Python xotirasini bo’lish va bo’llmaslik xususiyatga ega.pythonda C yoki C++ kabi tillardagi kabi oldindan xotiradan joy ajratish shart emas.

Pythonda xotirani ajratish uchun ikki strategiyadan foydalaniladi.

1.Reference counting

2.garbage collection.

**REFERENCE COUNTING**

Python va va boshqa turli dasturlash tillari obyektga necha marta havola qilganlligini kuzatish orqali xotirani avtomatik boshqarish uchun mos yozublarni hisoblash,xotirani boshqarish yondashuvidan foydalaniladi.Malumot sioni

Python tilidagi har bir obyektning xossasidir.

**GARBAGE COLLECTION**

Chiqindilarni yig’ish bu endi kirish imkoni bolmaagan yoki ilova tomonidan foydalanilmaydigan xotiarani avtomatik ravishda tiklash uchun dasturlash tillarida qo’llaniladigan xotuirani boshqarish usuli.Bu xotira oqishni oldini olishga xotiradan foydalanishni optimallashtirishga va xotirani samarali taqsimlashga yordam beradi.

**GARBAGE COLLECTION ni foydali tomonlari**

Avtomatlashtirilgan xotira boshqaruvi: Xotiraning oqishi oldini olish va xotira tugashi ehtimolini kamaytirish uchun Python axlat yig'uvchisi havola qilinmagan ob'ektlarni avtomatik ravishda olib tashlaydi.

Xotirani boshqarish osonlashdi: axlat yig'uvchi dasturchilarni xotirani qo'lda boshqarish zaruratidan ozod qiladi, shunda ular diqqatlarini kod yaratishga qaratishlari mumkin, bu esa Python-ni dasturchilar uchun yuqori darajadagi va amaliyroq tilga aylantiradi.

Xotirani samarali tozalash: Chiqindilarni yig'ish qurilmasi avlodlar yig'ish orqali qisqa muddatli ob'ektlarni tezda aniqlash va yig'ish paytida ishlash effektlarini minimallashtirish uchun mo'ljallangan.

Moslashtirilgan sozlamalar: Axlat yig'uvchi o'z sozlamalarini moslashtirish imkoniyatlarini taqdim etadi, masalan, turli avlodlar uchun chegaralarni sozlash, ishlab chiquvchilarga o'zlarining maxsus dastur talablari asosida axlat yig'ish jarayonini nozik sozlash imkonini beradi.

**GARBAGE COLLECTION ni kamchiliklari**

Ishlash samaradorligiga ta'siri: Garchi axlat yig'uvchi ishlatilmagan xotirani samarali tozalash uchun mo'ljallangan bo'lsa-da, ayniqsa ko'p sonli ob'ektlar bilan ishlashda protsessor sarfi va bajarish vaqti biroz yuqori bo'lishi mumkin.

Xotirani boshqarishning qiyinligi: Python axlat yig'uvchisi xotirani boshqarishni osonlashtirsa-da, undan muvaffaqiyatli foydalanish ob'ektning ishlash muddati, ob'ektga havolalar va axlat yig'ish algoritmlari kabi tushunchalarni bilishni talab qilishi mumkin.

Xotirani boshqarish ustidan cheklangan nazorat: axlat yig'uvchining avtonom tabiati ishlab chiquvchilarga xotirani tozalashning aniq vaqti va xatti-harakatlarini kam nazorat qiladi, bu esa xotirani boshqarish ustidan nozik nazorat zarur bo'lgan ko'plab dastur stsenariylari uchun ideal bo'lmasligi mumkin.

Xatolik potentsiali: Garchi axlat yig'uvchi ishonchli va samarali bo'lishi uchun mo'ljallangan bo'lsa-da, u xatolarga yoki atipik xatti-harakatlarga to'sqinlik qilmaydi, bu xotiraning oqishi yoki ob'ektni noto'g'ri tozalashga olib kelishi mumkin.

**DYNAMIC TYPING**

Python dinamik tarzda yoziladi, ya'ni o'zgaruvchilar turlari kompilyatsiya paytida emas, balki ish vaqtida aniqlanadi va tekshiriladi. Python kabi dinamik tarzda terilgan tillarda oʻzgaruvchi turini ishlatishdan oldin uni aniq eʼlon qilish shart emas.

**Mutable and immutable**

Python-da ma'lumotlar turlari o'zgaruvchan va o'zgarmas ma'lumotlar turlariga bo'linadi. O'zgaruvchan ma'lumotlar turi - bu qiymatlari o'zgartirilishi mumkin bo'lganlar, o'zgarmas ma'lumotlar turi esa qiymatlarni o'zgartirib bo'lmaydigan ma'lumotlar turi hisoblanadi.

O’zgarmas obyektlarga int,float,bool,string,tuple,Unicode kabi datatypelarni kiritishmiz mumkin.

O’zgaruvchan obyektlarga list dictionary set kabilar misol bo’la oladi.

Ozgarmas obyektlarga kirish tezroq lekin o’zgartirish qiyin.

Ozgaruvchan obyektni ozgartirish oson.

Biroq, o'zgarmaslikda ham istisno mavjud. Biz bilamizki, Python-dagi kortej o'zgarmasdir. Lekin kortej ob'ektlarga o'zgarmas bog'lanishlarga ega bo'lgan nomlar ketma-ketligidan iborat.

**IS OPERATORI**

python identifikatsiya operatorlari (is, is not) ob'ektlarni identifikatoriga qarab solishtirish uchun ishlatiladi. Agar operatorning har ikki tomonidagi o'zgaruvchilar aynan bir xil ob'ektga ishora qilsa, "is" operatorining baholashi to'g'ri bo'ladi. Aks holda, bu bizga noto'g'ri baho beradi.

**IS VA == NING FARQI**

Python identifikatsiya operatori (is) va tenglik operatori (==) o'rtasida nozik farq bor. Raqamlarni solishtirish uchun Python is operatoridan foydalanganda kodingiz yaxshi ishlashi mumkin, ammo u to'satdan yo'q. Python operatori == operatoridan tezroq ekanligini qaerdadir eshitgan bo'lishingiz mumkin yoki u ko'proq Pythonik ko'rinishini his qilishingiz mumkin. Biroq, shuni yodda tutish kerakki, ushbu operatorlar o'zlarini bir xil tutmaydilar.

**NONE**

None kalit so'zi null o'zgaruvchini yoki ob'ektni aniqlash uchun ishlatiladi. Pythonda None kalit so'zi ob'ekt bo'lib, u NoneType sinfining ma'lumotlar turidir. Biz hech qanday o'zgaruvchiga None ni belgilashimiz mumkin, lekin siz boshqa NoneType obyektlarini yarata olmaysiz.