Python: Boshlang'ichdan Mutaxassislikka

Avazbek Hazratov Obidjonovich

"Salom, men Avazbek Hazratov Obidjonovich. Ushbu kitob, Python dasturlash tilini o'rganib, endi ishga kirishni va intervyularda muvaffaqiyatli bo'lishni xohlaganlar uchun mo'ljallangan. Kitobda sizga kerakli barcha nazariy ma'lumotlar, savol-javoblar va ishga kirish uchun zarur ko'nikmalar berilgan. Python bo'yicha eng so'nggi tushunchalar, algoritmlar, dasturlash metodologiyalari va kod yozish uslublari haqida to'liq ma'lumot topasiz. Bu kitob nafaqat dasturlash bilimlarini oshirishga, balki ish suhbatlarida muvaffaqiyatli bo'lishga tayyorlashga yordam beradi."

"Python dasturlashni mukammal bilganingizni ko'rsatish uchun suhbatga tayyorlaning va o'z orzularingizdagi ishni qo'lga kiriting!"

Ma'lumot turlari (Data Types):

1. **int (Butun son):** Butun sonlar uchun ishlatiladi.
   - Misol: Do‘stingiz yoshini ifodalash uchun.
     "Men 15 yoshdaman."
2. **float (O‘nli son):** O‘nlik sonlar uchun ishlatiladi.
   - Misol: Yugurilgan masofa.
     "Bugun men 3.5 kilometr yugurdim."
3. **str (String):** Matnni ifodalash uchun ishlatiladi.
   - Misol: Ism yoki telefon raqami.
     "Mening ismim Avaz."
4. **bool (Boolean):** Ha yoki yo‘q (True/False) kabi mantiqiy qiymatlar uchun.
   - Misol: *"Chiroq yoqilganmi? Ha!"*
5. **list (Ro‘yxat):** Bir nechta narsani saqlash uchun ishlatiladi.
   - Misol: Xarid ro‘yxati.
     ["Non", "Sut", "Shakar"]
6. **tuple (O‘zgarmas ro‘yxat):** O‘zgarmaydigan narsalar ro‘yxati.
   - Misol: Tug‘ilgan sana.
     (2002, 9, 15)
7. **dict (Lug‘at):** Kalit-qiymat juftliklarini saqlash uchun.
   - Misol: Talabaning ma'lumoti.
     {"Ism": "Avaz", "Yosh": 15}

8. **set (To‘plam):** Takrorlanmaydigan qiymatlar to‘plami.
    - Misol: Do‘stlar o‘rtasidagi o‘yin natijalari.
      {10, 15, 20}

Ma'lumotlarni o‘zgartirish va ishlov berish (type, casting, slicing):

- ***type()****: Har qanday qiymatning turini aniqlash.*
  Misol: Do‘stingiz yoshi son yoki matn ekanligini bilish.
- ***Casting****: Ma'lumotlarni bir turdan boshqasiga o‘tkazish.*
  Misol: "15" matnini son qilish, chunki yoshni qo‘shish kerak.
- ***Slicing****: Ma'lumotlardan ma'lum qismini olish.*
  Misol: Telefon raqamidan faqat oxirgi 4 raqamni olish.
-

Operatorlar:

1. ***Aritmetik:*** *Qo‘shish, ayirish, ko‘paytirish, bo‘lish.*
    - *Misol: "Do‘kondan 2 non va 3 kg olma oldim. Hammasi qancha turadi?"*
2. ***Mantiqiy:*** *and, or, not.*
    - *Misol: "Uy vazifasi tayyor va xona tozalangan bo‘lsa, o‘yin o‘ynashga ruxsat."*
3. ***Taqqoslash:*** *>, <, ==, !=.*
    - *Misol: "Kimning bali katta?"*
4. ***Tayinlash:*** *=, +=, -= va boshqalar.*
    - *Misol: "Javonimdagi kitoblar sonini yangilab turish."*
5. ***is, in operatorlari:***
    - *is: Biror narsaning aynan o‘sha ekanligini tekshirish.*
    - *in: Elementning ro‘yxatda borligini aniqlash.*
      "Xarid ro‘yxatida sut bormi?"

Nazorat tuzilmalari:

1. ***if, elif, else:*** *Shartli tekshirish.*
    - *Misol: "Agar ob-havo yaxshi bo‘lsa, parkka boramiz; aks holda uyda qolamiz."*
2. ***for, while looplar:*** *Takroriy ishlarni bajarish.*
    - *Misol: "Barcha do‘stlaringizga SMS yuborish."*

Funktsiyalar:

1. ***def (Funktsiya yaratish):*** *Takroriy ishlarni bir joyga yig‘ish.*
    - *Misol: "Tug‘ilgan sanangizdan yoshingizni hisoblash."*
2. ***Lambda funksiyalar:*** *Tezkor va sodda hisoblashlar uchun.*
    - *Misol: "Yashash yilingizni aniqlash uchun."*
3. ***map, filter, reduce:***

- *map: Ro‘yxatdagi barcha elementlarga bir xil amalni qo‘llash.*
  "Narxlarni dollardan so‘mga o‘tkazish."
- *filter: Shartga mos keladigan elementlarni topish.*
  "Faqat juft sonlarni ajratish."
- *reduce: Ro‘yxatni bitta qiymatga qisqartirish.*
  "Hammasini qo‘shib, umumiy natijani olish."

2. Ob’ektga yo‘naltirilgan dasturlash (OOP)

OOP nima?

Ob’ektga yo‘naltirilgan dasturlash (OOP) — bu dasturlash paradigmasi bo‘lib, u dastur tuzilishini ob’ektlar atrofida tashkil qiladi. Ob’ektlar sinflar asosida yaratiladi va o‘ziga xos xususiyatlar (ma’lumotlar) va xatti-harakatlarga (usullar) ega bo‘ladi.

- ***Asosiy g‘oya:*** *Katta dasturlarni kichikroq, mustaqil bo‘limlarga ajratish va ular bilan mantiqiy ishlashni osonlashtirish.*
- ***Hayotiy misol:***
  Tasavvur qiling, siz bir nechta turdagi avtomobillarni boshqarishingiz kerak. Har bir avtomobil o‘ziga xos xususiyatlarga ega (model, rang, narx) va xatti-harakatlarni (tezlashish, to‘xtash) amalga oshiradi. OOP yordamida har bir avtomobilni alohida ob’ekt sifatida yaratib boshqarishingiz mumkin.

Asosiy tushunchalar: Sinf (class) va ob’ekt (object)

- ***Sinf (Class):***
  Sinf — bu ob’ektlar uchun shablon yoki andoza. U ob’ektlarning qanday xususiyatlari va funksiyalari bo‘lishini belgilaydi.
  - ***Hayotiy misol:***
    Zavod — "Avtomobil" sinfi bo‘lib, unda avtomobilning umumiy xususiyatlari (model, rang) va funksiyalari (tezlashish, signal chalish) belgilangan.
- ***Ob’ekt (Object):***
  Ob’ekt — bu sinfning aniq bir namunasi. Har bir ob’ekt o‘ziga xos qiymatlarga ega bo‘ladi.
  - ***Hayotiy misol:***
    Oq rangli "Toyota Corolla 2023" avtomobili — bu "Avtomobil" sinfining ob’ekti.

Konstruktorlar (__init__) va sinf usullari

- ***Konstruktor (__init__):***
  Bu sinfdan yangi ob’ekt yaratilganda avtomatik ishga tushadigan metod bo‘lib, ob’ektning boshlang‘ich xususiyatlarini sozlaydi.
  - ***Hayotiy misol:***
    Do‘kondan yangi telefon olganingizda, uni sozlash: tilni tanlash, vaqtni o‘rnatish kabi.
- ***Sinf usullari:***
  Sinfning barcha ob’ektlari uchun umumiy bo‘lgan funksiyalarni aniqlash uchun ishlatiladi.

    - ***Hayotiy misol:***
      Har bir avtomobil uchun yoqilg‘i sarfini hisoblash funksiyasi.

Polimorfizm, inkapsulyatsiya va meros olish

Polimorfizm (Polymorphism):

Bir xil nomdagi metodlarni turli sinflarda turlicha ishlatish imkoniyati.

- ***Hayotiy misol:***
  Hayvonlar turlicha tovush chiqaradi:
    - *Mushuk miyovlaydi.*
    - *It vovullaydi.*

Inkapsulyatsiya (Encapsulation):

Ma’lumot va funksiyalarni yashirish va faqat kerakli qismlarini foydalanuvchiga ko‘rsatish usuli.

- ***Hayotiy misol:***
  Telefoningizning ichki tizimi ko‘rinmaydi, lekin siz uning tugmalaridan foydalanib qo‘ng‘iroq qilishingiz mumkin.

Meros olish (Inheritance):

Bir sinfning xususiyat va funksiyalarini boshqa sinfga o‘tkazish imkoniyati.

- ***Hayotiy misol:***
  "Avtomobil" sinfi bor. "Sport avtomobili" sinfi undan meros qilib, qo‘shimcha xususiyatlarni qo‘shadi (masalan, yuqori tezlikda yurish).

@staticmethod va @classmethod

- ***@staticmethod:***
  Sinf bilan bog‘liq bo‘lmagan, lekin mantiqiy jihatdan unga tegishli bo‘lgan funksiyalarni aniqlash uchun ishlatiladi.
    - ***Hayotiy misol:***
      Zavod o‘z logotipini qayta ishlab chiqadi.
- ***@classmethod:***
  Sinfning o‘zini yoki sinfga tegishli ma’lumotlarni boshqaruvchi usullarni yaratish uchun ishlatiladi.
    - ***Hayotiy misol:***
      Zavod yangi standartlarni barcha avtomobillarga qo‘llashga qaror qiladi.

super() funksiyasidan foydalanish

- ***super()****:*
  Asosiy sinfning metodlarini yoki xususiyatlarini chaqirish uchun ishlatiladi.

- ***Hayotiy misol:***
  "Sport avtomobili" sinfi "Avtomobil" sinfidagi umumiy funksiyalardan foydalanadi va ularga o‘ziga xos o‘zgartirishlar qo‘shadi.

Dunder metodlari (__str__, __repr__, __eq__, va boshqalar)

__str__:

Ob’ektning foydalanuvchi uchun tushunarli ko‘rinishini beradi.

- ***Hayotiy misol:***
  Do‘stingiz haqida gapirganda: "Bu Avaz, u 22 yoshda."

__repr__:

Dasturchilar uchun ob’ektning texnik ko‘rinishini qaytaradi.

- ***Hayotiy misol:***
  Telefon texnik ma’lumotlari: "Samsung Galaxy S21, 128GB, Black."

__eq__:

Ikki ob’ektni tengligini solishtirish uchun ishlatiladi.

- ***Hayotiy misol:***
  Ikki avtomobilni solishtirish: "Bu ikkala avtomobil ham Toyota Corolla 2023 modeliga tegishli."

Boshqa dunder metodlari:

- ***__add__:*** *Ikki ob’ektni qo‘shadi.*
- ***__len__:*** *Ob’ekt uzunligini qaytaradi.*
- ***__getitem__:*** *Ob’ektning ma’lum bir elementiga murojaat qilish imkonini beradi.*

3. Kutubxonalar va ularning ishlatilishi

Python kutubxonalari murakkab funksiyalarni bajarishni osonlashtiradi va ko‘p vaqtni tejaydi. Quyida asosiy kutubxonalar va ularning qo‘llanilishi haqida kengroq ma’lumotlar keltirilgan.

Asosiy kutubxonalar

1. os — Operatsion tizim bilan ishlash

Bu kutubxona fayl tizimi va operatsion tizim bilan bog‘liq vazifalarni bajarishda yordam beradi.

- ***Imkoniyatlari:***

    - *Papkalar yaratish va o‘chirish.*
    - *Fayl va papka yo‘llari bilan ishlash.*
    - *Muhit o‘zgaruvchilariga (environment variables) murojaat qilish.*
- ***Hayotiy misol:***
  Yangi loyiha papkasi yaratish, unga fayllarni saqlash, yoki mavjud fayllarni avtomatik o‘chirish.

2. sys — Tizim parametrlari va buyruqlar bilan ishlash

Tizim darajasidagi funksiyalarni boshqarish imkonini beradi.

- ***Imkoniyatlari:***
    - *Buyruq satri parametrlarini olish.*
    - *Tizimda ishlayotgan platformani aniqlash.*
    - *Dasturni istalgan joyda to‘xtatish.*
- ***Hayotiy misol:***
  Dasturni turli parametrlar bilan ishga tushirib, ularning natijasini boshqarish.

3. datetime — Sana va vaqt bilan ishlash

Sana va vaqtni boshqarish uchun ishlatiladi.

- ***Imkoniyatlari:***
    - *Joriy vaqtni olish va ko‘rsatish.*
    - *Vaqtlar orasidagi farqni hisoblash.*
    - *Sanalarni ma’lum bir formatda ko‘rsatish.*
- ***Hayotiy misol:***
  Biror vazifa bajarilishi uchun qancha vaqt ketishini hisoblash yoki tugash muddatini aniqlash.

4. math — Matematik hisob-kitoblar

Murakkab matematik funksiyalarni bajarishda ishlatiladi.

- ***Imkoniyatlari:***
    - *Trigonometrik funksiyalar (sin, cos, tan).*
    - *Logarifmlar va eksponentlar hisoblash.*
    - *Kasrlar va ildizlarni olish.*
- ***Hayotiy misol:***
  Muayyan burchak ostida ob’ektning harakatini hisoblash.

5. random — Tasodifiylik bilan ishlash

Tasodifiy sonlar yoki elementlarni tanlashga yordam beradi.

- ***Imkoniyatlari:***
    - *Tasodifiy sonlar yaratish.*
    - *Ro‘yxatdan tasodifiy element tanlash.*
    - *Tasodifiy tanlovlar uchun ehtimollikni sozlash.*

- ***Hayotiy misol:***
  Lotereya o‘yinlarini yaratish yoki o‘yin ichidagi tasodifiy voqealarni belgilash.

Qo‘shimcha kutubxonalar

1. requests — API bilan ishlash

Veb xizmatlardan ma’lumotlarni olish va yuborishga imkon beradi.

- ***Imkoniyatlari:***
  - *GET va POST so‘rovlarini amalga oshirish.*
  - *API orqali ma’lumotlarni olish va jo‘natish.*
  - *Ma’lumotlarni JSON formatida qaytarish.*
- ***Hayotiy misol:***
  Ob-havo ma’lumotlarini onlayn xizmatlardan olish yoki yangiliklar bilan ishlash.

Ma’lumotlar tahlili uchun kutubxonalar

1. pandas — Ma’lumotlarni tahlil qilish va boshqarish

Ma’lumotlar bilan ishlashni osonlashtiruvchi eng mashhur kutubxonalardan biri.

- ***Imkoniyatlari:***
  - *Jadval ko‘rinishida ma’lumotlarni boshqarish.*
  - *Ma’lumotlarni saralash, guruhlash, va tozalash.*
  - *CSV va Excel fayllari bilan ishlash.*
- ***Hayotiy misol:***
  Talabalar ro‘yxatini CSV faylida saqlash va ular bo‘yicha statistik hisobot yaratish.

2. numpy — Matematik tahlil va massivlar bilan ishlash

Matematik hisob-kitoblar va katta hajmdagi massivlarni boshqarish uchun ishlatiladi.

- ***Imkoniyatlari:***
  - *Yuqori tezlikda matematik operatsiyalarni bajarish.*
  - *Katta massivlar yaratish va ularga ishlov berish.*
  - *Matritsa hisob-kitoblarini amalga oshirish.*
- ***Hayotiy misol:***
  Ko‘p o‘lchovli ma’lumotlarni boshqarish va statistik tahlilni tezkor amalga oshirish.

+==========+

Boshlang‘ich daraja (Beginner)

1. **Python qanday dasturlash tili va uning asosiy xususiyatlari nimalardan iborat?**
   Javob: Python yuqori darajali, universal maqsadli, oson o‘qiladigan dasturlash tili. Asosiy xususiyatlari:

- Oddiy va o‘qilishi oson sintaksis.
- Dinamik tiplangan, ya’ni o‘zgaruvchilar turini oldindan aniqlash talab qilinmaydi.
- Katta va boy standart kutubxona.

2. **Python’ning asosiy ma’lumot turlari qaysilar?**
   Javob: Python asosiy ma’lumot turlari:
   - **int**: Butun sonlarni ifodalaydi.
   - **float**: Haqiqiy (kasrli) sonlarni ifodalaydi.
   - **str**: Qatorlar, ya’ni matnlarni ifodalaydi.
   - **bool**: Mantiqiy qiymatlar (True yoki False).
   - **list**: O‘zgartiriladigan elementlar ro‘yxati.
   - **tuple**: O‘zgartirilmaydigan elementlar ro‘yxati.
   - **dict**: Kalit-qiymat juftliklari to‘plami.
   - **set**: Noyob elementlarning tartiblanmagan to‘plami.
3. **O‘zgaruvchi (variable) nima va Python’da qanday aniqlanadi?**
   Javob: O‘zgaruvchi ma’lumotlarni saqlash uchun ishlatiladi va qiymatni nom bilan belgilash orqali aniqlanadi.
4. **Python’da qanday operatorlar mavjud?**
   Javob:
   - Aritmetik operatorlar (qo‘shish, ayirish, ko‘paytirish, bo‘lish).
   - Mantiqiy operatorlar (and, or, not).
   - Taqqoslash operatorlari (kattaroq, kichikroq, tenglik va boshqalar).
   - Tayinlash operatorlari (qiymatni o‘zgaruvchiga tayinlash).
5. **if-else nazorat tuzilmasi nima?**
   Javob: Shartni tekshirish va shartga qarab turli xil kodlarni bajarish uchun ishlatiladigan tuzilma.
6. **Python’da funksiya nima va u qanday aniqlanadi?**
   Javob: Funksiya — bu takrorlanuvchi kodni qayta ishlatishga imkon beruvchi blok. Funksiya nomlanadi va parametrlar bilan chaqiriladi.

O‘rta daraja (Intermediate)

1. **Ro‘yxat (list) va o‘chmas ro‘yxat (tuple) o‘rtasidagi farq nima?**
   Javob:
   - Ro‘yxat (list): O‘zgartiriladigan ma’lumotlar turi.
   - O‘chmas ro‘yxat (tuple): O‘zgartirilmaydigan ma’lumotlar turi.
2. **Python’da dict va set o‘rtasidagi farq nima?**
   Javob:
   - dict: Kalit-qiymat juftliklaridan iborat. Kalitlar noyob bo‘lishi kerak.
   - set: Tartiblanmagan noyob elementlarning to‘plami.
3. **Python’da iterator va iterable nima?**
   Javob:
   - **Iterable:** Ichidagi elementlarni birma-bir olishga imkon beruvchi obyekt.
   - **Iterator:** next() funksiyasi orqali iterable obyektning keyingi elementini olish imkonini beruvchi obyekt.
4. **Fayl bilan ishlashda qanday asosiy tushunchalar mavjud?**
   Javob: Fayllarni ochish, o‘qish, yozish va yopish jarayonlari mavjud. Fayl rejimlari:

- r: Faqat o‘qish uchun.
- w: Faqat yozish uchun (agar fayl mavjud bo‘lsa, ustidan yozadi).
- a: Yozish uchun (eski ma’lumotlarni o‘chirib yubormaydi).

5. **Funksional dasturlash usullari qanday?**
   Javob: map, filter, va reduce funksiyalari funksional dasturlash usullariga kiradi. Bu usullar orqali ma’lumotlarni qayta ishlash jarayonlari amalga oshiriladi.

Ilg‘or daraja (Advanced)

1. **Python’da ob’ektga yo‘naltirilgan dasturlashning asosiy tushunchalari qanday?**
   Javob:
   - **Klass**: Ma’lumot va metodlarni saqlovchi shablon.
   - **Obyekt**: Klass asosida yaratilgan haqiqiy instansiya.
   - **Meros olish**: Bir klassning boshqa klassdan xususiyatlarni meros qilib olishi.
   - **Inkapsulyatsiya**: Ma’lumotlarni yashirish va faqat zarur funksiyalar orqali ularga murojaat qilish.
   - **Polimorfizm**: Bir xil metod nomini turli xil klasslarda turli xil vazifalarni bajarish uchun ishlatish.
2. **staticmethod va classmethod o‘rtasidagi farq nima?**
   Javob:
   - staticmethod: Klass yoki obyektga bog‘liq bo‘lmagan metod.
   - classmethod: Klassga bog‘liq metod bo‘lib, u klassni birinchi parametr sifatida qabul qiladi.
3. **Dunder metodlar nima va ular qanday ishlatiladi?**
   Javob:
   Dunder metodlar (double underscore methods) klass xatti-harakatlarini aniqlashga yordam beradi:
   - __init__: Klassni boshlang‘ichlash.
   - __str__: Obyektni foydalanuvchi uchun qulay shaklda ko‘rsatish.
   - __repr__: Obyektni texnik ko‘rinishda ifodalash.
4. **Python’da kontekst menejeri nima va qanday foydalaniladi?**
   Javob: Kontekst menejeri resurslarni boshqarish uchun ishlatiladi. U fayllarni ochib-yopish, ma’lumotlarni xavfsiz saqlashda yordam beradi.

+==========+

Mutable (O‘zgartiriladigan)

"Mutable" so‘zi ingliz tilidan tarjima qilganda "o‘zgaruvchan" degan ma’noni anglatadi. Bu ma’lumot turlarini dastur ishlashi davomida o‘zgartirish mumkin.

Asosiy xususiyatlari:

1. Mavjud obyektni o‘zgartirish mumkin.
2. Xotirada yangi obyekt yaratilmaydi; o‘sha obyektning o‘zi yangilanadi.
3. Bitta obyekt bilan bog‘liq bo‘lgan barcha o‘zgaruvchilar o‘zgarishlarni ko‘radi.

Misol ma’lumot turlari:

- **list** (ro‘yxat)
- **dict** (lug‘at)
- **set** (to‘plam)

Immutable (O‘zgartirilmaydigan)

"Immutable" so‘zi "o‘zgarmaydigan" degan ma’noni bildiradi. Bu turdagi ma’lumotlarni o‘zgartirishning iloji yo‘q; agar o‘zgartirishga harakat qilinsa, yangi obyekt yaratiladi.

Asosiy xususiyatlari:

1. Obyektni o‘zgartirishning iloji yo‘q.
2. Har bir o‘zgarish uchun yangi obyekt yaratiladi.
3. Ma’lumotlar xavfsizroq, chunki ular o‘zgarmaydi.

Misol ma’lumot turlari:

- **int** (butun son)
- **float** (kasrli son)
- **str** (qator)
- **tuple** (o‘chmas ro‘yxat)
- **frozenset** (muzlatilgan to‘plam)

Foydalanish holatlari:

- **Mutable:** Tez-tez o‘zgaradigan ma’lumotlarni saqlashda ishlatiladi. Masalan, foydalanuvchi tomonidan kiritilayotgan ma’lumotlar.
- **Immutable:** Barqaror ma’lumotlarni saqlash uchun ishlatiladi. Masalan, dastur ishlashi davomida o‘zgarmas kerak bo‘lgan konstantalar.

+==========+

1. **Statik tiplangan dasturlash tillari (Statically Typed Languages):**

Statik tiplash tizimida, o‘zgaruvchilarning turlari dastur kompilatsiya qilinishidan oldin belgilangan bo‘ladi. Ya'ni, o‘zgaruvchi turlari kod yozish vaqtida aniqlanadi va kompilator bu turlarni tekshiradi.

Xususiyatlari:

- O‘zgaruvchilarning turlari dastur yozilishi vaqtida belgilanadi.
- Dastur bajarilishini boshlashdan oldin, kompilator o‘zgaruvchi turlarini tekshiradi va xatoliklar aniqlanadi.

- Odatda, bu tillar ishlash tezligi yuqori bo‘ladi, chunki kompilator turlarning aniq ekanligini tekshiradi.
- Turlarni o‘zgartirish yoki xatoliklarni aniqlash dastur bajarilishidan oldin bo‘ladi.

Misollar:

- **C**: O‘zgaruvchilar aniq turlarga ega bo‘ladi (masalan, int, float, char).
- **C++**: Xuddi shunday, o‘zgaruvchilarni aniq belgilash talab qilinadi.
- **Java**: Har bir o‘zgaruvchining turi kompilyatsiya vaqtida belgilanadi (masalan, int, String).
- **Swift**: Turlar aniq belgilanadi va tekshiriladi.

2. **Dinamik tiplangan dasturlash tillari (Dynamically Typed Languages):**

Dinamik tiplash tizimida, o‘zgaruvchilarning turlari dastur bajarilishi vaqtida belgilanadi. Ya'ni, kod ishlash vaqtida o‘zgaruvchi turini aniqlash mumkin.

Xususiyatlari:

- O‘zgaruvchilarning turlari kod bajarilayotganda aniqlanadi.
- Turlarga oid xatoliklar dastur bajarilganda aniqlanadi.
- Dasturlashda tezlik jihatidan biroz sekin bo‘lishi mumkin, chunki kod bajarilishida turlarni aniqlash kerak.
- Dasturchi o‘zgaruvchilarni o‘zgartirishda tur haqida o‘ylash kerak emas, til o‘zi turlarni boshqaradi.

Misollar:

- **Python**: O‘zgaruvchi turi kod bajarilishida aniqlanadi. Masalan, x = 10 va keyin x = "Hello" bo‘lishi mumkin.
- **JavaScript**: O‘zgaruvchilarni turini aniq belgilash shart emas, let x = 10 yoki let x = "Hello" ishlashi mumkin.
- **Ruby**: Dinamik tiplangan til bo‘lib, o‘zgaruvchilar turlari bajarilish vaqtida belgilanadi.
- **PHP**: O‘zgaruvchilarning turlari bajarilish vaqtida belgilanadi, masalan, x = 10 va keyin x = "Hello" bo‘lishi mumkin.

Xulosa:

- **Statik tiplangan tillar**: Turlar dastur yozilishi vaqtida aniqlanadi, kompilator tekshiradi. Bu, turlarga oid xatoliklarni dastur bajarilishidan oldin topish imkonini beradi, lekin dastur yozishda ba'zi cheklovlarni keltirib chiqaradi.
- **Dinamik tiplangan tillar**: Turlar dastur bajarilishi vaqtida aniqlanadi, kodni tez yozish imkonini beradi, lekin xatoliklar faqat bajarilish vaqtida aniqlanadi.

Dasturlashda har ikki turdagi tillar o‘ziga xos afzalliklarga ega bo‘lib, dastur va loyiha talablariga qarab tanlanadi.

+==+

O'zgaruvchilar dasturda ma'lum bir qiymatni saqlash uchun ishlatiladigan nomlangan joylardir. Ular dasturda turli darajalarda ishlatilishi mumkin, ya'ni **local (mahalliy)** va **global (global)** o'zgaruvchilarni ajratish mumkin. Har bir turdagi o'zgaruvchilarning o'ziga xos ishlatilish holatlari va chegaralari mavjud.

1. **Local (Mahalliy) O'zgaruvchilar**

Local o'zgaruvchilar funksiyalar yoki metodlar ichida e'lon qilinadi va faqat shu funksiyaning ichida mavjud bo'ladi. Ular faqat o'zining mavjud bo'lgan blokida (masalan, funksiyada) ishlatiladi va tashqaridagi koddan foydalanib bo'lmaydi. Dastur ishlash vaqtida shu funksiya ishlatilganida mavjud bo'ladi va funksiya tugaganidan so'ng o'chadi.

Xususiyatlari:

- **Doimiy mavjudligi**: Faqat funksiyada yoki blokda mavjud bo'ladi.
- **Ko'rinishi**: Faqat o'zining e'lon qilingan doirasida ko'rinadi, tashqarida ishlatilishi mumkin emas.
- **Hayotiy davri**: Funksiya chaqirilganda yaratiladi va tugatilganda yo'qoladi.

Misol:

```
def my_function():
    x = 5  # 'x' - mahalliy o'zgaruvchi
    print(x)

my_function()  # x ni faqat bu yerda chaqirish mumkin
# print(x)  # Bu xatolik, chunki x faqat 'my_function' ichida mavjud
```

Bu yerda x faqat my_function funksiyasida mavjud. Agar xni funksiya tashqarisida chaqirsak, xatolik yuz beradi.

2. **Global (Global) O'zgaruvchilar**

Global o'zgaruvchilar dasturda har qanday joydan, ya'ni funksiyalar, metodlar va boshqa bloklardan foydalanish uchun e'lon qilinadi. Ular dastur bo'ylab mavjud va boshqa funksiyalar yoki bloklar tomonidan o'qilishi mumkin. Global o'zgaruvchilar dastur ishga tushgandan so'ng yaratiladi va dastur tugagunga qadar mavjud bo'ladi.

Xususiyatlari:

- **Doimiy mavjudligi**: Dastur tugaguncha mavjud bo'ladi.
- **Ko'rinishi**: Dasturdagi barcha kod bloklari, shu jumladan funksiyalar va metodlar tomonidan foydalanilishi mumkin.

- **Hayotiy davri**: Dastur ishga tushganda yaratiladi va tugashgacha mavjud bo'ladi.

Misol:

```
x = 10  # 'x' - global o'zgaruvchi

def my_function():
    print(x)  # 'x' ga global o'zgaruvchi sifatida murojaat qilinadi

my_function()  # 10
print(x)  # 10, global o'zgaruvchi
```

Bu yerda x global o'zgaruvchi sifatida butun dasturda mavjud. Funksiyaning ichida ham, tashqarisida ham ishlatilishi mumkin.

Global va Local O'zgaruvchilarni Farqlash

- **Local** o'zgaruvchilar faqat o'zining funksiya yoki blokida ishlatiladi. Ular tashqarida mavjud emas va xatolikni oldini olish uchun har bir funksiya o'zining xususiy o'zgaruvchilarini ishlatadi.
- **Global** o'zgaruvchilar dasturda istalgan joydan ishlatilishi mumkin, bu esa ular orqali umumiy holatni boshqarish imkonini beradi.

Global o'zgaruvchidan Mahalliy o'zgaruvchi ichida foydalanish

Global o'zgaruvchini mahalliy o'zgaruvchi ichida to‘g‘ridan-to‘g‘ri chaqirish mumkin, lekin agar siz mahalliy o‘zgaruvchini global o‘zgaruvchini o‘zgartirishni istasangiz, global kalit so‘zidan foydalanishingiz kerak.

Global o‘zgaruvchini o‘zgartirish:

```
x = 10  # global o'zgaruvchi

def my_function():
    global x
    x = 20  # global x o'zgartirildi
    print(x)

my_function()  # 20
print(x)  # 20, global x o'zgartirildi
```

Bu holatda, x global o‘zgaruvchisi my_function ichida global kalit so‘zi orqali o‘zgartirildi. Agar global kalit so‘zi ishlatilmasa, bu faqat mahalliy o‘zgaruvchi bo‘ladi va global o‘zgaruvchi o‘zgarmaydi.

+==========+

Funksiya (Function):

- **Funksiya** — bu ma'lum bir vazifani bajarish uchun yozilgan kodning bir bo‘lagi bo‘lib, uni turli joylarda chaqirib ishlatish mumkin.
- Funksiya ko‘pincha **global** (yoki dastur bo‘ylab) doiraga tegishli bo‘ladi, ya'ni u dasturdagi har qanday joydan chaqirilishi mumkin.
- Funksiyalar ba'zan parametrlar (kirish qiymatlari) qabul qiladi va **qaytarish** qiymatiga ega bo‘ladi (return).
- Funksiyalarning asosiy vazifasi — aniq bir vazifani bajarish va natija qaytarishdir.

Xususiyatlari:

- Funksiya umumiy vazifalarni bajarish uchun ishlatiladi.
- Har qanday kod blokidan, xususan, global doiradagi koddan chaqirilishi mumkin.
- Oddiy funksiyalar def kalit so‘zi bilan e'lon qilinadi.

Metod (Method):

- **Metod** — bu **ob'ektga yo‘naltirilgan dasturlash (OOP)** prinsiplarida sinf (class)ga tegishli bo‘lgan funksiyadir.
- Metodlar, ko‘pincha, **ob'ektning** xususiyatlari bilan ishlash uchun yoziladi va faqat o‘sha sinfning obyektiga tegishli bo‘ladi.
- Metodlar sinfda e'lon qilinadi va ular **ob'ektni (yoki sinfni)** chaqirgan holda ishlatiladi.
- **Metodlar** odatda ob'ektning holatini o‘zgartirish yoki uni qayta ishlash uchun ishlatiladi, ular **self** (yoki o‘zgaruvchi) orqali sinfning a'zolariga murojaat qilishi mumkin.

Xususiyatlari:

- Metodlar faqat sinfning obyektlari orqali chaqiriladi.
- Metodlar sinfning xususiyatlarini boshqaradi va sinfga tegishli bo‘ladi.
- Odatda, metodlar def kalit so‘zi bilan e'lon qilinadi, lekin ularning birinchi argumenti sifatida self mavjud bo‘ladi, bu sinf ob'ektiga murojaat qilishni anglatadi.

Farqlari:

1. **Tegishli joy**:
   - **Funksiya**: Funksiya dasturda global bo‘lishi mumkin, ya'ni uni istalgan joydan chaqirish mumkin.
   - **Metod**: Metodlar faqat sinf ob'ekti orqali chaqiriladi va sinfga tegishli bo‘ladi.
2. **Chaqirish usuli**:
   - **Funksiya**: Funksiya bevosita chaqiriladi, va u hech qanday ob'ektga tegishli bo‘lmasligi mumkin.

    - **Metod**: Metod ob'ektning yoki sinfning a'zosi sifatida chaqiriladi.
3. **Qaysi turdagi dasturlashda ishlatiladi**:
    - **Funksiya**: Funksiya har qanday dasturlash paradigmasida ishlatiladi, masalan, imperativ dasturlashda.
    - **Metod**: Metodlar asosan ob'ektga yo‘naltirilgan dasturlash (OOP)da ishlatiladi.
4. **O'zgaruvchilar**:
    - **Funksiya**: Funksiya o‘zining lokal o‘zgaruvchilariga ega bo‘lishi mumkin, lekin uning holatini o‘zgartirish uchun global o‘zgaruvchilarni ishlatish mumkin.
    - **Metod**: Metod sinf ob'ektining holatini va xususiyatlarini o‘zgartirishi mumkin, shuningdek, self orqali ob'ektning o‘zgaruvchilarini boshqaradi.

Xulosa:

- **Funksiya** — bu umumiy vazifalarni bajarish uchun ishlatiladigan va dasturda istalgan joydan chaqirilishi mumkin bo‘lgan kod blokidir.
- **Metod** — bu sinfga tegishli bo‘lgan funksiya bo‘lib, faqat ob'ektlar yoki sinflar orqali chaqiriladi. Odatda metodlar sinfning xususiyatlarini boshqaradi va o‘zgartiradi.

+==========+

1. **Matnli fayllar** (Text Files)

- Matnli fayllar oddiy matn ma'lumotlarini saqlash uchun ishlatiladi. Bu fayllarda yozilgan barcha ma'lumotlar insonga o‘qilishi oson bo‘ladi.
- **Kengaytmalari**: .txt, .csv, .log, .html, .json, .xml, va boshqalar.
- **Misollar**:
    - .txt — oddiy matnli fayl.
    - .csv — ma'lumotlar ajratilgan (comma-separated values) fayl, ko‘pincha jadval ma'lumotlarini saqlash uchun ishlatiladi.
    - .json — JavaScript Object Notation, ko‘pincha ma'lumot almashish va saqlash uchun ishlatiladi.
    - .html — veb-sahifa uchun fayl (Hypertext Markup Language).
    - .xml — extensible markup language, strukturaviy ma'lumotlarni saqlash uchun ishlatiladi.

2. **Binar fayllar** (Binary Files)

- Binar fayllar — bu matn bo‘lmagan fayllar, ularning ichidagi ma'lumotlar kompyuter tizimiga tegishli bo‘lib, to‘g‘ridan-to‘g‘ri insonga o‘qilishi qiyin.
- **Kengaytmalari**: .exe, .jpg, .png, .mp3, .pdf, .zip, .tar, va boshqalar.
- **Misollar**:
    - .exe — bajariladigan fayl (executable file), odatda dasturlarni ishga tushirish uchun ishlatiladi.
    - .jpg, .png, .gif — rasm (image) fayllari.
    - .mp3, .wav — musiqa va audio fayllari.
    - .pdf — Portable Document Format, hujjat fayli.
    - .zip, .tar — arxivlash fayllari.

3. **Ma'lumotlar bazasi fayllari** (Database Files)

- Ma'lumotlar bazasi fayllari maxsus formatda bo‘lib, ma'lumotlarni tashkil etilgan tarzda saqlash va o‘zgartirish imkonini beradi.
- **Kengaytmalari**: .db, .sqlite, .mdb, .accdb.
- **Misollar**:
  - .db — odatda SQLite ma'lumotlar bazasi fayllari.
  - .mdb, .accdb — Microsoft Access ma'lumotlar bazasining fayllari.

4. **Kompressiya fayllari** (Compressed Files)

- Kompressiya fayllari ma'lumotlarni siqib saqlash uchun ishlatiladi va ularning hajmini kamaytiradi.
- **Kengaytmalari**: .zip, .rar, .tar.gz, .7z.
- **Misollar**:
  - .zip, .rar — kompressiya formatlari.
  - .tar.gz, .tar.bz2 — Unix tizimlarida ko‘p ishlatiladigan arxivlash va kompressiya formatlari.

5. **Log fayllari** (Log Files)

- Log fayllari dasturlar va tizimlar tomonidan amalga oshirilgan faoliyatlar, xatoliklar va boshqa tizim holati haqida ma'lumotlarni saqlaydi.
- **Kengaytmalari**: .log.
- **Misollar**:
  - .log — tizimlar va ilovalar tomonidan yaratilgan jurnal fayllari, tizimdagi xatoliklar va faoliyatlarni yozib borish uchun ishlatiladi.

6. **Config fayllari** (Configuration Files)

- Config fayllari dasturlar yoki tizimlarning sozlamalari va parametrlarini saqlaydi. Bu fayllar ko‘pincha tizimni moslashtirish uchun ishlatiladi.
- **Kengaytmalari**: .ini, .yaml, .json, .xml.
- **Misollar**:
  - .ini — oddiy sozlama fayli, asosan Windows dasturlarida ishlatiladi.
  - .yaml — odatda konfiguratsiya ma'lumotlarini saqlash uchun ishlatiladi (YAML Ain't Markup Language).
  - .json — JSON formatida konfiguratsiya fayllari.

7. **Video va Audio fayllari** (Multimedia Files)

- Video va audio fayllari har xil media formatlarida ma'lumotlarni saqlaydi, masalan, videolar yoki musiqa.
- **Kengaytmalari**: .mp4, .avi, .mp3, .wav.
- **Misollar**:
  - .mp4 — video fayl formati.
  - .mp3, .wav — audio fayllari.

8. **Veb fayllari** (Web Files)

- Veb fayllari internetda ishlatiladigan va ko‘rinadigan ma'lumotlarni saqlaydi.
- **Kengaytmalari**: .html, .css, .js.
- **Misollar**:
  - .html — veb-sahifa uchun fayl.
  - .css — veb-sahifa dizaynini belgilovchi fayl.
  - .js — JavaScript fayli, veb-sahifa interaktivligini qo‘llab-quvvatlaydi.

Xulosa:

- Fayllar turli maqsadlarda ishlatiladi va ular bir nechta formatda mavjud. Fayl turi uning qanday ishlatilishi va qanday ma'lumotni saqlashi bilan belgilangan. Ma'lumotlarni saqlash va ishlashda turli fayl turlari sizga kerakli natijaga erishishda yordam beradi.

+==========+

Python'da quyidagi **kalit so'zlar** (reserved keywords) mavjud. Bu so'zlar Python tilida maxsus ma'no va funksiyaga ega bo'lib, ular o'zgaruvchi nomlari yoki funksiyalar sifatida ishlatilmaydi:

1. False
2. None
3. True
4. and
5. as
6. assert
7. async
8. await
9. break
10. class
11. continue
12. def
13. del
14. elif
15. else
16. except
17. finally
18. for
19. from
20. global
21. if
22. import
23. in
24. is
25. lambda
26. nonlocal
27. not

28. or
29. pass
30. raise
31. return
32. try
33. while
34. with
35. yield

Bu kalit so'zlar Python dasturlash tilida o'ziga xos ma'noga ega bo'lib, kodda ular o'zgaruvchi yoki funksiyalar sifatida ishlatilmaydi.