PYTHON DASTURLASH TILIDA VORISLIK

Tojimamatov Israil Nurmamatovich
Farg‘ona davlat universiteti katta o‘qituvchisi
israiltojimamatov@gmail.com
Komilova Zulxumor Xokimovna
Farg’ona davlat unversiteti o‘qituvchi
Omonaliyeva Elinur Umidbek qizi
Farg‘ona davlat unversiteti talabasi
omonaliyevaelinur@gmail.com

Annotatsiya *Ushbu maqolada Python dasturlash tilida vorislik turlari tushuntirilgan. Vorislik, bir sinfning boshqa sinfdan atributlar va metodlarni meros qilib olishiga imkon beradi. Maqolada yagona, ko‘p, ko‘p darajali, ierarxik va gibrid vorislik turlari ko‘rib chiqiladi. Har bir tur sinflar o‘rtasidagi bog‘lanishlarni va kodni qayta ishlatish imkoniyatlarini taqdim etadi, bu esa dastur dizaynini soddalashtiradi va samaradorligini oshiradi.*

Kalit so‘zlar: *obyektga yo‘naltirilgan dasturlash, modullikni, ierarxik, Single Inheritance, Multiple Inheritance, Multilevel Inheritance, Hierarchical Inheritance.*

Annotatsion *This article explains the types of inheritance in Python programming language. Inheritance allows one class to inherit attributes and methods from another class. The article covers the types of inheritance: single, multiple, multilevel, hierarchical, and hybrid. Each type establishes relationships between classes and provides opportunities for code reuse, which simplifies program design and enhances efficiency.*

Keywords: *object-oriented programming, modularity, hierarchical, Single Inheritance, Multiple Inheritance, Multilevel Inheritance, Hierarchical Inheritance.*

Аннотация *В данной статье объясняются типы наследования в языке программирования Python. Наследование позволяет одному классу наследовать атрибуты и методы другого класса. В статье рассматриваются типы наследования: одноуровневое, множественное, многоуровневое, иерархическое и гибридное. Каждый тип устанавливает связи между классами и предоставляет возможности для повторного использования кода, что упрощает проектирование программы и повышает её эффективность.*

Ключевые слова: *объектно-ориентированное программирование, модульность, иерархичность, одиночное наследование, множественное наследование, многоуровневое наследование, иерархическое наследование.*

Vorislik — obyektga yo‘naltirilgan dasturlash (OYD)da asosiy tushuncha bo‘lib, bu bir sinfga (farzand yoki ost sinf deb ataladi) boshqa sinfdan (ota yoki asosiy sinf deb ataladi) atributlar va metodlarni meros qilib olish imkonini beradi. Bu koddan qayta foydalanishni, modullikni va ierarxik sinf strukturasini rivojlantiradi. Ushbu maqolada biz Pythondagi vorislikni o‘rganamiz. Vorislik bizga boshqa sinfdan barcha metodlar va xususiyatlarni meros qilib olishni ta’minlaydigan sinfni aniqlash imkonini beradi.

```python
# Ota klass
class Hayvon:
    def __init__(self, nomi):
        self.nomi = nomi  # Nomi atributini boshlash

    def gapir(self):
        pass  # Farzand klasslar tomonidan o‘zgartirilishi kerak bo‘lgan joy

# Hayvondan meros olgan farzand klassi
class It(Hayvon):
    def gapir(self):
        return f"{self.nomi} havlaydi!"  # gapir metodini o‘zgartirish

# It klassidan obyekt yaratish
it = It("Buddy")
print(it.gapir())
```

Hayvon sinfi asosiy sinf sifatida ishlaydi va nomi atributini boshlaydi. gapir metodini esa ost sinflar o‘zgartirishi kerak. It sinfi esa Hayvon sinfidan meros oladi va gapir metodini o‘zgartiradi, bu yerda It sinfi "havlaydi!" deb qaytaradi. it nomli obyekt It sinfidan yaratilgan va gapir metodini chaqirib, "Buddy havlaydi!" deb chiqaradi.

Chiqish:

Buddy havlaydi!

Hayvon—asosiy sinf bo‘lib, unda __init__ va gapir metodlari mavjud. It esa Hayvon sinfidan meros olgan ost sinfi. gapir metodining It sinfida o‘zgartirilgan versiyasi aniq xulq-atvorni taqdim etadi.

Endi vorislikni batafsil tushunib olaylik:

Ota Klass:

```python
class OtaKlass:

    # Ota klassining kodi bu yerga yoziladi
    pass
```

Farzand Klass:

```python
class FarzandKlass(OtaKlass):

    # Farzand klassining kodi bu yerga yoziladi
    pass
```

Otasinf:Bu — boshqa sinflar meros oladigan asosiy sinfdir. U farzand sinfi qayta foydalanishi mumkin bo‘lgan atributlar va metodlarni o‘z ichiga oladi.

Ostsinf:Bu—otasinfdan meros olgan olingan sinfdir. Vorislik sintaksisi quyidagicha bo‘ladi: class FarzandSinf(OtaSinf). Farzand sinfi avtomatik ravishda ota sinfning barcha atributlari va metodlarini oladi, agar ular o‘zgartirilmagan bo‘lsa.

Obyektga yo‘naltirilgan dasturlashda, ota sinf (yoki asosiy sinf deb ataladi) boshqa sinflar tomonidan meros qilib olinadigan umumiy atributlar va metodlarni belgilaydi. Ushbu atributlar va metodlar ost sinflari uchun asos bo‘lib xizmat qiladi. Vorislikdan foydalangan holda, ost sinflari ota sinf tomonidan taqdim etilgan funksionallikka kirish imkoniyatiga ega bo‘ladi va uni kengaytirishi mumkin. Masalan, Person (Shaxs) ota sinfi sifatida ishlatilgan bir misol:

```python
# Merosiylikni ko‘rsatish uchun Python dasturi
class Shaxs(object):

  # Konstruktor
  def __init__(self, ism, id):
    self.ism = ism
    self.id = id

  # Ushbu shaxs xodim ekanligini tekshirish
  def Ko‘rsatish(self):
    print(self.ism, self.id)


# Dastur kodining ishga tushirilishi
xodim = Shaxs("Satyam", 102) # Shaxsning obyekti
xodim.Ko‘rsatish()
```

Chiqish:

```
Satyam 102
```

Shaxs sinfi ikkita atributga ega: ism va id. Bu atributlar sinfdan obyekt yaratilganda o‘rnatiladi. Ko‘rsatish metodi shaxsning ism va id ma’lumotlarini ekranga chiqaradi.

Farzand sinfi (yoki ostsinf) — bu ota sinfdan atributlar va metodlarni meros qilib oladigan sinfdir. Farzand sinfi shuningdek qo‘shimcha atributlar va metodlar kiritishi yoki ota sinfdan olingan metodlarni o‘zgartirishi mumkin.

Ushbu misolda, Emp — Shaxs sinfidan meros olgan ost sinfidir.

```python
class Emp(Shaxs):

  def Chiqarish(self):
    print("Emp klassi chaqirildi")

Emp_ma'lumotlari = Emp("Mayank", 103)

# Ota klass funksiyasini chaqirish
Emp_ma'lumotlari.Ko‘rsatish()

# Farzand klass funksiyasini chaqirish
Emp_ma'lumotlari.Chiqarish()
```

Emp sinfi Shaxs sinfidan ism va id atributlari hamda Ko‘rsatish metodini meros qilib oladi.

Emp sinfidagi __init__ metodi super().__init__(ism, id) orqali Shaxs sinfining konstruktorini chaqiradi va meros olingan atributlarni boshlaydi.

Emp sinfi qo‘shimcha atribut, ya’ni rolni kiritadi va shuningdek Ko‘rsatish metodini o‘zgartirib, ism va id bilan birga rolni ham chiqaradi.

__init__() funksiyasi Pythondagi konstruktor metodidir. U obyekt yaratilganda obyektning holatini boshlaydi. Agar farzand sinfi o‘zining __init__() metodini aniqlamasa, u avtomatik ravishda asosiy sinfdan meros qilib oladi. Yuqoridagi misolda, Employee sinfidagi __init__() metodi meros olingan va yangi atributlarning to‘g‘ri boshlanishini ta’minlaydi.

```python
# Ota klass: Shaxs
class Shaxs:
    def __init__(self, ism, idraqam):
        self.ism = ism
        self.idraqam = idraqam

# Farzand klass: Xodim
class Xodim(Shaxs):
    def __init__(self, ism, idraqam, maosh, lavozim):
        super().__init__(ism, idraqam)  # Shaxs klassining __init__() metodini chaqiradi
        self.maosh = maosh
        self.lavozim = lavozim
```

Shaxs sinfidagi __init__() metodi ism va id raqam atributlarini boshlaydi. Xodim sinfidagi __init__() metodi super().__init__(ism, idraqam) orqali Shaxs sinfidan olingan ism va id raqam atributlarini boshlaydi va yangi maosh va lavozim atributlarini qo‘shadi. super() funksiyasi ota sinfining metodlarini chaqirish uchun ishlatiladi. Ayniqsa, bu ost sinfining __init__() metodida meros olingan atributlarni boshlash uchun keng qo‘llaniladi. Shunday qilib, ost sinfi asosiy sinfning funksionalligidan foydalanishi mumkin.

Misol:

```python
# Ota klass: Shaxs
class Shaxs:
    def __init__(self, ism, idraqam):
        self.ism = ism
        self.idraqam = idraqam

    def Ko‘rsatish(self):
        print(self.ism)
        print(self.idraqam)

# Farzand klass: Xodim
class Xodim(Shaxs):
    def __init__(self, ism, idraqam, maosh, lavozim):
        super().__init__(ism, idraqam)  # super() yordamida Shaxs klassining __init__() metodini
        self.maosh = maosh
        self.lavozim = lavozim
```

super() funksiyasi Xodim sinfining __init__() metodida Shaxs sinfining konstruktorini chaqirish uchun ishlatiladi va meros olingan atributlar (ism va idraqam)ni boshlaydi. Bu, asosiy sinfning funksionalligidan foydalanishni ta’minlaydi, ost sinfida kodni qayta yozishga hojat qolmaydi.

Vorislik o‘rnatilgandan so‘ng, asosiy va ost sinflari o‘zlariga xos atributlarga ega bo‘lishi mumkin. Atributlar — bu sinfga tegishli bo‘lgan xususiyatlar bo‘lib, ular ma’lumotlarni saqlash uchun ishlatiladi.

Misol:

```python
# Ota klass: Shaxs
class Shaxs:
    def __init__(self, ism, idraqam):
        self.ism = ism
        self.idraqam = idraqam

    def Ko‘rsatish(self):
        print(self.ism)
        print(self.idraqam)

# Farzand klass: Xodim
class Xodim(Shaxs):
    def __init__(self, ism, idraqam, maosh, lavozim):
        super().__init__(ism, idraqam)  # Shaxs klassining __init__() metodini chaqirish
        self.maosh = maosh
        self.lavozim = lavozim
```

Shaxs sinfi ism va id raqam atributlariga ega. Xodim sinfi maosh va lavozim atributlarini qo‘shadi. Atributlar obyekt yaratilganda boshlanadi va ular Shaxs va Xodim bilan bog‘liq aniq ma’lumotlarni ifodalaydi.

Python Vorislik Turlari:

1. Yagona Vorislik (Single Inheritance): Ost sinfi faqat bitta asosiy sinfdan meros oladi.
2. Ko‘p Vorislik (Multiple Inheritance): Ost sinfi bir nechta asosiy sinflardan meros oladi.
3. Ko‘p Darajali Vorislik (Multilevel Inheritance): Bir sinf boshqa bir sinfdan meros oladi, va shu sinf yana boshqa bir sinfdan meros oladi.
4. Ierarxik Vorislik (Hierarchical Inheritance): Bir nechta sinflar yagona asosiy sinfdan meros oladi.
5. Gibrid Vorislik (Hybrid Inheritance): Bir nechta turdagi vorisliklarni birlashtirish.

Misol:

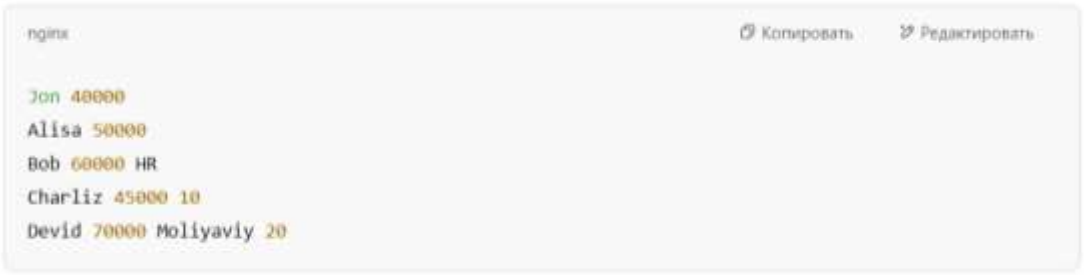
```
# 1. Yagona Vorislik (Single Inheritance)
class Shaxs:
    def __init__(self, ism):
        self.ism = ism
class Xodim(Shaxs):  # Xodim Shaxsdan meros oladi
    def __init__(self, ism, maosh):
        super().__init__(ism)
        self.maosh = maosh
```

```
# 2. Ko‘p Vorislik (Multiple Inheritance)
class Ish:
    def __init__(self, maosh):
        self.maosh = maosh
class XodimShaxsIsh(Xodim, Ish):  # Xodim va Ishdan meros oladi
    def __init__(self, ism, maosh):
        Xodim.__init__(self, ism, maosh)  # Xodimni boshlash
        Ish.__init__(self, maosh)       # Ishni boshlash
# 3. Ko‘p Darajali Vorislik (Multilevel Inheritance)
class Menejer(XodimShaxsIsh):  # XodimShaxsIshdan meros oladi
    def __init__(self, ism, maosh, bo‘lim):
        XodimShaxsIsh.__init__(self, ism, maosh)  # XodimShaxsIshni aniq boshlash
        self.bo‘lim = bo‘lim
# 4. Ierarxik Vorislik (Hierarchical Inheritance)
class YordamchiMenejer(XodimShaxsIsh):  # XodimShaxsIshdan meros oladi
    def __init__(self, ism, maosh, jamoa_hajmi):
        XodimShaxsIsh.__init__(self, ism, maosh)  # XodimShaxsIshni aniq boshlash
        self.jamoa_hajmi = jamoa_hajmi
# 5. Gibrid Vorislik (Multiple + Multilevel)
class KeksayganMenejer(Menejer, YordamchiMenejer):  # Menejer va YordamchiMenejerdan meros oladi
    def __init__(self, ism, maosh, bo‘lim, jamoa_hajmi):
        Menejer.__init__(self, ism, maosh, bo‘lim)  # Menejerni boshlash
        YordamchiMenejer.__init__(self, ism, maosh, jamoa_hajmi)  # YordamchiMenejerni boshlash
# Vorislikni ko‘rsatish uchun obyektlar yaratish
# Yagona Vorislik
xodim = Xodim("Jon", 40000)
print(xodim.ism, xodim.maosh)
# Ko‘p Vorislik
xodim2 = XodimShaxsIsh("Alisa", 50000)
print(xodim2.ism, xodim2.maosh)
# Ko‘p Darajali Vorislik
menejer = Menejer("Bob", 60000, "HR")
print(menejer.ism, menejer.maosh, menejer.bo‘lim)
# Ierarxik Vorislik
yordamchi_menejer = YordamchiMenejer("Charliz", 45000, 10)
print(yordamchi_menejer.ism,yordamchi_menejer.maosh,yordamchi_menejer.jamoa_hajmi)
# Gibrid Vorislik
```

keksaygan_menejer = KeksayganMenejer("Devid", 70000, "Moliyaviy", 20)

print(keksaygan_menejer.ism,keksaygan_menejer.maosh,keksaygan_menejer.bo‘lim, keksaygan_menejer.jamoa_hajmi)

Natija:



Vorislik turlari:

1. Yagona Vorislik (Single Inheritance): Xodim Shaxsdan meros oladi va unga maosh atributini qo‘shadi.
2. Ko‘p Vorislik (Multiple Inheritance): XodimShaxsIsh Xodim va Ishdan meros oladi, bu esa nom va maoshga kirish imkoniyatini beradi.
3. Ko‘p Darajali Vorislik (Multilevel Inheritance): Menejer XodimShaxsIshdan meros oladi, bu esa Xodim va Ishni ham o‘z ichiga oladi.
4. Ierarxik Vorislik (Hierarchical Inheritance): YordamchiMenejer ham XodimShaxsIshdan meros oladi, bu bir nechta bolalar sinflarining bitta ota sinfdan meros olishini ko‘rsatadi.
5. Gibrid Vorislik (Hybrid Inheritance): KeksayganMenejer Menejer (ko‘p darajali) va YordamchiMenejer (ierarxik)dan meros oladi, bu ikki turdagi vorislikni birlashtiradi.

Xulosa

Ushbu maqolada Python dasturlash tilidagi vorislikning turli shakllari – yagona, ko‘p, ko‘p darajali, ierarxik va gibrid vorislik – taqdim etilgan. Har bir vorislik turi sinflar o‘rtasidagi aloqalarni tashkil etib, kodni qayta ishlatish imkoniyatlarini yaratadi. Bu esa dastur dizaynini yaxshilash, uning optimallashuvini ta’minlash va kodning samarali boshqarilishini osonlashtiradi. Vorislikning turli turlari, shuningdek, dasturiy ta’minotning kengayishini qo‘llab-quvvatlab, modullilik va dasturiy ta’minotning moslashuvchanligini oshiradi.

FOYDALANILGAN ADABIYOTLAR

1. "Python Crash Course" - Eric Matthes
2. "Fluent Python" - Luciano Ramalho
3. "Learning Python" - Mark Lutz
4. Tojimamatov, I., & Mirsiddiqova, M. (2025). Berilganlar bazasida hayotiy sikl. Модели и методы в современной науке, 4(6), 66-70.

5. Tojimamatov, I., & Siddiqova, G. (2025). Tranzaksiyalarni taqsimlangan tarzda qayta ishlash modellari. Современные подходы и новые исследования в современной науке, 4(6), 30-35.
6. Нурмаматович, Т. И., & Рахила, А. (2025). На основе математических моделей повышение устойчивости к поломкам и авариям. Yangi o ‘zbekiston, yangi tadqiqotlar jurnali, 2(8), 197-204.
7. Tojimamatov, I., & Ahmataliyeva, S. (2025). Berilganlarni markazlashgan tarzda boshqarish tamoyillari. Академические исследования в современной науке, 4(21),59-64.
8. Tojimamatov, I., & Adxamova, C. (2025). Amaliy tizimlarda berilganlar bazasini boshqarish tizimlari o‘rni. Академические исследования в современной науке, 4(21), 77-82.
9. Tojimamatov, I., & Fazliddinov, X. (2025). Berilganlar bazasi administratori va uning xususyatlar. Академические исследования в современной науке, 4(21), 90-95.
10. Karimberdiyevich, O. M., Nurmamatovich, T. I., & Abdulaziz o‘g‘li, Y. M. (2024). Big data sohasidagi xalqaro loyihalar. Izlanuvchi, 1(1), 39-45.
11. Karimberdiyevich, O. M., Abdulaziz o‘g‘li, Y. M., & Zarifjon o‘g‘li, X. N. (2024). Data mining metodlari va bosqichlari. Yangi o ‘zbekiston, yangi tadqiqotlar jurnali, 1(4), 303-311.
12. Nurmamatovich, T. I. (2024). Berilganlarning tarmoq modellari: oddiy va murakkab tarmoq tuzilishlari.
13. "Python Object-Oriented Programming" - Steven F. Lott
14. "Mastering Object-Oriented Python" - Steven Lott

Foydalanilgan saytlar

1. https://www.geeksforgeeks.org/method-overriding-in-python/
2. https://realpython.com/
3. https://www.w3schools.com/python/
4. https://docs.python.org/3/tutorial/classes.html