlash tezligiga bevosita ta’sir qiladi.

Quyidagi oddiy misolga qarang – bu yerda foydalanuvchi yoshini tahlil qilamiz:

```python
Yosh = int(input("Yoshingizni kiriting: "))

If yosh < 0:
Print("Noto'g'ri qiymat")
Elif yosh < 18:
Print("Kichik yoshdagi foydalanuvchi")
Elif yosh < 60:
Print("Voyaga yetgan foydalanuvchi")
Else:
Print("Katta yoshdagi foydalanuvchi")
```

Bu yerda shartlar eng kichikdan kattaga qarab tartiblangan – bu optimallashtirilgan ketma-ketlikdir. Agar foydalanuvchi 10 yosh kiritsa, dastur faqat birinchi shartni tekshiradi va natijani chiqaradi, qolgan shartlarni o'qimaydi – bu esa samaradorlikni oshiradi.

Obyektga yo'naltirilgan dasturlashda qaror qabul qilish algoritmlari yanada modulli shaklga kiradi. Har bir obyekt o'z xususiyatlari va metodlari orqali qaror chiqaradi. Masalan, "Foydalanuvchi" klassida "tasnifla()" metodi bo'lishi mumkin. Bu metod yosh, mijoz statusi, faoliyat turi yoki boshqa atributlarga qarab qaror chiqaradi. OOP yondashuvi mantiqni bo'limlarga ajratadi, bu esa modellashtirishni ancha tartibli va boshqarishni osonlashtiradi.

Qaror qabul qilish tizimlarida xatolarga yo'l qo'ymaslik juda muhim. Noto'g'ri shart yozilishi yoki noto'g'ri tartibda joylashtirilishi algoritmni butunlay ishdan chiqarishi mumkin. Shuning uchun tajribali dasturchilar qaror daraxtlarini avval diagramma shaklida chizishadi, keyin kodga o'tkazadilar. Bu jarayon mantiqiy tuzilma xatolarini kamaytiradi va murakkab qaror jarayonini soddalashtiradi. Shart operatorlari tashqi ko'rinishda juda oddiy bo'lishiga qaramay, ular to'g'ri tashkil etilganda kuchli qaror qabul qilish tizimini yaratadi. Hozirgi kunda sug'urta tizimlari, bank scoring modullari, baholash tizimlari, tavsiya algoritmlari va hatto tibbiy tashxis qo'yish jarayonlarida ham shart konstruktsiyalari asosiy mantiqiy mexanizm sifatida keng qo'llaniladi.

Xulosa

Ushbu maqolada shart operatorlarining dasturlashdagi ahamiyati, ular orqali real hayot jarayonlarini qanday modellashtirish mumkinligi va Python tilida qaror qabul qilish mexanizmlarini yaratishning amaliy jihatlari har tomonlama o'rganildi. "If-elif-else" tarkibiy qismlari yordamida imtihon baholash, foydalanuvchilarni tasniflash, kredit berishni tasdiqlash kabi turli amaliy vaziyatlarni avtomatlashtirish misollari

bilan ko'rsatildi. Obyektga yo'naltirilgan dasturlash yondashuvi esa shart operatorlarini yanada tizimli, modulli va kelajakda kengaytirishga qulay shaklda qo'llash imkoniyatini berishi ta'kidlandi. Algoritmlarni samarali ishlashini ta'minlashda shartlarning to'g'ri ketma-ketligi, mantiqiy chegaralarni aniq belgilash va keraksiz tekshiruvlardan qochish kabi optimallashtirish usullarining ahamiyati alohida ta'kidlandi. Keltirilgan kod namunalari, sxemalar va nazariy tushuntirishlar shuni isbotladiki, shart operatorlari nafaqat dasturiy mantiqning asosiy tarkibiy qismi, balki haqiqiy hayotni raqamli shaklda ifodalashda eng ishonchli vositalardan biridir. Natijada, ushbu mavzu ham dasturlashni endi o'rganayotganlar, ham murakkab tizimlar ustida ishlayotgan mutaxassislar uchun keng amaliy qo'llanilishiga ega.

Adabiyotlar ro‘yxati

1. Rasulxo‘jayev A., Tursunov Q. Dasturlash asoslari (Python misolida). Toshkent: Innovatsion rivojlanish nashriyoti, 2021.
2. Rahimov B. Algoritmlar va dasturlash tamoyillari. Toshkent: Fan va texnologiya, 2019.
3. Xolmirzayev U. Kompyuterlar va dasturlashning asosiy tushunchalari. Toshkent: TDYU nashriyoti, 2020.
4. Karimov A. Obyektga yo‘naltirilgan dasturlash asoslari. Toshkent: TATU nashriyoti, 2018.
5. Zelle J. Python Programming: An Introduction to Computer Science. Franklin, Beedle & Associates, 2016.
6. Lutz M. Learning Python. 5th Edition. O’Reilly Media, 2013.
7. Sweigart A. Automate the Boring Stuff with Python. No Starch Press, 2015. Havola: https://automatetheboringstuff.com
8. Python Software Foundation. Python Documentation – Conditional Statements. Havola: https://docs.python.org/3/tutorial/controlflow.html