ISSN 3060-4567

PYTHON DASTURLASH TILIDA VORISLIK

Tojimamatov Israiljon Nurmamatovich
Farg‘ona davlat unversiteti o‘qituvchisi
E-mail: ***israiltojimamatov@gmail.com***
Qo’shaqova Dilshoda Robiljon qizi
Farg‘ona davlat unversiteti talabasi
E-mail :qoshaqovadilshodahon@gmail.com

Annotatsiya:*Mazkur maqolada Python dasturlash tilida voris sinflar (inheritance) va voris metodlar (inherited methods) tushunchalari yoritilgan. Obyektga yo‘naltirilgan dasturlashning muhim jihatlaridan biri bo‘lgan vorislik mexanizmi orqali dasturiy ta'minotda kodni qayta ishlatish, kengaytirish va modul qilish imkoniyatlari tahlil qilingan. Misollar orqali bir nechta sinflar o‘rtasida meros olish, metodlarni qayta aniqlash (override) va super() funksiyasining vazifasi tushuntiriladi. Ushbu yondashuv zamonaviy dasturlashda strukturaviy va samarali tizimlar yaratishda muhim o‘rin egallaydi.*

Kalit so‘zlar: *Python, voris sinf, voris metod, obyektga yo‘naltirilgan dasturlash, super funksiyasi, kodni qayta ishlatish.*

Abstract:*This article explores the concepts of class inheritance and inherited methods in the Python programming language. Inheritance is a core feature of object-oriented programming, enabling code reuse, extension, and modularization. The study presents examples of class relationships, method overriding, and the use of the super() function. This approach is essential for building structured and efficient software systems in modern development practices.*

Keywords: *Python, inheritance, methods, object-oriented programming, super function, code reuse.*

Аннотация:*В данной статье рассматриваются понятия наследования классов (inheritance) и унаследованных методов (inherited methods) в языке программирования Python. Наследование является важной частью объектно-*

ориентированного программирования, обеспечивающей повторное использование и расширение кода. Приведены примеры реализации наследования между классами, переопределения методов и использования функции super(). Такой подход играет ключевую роль в построении структурированных и эффективных программных решений.

Ключевые слова: *Python, наследование, методы, объектно-ориентированное программирование, функция super, повторное использование кода.*

Kirish

Hozirgi zamon dasturlash tillari orasida Python o‘zining soddaligi, o‘qish osonligi va kuchli obyektga yo‘naltirilgan imkoniyatlari bilan ajralib turadi. Ayniqsa, murakkab tizimlarni modullashtirish, ularni sodda va strukturaviy ko‘rinishda ifodalashda Python tilining obyektga yo‘naltirilgan xususiyatlari muhim o‘rin tutadi. Ushbu xususiyatlar ichida **vorislik (inheritance)** va **voris metodlar (inherited methods)** tushunchasi dasturiy ta’minotni loyihalashda asosiy tushunchalardan biri hisoblanadi.

Vorislik orqali dasturda mavjud sinfga asoslanib yangi sinf yaratish, kodni qayta ishlatish, uning ustida kengaytma kiritish imkoniyati yaratiladi. Bu nafaqat dasturiy kodni soddalashtiradi, balki dasturiy tizimni modullashtirish, texnik xizmat ko‘rsatish va kengaytirish jarayonlarini ham yengillashtiradi. Mazkur maqolada vorislik tushunchasining nazariy asoslari, Python dasturlash tilida uni amalga oshirish usullari, metodlarni qayta aniqlash (override), super() funksiyasi va ko‘p vorislik (multiple inheritance) kabi jihatlar chuqur tahlil qilinadi. Bu orqali dasturchilar vorislik imkoniyatlaridan samarali foydalanish va kod sifatini oshirishga qaratilgan yondashuvlarni o‘zlashtirishlari mumkin.

Obyektga yo‘naltirilgan dasturlash (OOP — Object-Oriented Programming) — bu dasturlash paradigmasi bo‘lib, u murakkab dasturiy tizimlarni modullarga ajratish, har bir modulni obyekt sifatida ko‘rib chiqish, va ularni sinflar (class) asosida modellashtirishga asoslanadi. OOPning to‘rt asosiy prinsipi mavjud: **inkapsulyatsiya**, **abstraktsiya**, **vorislik** va **polimorfizm**. Ushbu maqolada ular

ISSN 3060-4567

ichida vorislik (inheritance) va voris metodlar (inherited methods) konseptual asos sifatida chuqur tahlil qilinadi.

Vorislik tushunchasi

Vorislik — bu bir sinf (ota sinf, yoki **superclass**) boshqa sinfga (voris sinf, yoki **subclass**) o‘zining atribut va metodlarini meros qilib berishidir. Bunda voris sinf ota sinfning barcha funksiyalaridan foydalanadi yoki ularni o‘z ehtiyojiga ko‘ra qayta aniqlaydi (override). Vorislik orqali dasturiy loyihalarda kodni qayta ishlatish, modul tuzilmalarni yaratish va murakkab tizimlarni oddiy sinflar orqali boshqarish mumkin bo‘ladi.

Python dasturlash tilida vorislik quyidagi sintaksis orqali amalga oshiriladi:

```
class Transport:
    def harakatlan(self):
        print("Harakatlanmoqda...")
class Avtomobil(Transport):
    def signal(self):
        print("Bip-bip!")
```

Bu yerda Avtomobil sinfi Transport sinfidan voris olmoqda. U Transport sinfidagi harakatlan() metodidan foydalanishi mumkin, shu bilan birga o‘ziga xos signal() metodiga ham ega.

Metodlarni qayta aniqlash (Method Overriding)

Voris sinf o‘zining ota sinfidagi metodni o‘z ehtiyojiga ko‘ra qayta aniqlashi mumkin. Bu dasturda polimorfizmni ta’minlashga xizmat qiladi.

```
class VorisSinf(OtaSinf):
    def salom(self):
        print("Men voris sinfman!")
```

Bundan tashqari, super() funksiyasi yordamida voris sinf ota sinfning metodlarini chaqira oladi. Bu, ayniqsa, konstruktorlar (ya’ni __init__ metodlari) bilan ishlaganda muhim ahamiyatga ega.

```
class Ota:
    def __init__(self, ism):
```

ISSN 3060-4567

```
        self.ism = ism
class Bola(Ota):
    def __init_ _(self, ism, yosh):
        super().__init__(ism)
        self.yosh = yosh
```

Python tilida ko‘p vorislik (multiple inheritance) ham qo‘llab-quvvatlanadi, ya’ni bir sinf bir nechta sinfdan meros olishi mumkin. Bu yondashuv ehtiyotkorlik bilan qo‘llanilishi kerak, chunki metodlar zanjiri (MRO — Method Resolution Order) muammolariga olib kelishi mumkin.

Vorislik dasturiy ta'minotning modulligini oshirish, takrorlanadigan kodni kamaytirish va loyihani texnik xizmat ko‘rsatishga qulaylashtirish kabi afzalliklarga ega. Bu esa dasturiy arxitekturaning barqarorligi va rivojlanishiga xizmat qiladi.

Vorislikning turlari

Python tilida vorislikning quyidagi turlari mavjud:

1. Oddiy vorislik (Single Inheritance):Faqat bitta ota klassdan voris olish.

```
class Hayvon:
    def tovush(self):
        print("Hayvon tovushi")
class It(Hayvon):
    pass
```

2.Ko‘p martalik vorislik (Multiple Inheritance):Bir nechta ota klasslardan meros olish.

```
class A:
    pass
class B:
    pass
class C(A, B):  # C klassi A va B dan voris olgan
    pass
```

Python’da bu holatda Method Resolution Order (MRO) asosida metodlar tartibi aniqlanadi.

ISSN 3060-4567

3. Ko‘p bosqichli vorislik (Multilevel Inheritance):Farzand klass o‘zi ham boshqa klassga ota bo‘ladi.

```
class A:
    pass
class B(A):
    pass
class C(B):
    pass
```

4.Gibrid vorislik (Hybrid Inheritance):Yuqoridagi turli vorislik turlarining aralashmasi.

super() funksiyasi va metodlar zanjiri

Python dasturlash tilida super() funksiyasi yordamida farzand klassdan ota klassning metodlariga to‘g‘ridan-to‘g‘ri murojaat qilish mumkin. Bu, ayniqsa, konstruktorlarni chaqirishda juda foydalidir.

```
class Ota:
    def init(self):
        print("Ota klass")
class Bola(Ota):
    def init(self):
        super().init()  # Ota klass konstruktorini chaqiradi
        print("Bola klass")
```

Metod va atributlarning ustuvorligi

Agar farzand klassda ota klassdagi metodlar qayta aniqlansa, ular ustunlikka ega bo‘ladi. Bu xususiyatga polimorfizm deyiladi.

```
class A:
    def x(self):
        print("A")
class B(A):
    def x(self):
        print("B")
```

```
b = B()
b.x()  # Natija: B
```

Bunday holatda farzand klassdagi metod avtomatik ravishda ota klassdagi metodni almashtiradi. Ammo istalgan vaqtda super() yordamida ota klassdagi metodni chaqirish mumkin.

Vorislikning afzalliklari

Obyektga yo‘naltirilgan dasturlash (OOP) konsepsiyasida vorislik (ing. inheritance) tushunchasi muhim markaziy o‘rinni egallaydi. Vorislik — bu mavjud klass asosida yangi klass yaratish imkonini beruvchi dasturlash mexanizmi bo‘lib, yangi klass avvalgi klassning barcha atribut va metodlarini o‘zlashtiradi. Bunday yondashuv dasturiy ta’minotni modullashtirish, kengaytirish va texnik xizmat ko‘rsatishni yengillashtirishda muhim rol o‘ynaydi.Vorislikdan foydalanish dasturiy tizimlarda ko‘p marotaba takrorlanadigan kodlardan xalos bo‘lish, umumiy funksionallikni bitta markazlashtirilgan ota klassda jamlash va uni boshqa maxsus klasslarda qayta ishlatish imkonini beradi. Bu esa dasturchiga nafaqat vaqt va kuchni tejash, balki kodni tushunarli, soddalashgan va texnik xizmat ko‘rsatishga yaroqli holatda saqlash imkonini beradi.Vorislikning muhim afzalliklaridan biri — kodning qayta ishlatilishidir. Bu degani, biror klassdagi funksiyalar yoki atributlar boshqa klasslar uchun ham ayni holatda xizmat qilishi mumkin. Misol uchun, “Hayvon” nomli umumiy klassda harakat, ovqatlanish kabi umumiy metodlar bo‘lishi mumkin; ushbu klassdan voris olgan “It”, “Mushuk”, “Quyon” kabi klasslar esa ushbu umumiy metodlarni o‘zida aks ettiradi va zarur hollarda ularni o‘zgartirib yoki kengaytirib ishlatadi.Bundan tashqari, vorislik orqali dastur modullashtiriladi — ya’ni kod mantiqiy bo‘linmalarga ajratiladi. Har bir klass o‘ziga tegishli vazifani bajaradi va bu modullar o‘zaro ierarxik aloqada bo‘ladi. Bunday strukturaviy yondashuv murakkab dasturiy tizimlarni soddalashtiradi va har bir komponentni alohida test qilish imkonini beradi.Yana bir muhim jihat — moslashuvchanlik va kengaytiruvchanlikdir. Dasturiy tizimlar doimiy rivojlanib boradi, yangi talablar qo‘yiladi. Vorislik bu jarayonni muammosiz amalga oshirishga yordam beradi. Agar yangi xususiyat yoki metod kerak bo‘lsa, mavjud klassni o‘zgartirmasdan, undan yangi klass hosil qilish mumkin.

Bu prinsip dasturiy arxitekturada ochiq–yopiq printsipi (open-closed principle) deb ataladi: “dastur kengaytirishga ochiq, lekin o‘zgartirishga yopiq bo‘lishi kerak.”Shuningdek, vorislikning asosida polimorfizm yotadi. Polimorfizm shuni anglatadiki, bir xil nomdagi metod har bir farzand klassda o‘ziga xos tarzda ishlashi mumkin. Bu orqali bitta interfeysga ega bo‘lgan turli klasslar har xil harakat qilishi ta’minlanadi. Dasturiy nuqtai nazardan, bu murakkab mantiqlarni soddalashtirish va ko‘p holatlarni bitta funksiya orqali boshqarish imkonini beradi.

Vorislikning foydali jihatlaridan yana biri — texnik xizmat ko‘rsatishning soddalashuvidir. Agar tizimda muayyan xatolik bo‘lsa, bu odatda ota klassdagi umumiy metodda yuzaga chiqadi va uni tuzatish farzand klasslarda ham avtomatik tarzda aks etadi. Bu esa tizimni nazorat qilishni soddalashtiradi, xatoliklarni aniqlash va ularni tuzatish vaqtini qisqartiradi.Barcha yuqoridagi afzalliklar vorislik mexanizmini nafaqat texnik, balki metodologik jihatdan ham dasturlashning eng muhim vositalaridan biriga aylantiradi. Ayniqsa, yirik dasturiy loyihalarda, modul tizimlarda, interfeysga ega platformalarda, foydalanuvchi rollarini boshqarishda va real dunyo obyektlarini model qilishda vorislik konsepsiyasi alohida o‘rin egallaydi.

Masalan, ta’lim tizimlari uchun ishlab chiqilayotgan platformada umumiy Foydalanuvchi klassi yaratiladi. Bu klassdan O‘qituvchi, O‘quvchi, Administrator kabi maxsus rollar uchun farzand klasslar voris oladi. Har bir rol o‘ziga xos huquq va imkoniyatlarga ega bo‘ladi. Shuningdek, tijorat ilovalari (masalan, savdo boshqaruvi)da Mahsulot, Elektron mahsulot, Jismoniy mahsulot kabi klasslar Mahsulot nomli ota klassdan voris olishi mumkin. Bunday struktura biznes tizimni modullashtirishda juda foydali.

Xulosa

Zamonaviy dasturlash paradigmasi bo‘lgan obyektga yo‘naltirilgan dasturlash (OOP) konsepsiyasi dasturiy mahsulotlarni modullashtirish, kengaytirish va boshqarishni sezilarli darajada yengillashtiradi. Ushbu paradigmaning asosiy tamoyillaridan biri – vorislik (inheritance) dasturlash jarayonida klasslar o‘rtasidagi ierarxik tuzilmani shakllantirish, kodni qayta ishlatish va tizimni evolyutsion rivojlantirish imkonini beradi.

Ushbu maqolada Python dasturlash tilida vorislik tushunchasi nazariy va amaliy jihatdan batafsil yoritildi. Python tilida klasslar orqali vorislikni amalga oshirish sintaktik jihatdan sodda, lekin imkoniyatlar nuqtayi nazaridan kengdir. Maqolada vorislik turlari (oddiy, ko‘p martalik, ko‘p bosqichli va gibrid vorislik), super() funksiyasining ahamiyati, metod ustuvorligi, polimorfizm bilan bog‘liq mexanizmlar va ular orqali OOP tamoyillarining to‘liq ifodalanishi ko‘rib chiqildi.

Shuningdek, maqolada keltirilgan amaliy masala orqali vorislik mexanizmlarining dasturda qanday ishlashi aniq misol bilan ko‘rsatib berildi. Bu esa mavzuni nafaqat nazariy, balki amaliy nuqtai nazardan ham chuqur anglash imkonini beradi.Umuman olganda, vorislik OOP asosida ishlab chiqilayotgan dasturiy ta’minotlar uchun funksionallikni kengaytirish, modul tuzilmalarni yaratish va murakkab tizimlarni soddalashtirib boshqarish vositasi sifatida katta ahamiyatga ega. Python dasturlash tilining soddaligi va moslashuvchanligi esa bu imkoniyatlardan to‘liq foydalanish uchun keng yo‘l ochadi.

FOYDALANILGAN ADABIYOTLAR RO‘YHATI

1. Lutz, M. (2013). Learning Python (5th ed.). O'Reilly Media.
2. Martelli, A., Ravenscroft, A., & Ascher, D. (2005). Python Cookbook. O'Reilly Media.
3. Downey, A. (2015). Think Python: How to Think Like a Computer Scientist. Green Tea Press.
4. Hetland, M. L. (2017). Beginning Python: From Novice to Professional (3rd ed.). Apress.
5. Pilgrim, M. (2004). Dive Into Python. Apress.
6. Van Rossum, G., & Drake, F. L. (2009). The Python Language Reference Manual. Network Theory Ltd.
7. Python Software Foundation. (2024). Python 3.12 documentation. Retrieved from https://docs.python.org/3.
8. Abdurahmonov, A. (2021). Python dasturlash tili asoslari. Toshkent: Fan va texnologiya nashriyoti.

ISSN 3060-4567

9. Nishonov, S. va boshqalar. (2020). Dasturlash asoslari (Python misolida). Toshkent: O‘zbekiston Respublikasi Oliy va o‘rta maxsus ta’lim vazirligi.

10. To‘xtayev, N. (2022). Algoritmlar va Python dasturlash tili. Samarqand: SamDU nashriyoti.