|  |
| --- |
| Programming |
| Azizbek Xabibullayev |

**Programming**

**O‘qituvchi:** Islomboyev Botir

**Guruh identifikatori: 23-407**

**Talaba ID: 230757**

**Taqdim etilgan sana:** 17.01.2025 yil

**BTEC o'quvchilar topshiriqlarini baholash va deklaratsiya**

Baholash uchun ishlarni taqdim etganda, har bir o'quvchi ish o'ziniki ekanligini tasdiqlovchi deklaratsiyani imzolashi kerak.

|  |  |
| --- | --- |
| **O‘quvchi (talaba) identifikatori:** | 230757 |
| **Baholovchi nomi:** | Islomboyev Botir |
| **Birlik yoki komponent raqami va nomi:** | Programming |
| **Topshiriq nomi:** | Internal Assignment |
| **Topshiriq topshirilgan sana:** | 17.01.2025yil |

Iltimos, har bir topshiriq uchun berilgan ishlarni sanab o'ting. Ishlarni topish mumkin bo'lgan sahifa raqamlarini ko'rsating yoki ishlarning mohiyatini tavsiflang (masalan, grafik, rasm).

|  |  |  |
| --- | --- | --- |
| **Topshiriq vazifasi ma'lumoti** | **Ishlar taqdim etildi** | **Sahifa** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |
| --- |
| **O'quvchi deklaratsiyasi**  Men ushbu topshiriq uchun taqdim etilgan ish meniki ekanligini tasdiqlayman. Ishda foydalanilgan manbalarga aniq havola qildim. Men noto'g'ri deklaratsiya noto'g'ri ishlashning bir shakli ekanligini tushunaman.  **O‘quvchi imzosi:  Sana: 17** .01.2025 yil |

Mundarija

[Kirish 3](#_Toc188053302)

[A.P1: 4](#_Toc188053303)

[A.M1: 5](#_Toc188053304)

[**B.D1:** 6](#_Toc188053305)

[A.P2: 8](#_Toc188053306)

[**B.M2** 10](#_Toc188053307)

[**C.D2:** 13](#_Toc188053308)

[**B.P3:** 16](#_Toc188053309)

[**C.M3:** 21](#_Toc188053310)

[**D.D3:** 23](#_Toc188053311)

[B.P4 24](#_Toc188053312)

[C.P5: 27](#_Toc188053313)

[D.M4: 28](#_Toc188053314)

[D.D4: 30](#_Toc188053315)

[Xulosa 31](#_Toc188053316)

[adabiyotlar ro'yxati 32](#_Toc188053317)

# Kirish

Bugungi tezkor chakana savdo muhitida mijozlarni qoniqtirishni ta'minlash va operatsiyalarni optimallashtirish uchun samaradorlik, aniqlik va ishonchlilik muhim ahamiyatga ega. Ushbu loyiha Toshkentda joylashgan, 5000 nafar mijoz bazasiga ega bo‘lgan xayoliy NewEra Cash & Carry oziq-ovqat do‘koni uchun avtomatlashtirilgan buyurtma boshqaruvi tizimini yaratishga qaratilgan. Ayni paytda do‘kon kundalik operatsiyalarni qo‘lda boshqarish tizimidan foydalanadi, bu esa ma'lumotlarning noaniqligi, samarasizligi va takrorlanishiga olib keladi.

Loyiha ushbu muammolarni hal qilish maqsadida Python va Django asosida kompyuterlashtirilgan ma'lumotlar bazasini boshqarish tizimini yaratishni maqsad qilgan. Ushbu yechim mahsulot inventarizatsiyasi, buyurtmalarni qayta ishlash va mijozlarni boshqarish kabi vazifalar uchun ishonchli ma'lumotlar boshqaruvini ta'minlaydi. Django platformasining tez rivojlanish va kengaytirilish imkoniyatlaridan foydalanib, do‘konning kundalik jarayonlarini samarali avtomatlashtirish uchun qulay interfeysga ega bo‘lgan tizim ishlab chiqiladi.

Tizimning asosiy funksiyalari quyidagilardan iborat:

1. Administrator uchun imkoniyatlar: Mahsulot ma'lumotlarini qo'shish, tahrirlash va o'chirish imkoniyati.
2. Mijozlar uchun interfeys: Mahsulot kataloglarini ko'rib chiqish va buyurtma berish qulayligi.
3. Ma'lumotlarning yaxlitligi: Ma'lumotlar bazasini aniq, izchil va real vaqt rejimida yangilanishini ta'minlash.
4. Ushbu loyiha protsessual va obyektga yo‘naltirilgan dasturlashning afzalliklarini namoyish etadi, shuningdek, murakkab dasturlash jarayonlarini boshqarishda integratsiyalashgan dasturlash muhiti (IDE)ning qo‘llanilishini ko‘rsatadi. Tizim zamonaviy kodlash standartlari va xatolarni boshqarish usullarini qo‘llash orqali xavfsiz, ishonchli va texnik xizmat ko‘rsatishga qulay bo‘lgan yechimni ta'minlaydi.

# A.P1:

**Algoritmning ta'rifi**  
Algoritm – bu muammoni hal qilish yoki ma'lum bir vazifani bajarish uchun bosqichma-bosqich ko'rsatmalar to'plamidir. Algoritmlar aniq va mantiqiy bo'lishi kerak, shuningdek, kiruvchi ma'lumotlarni qayta ishlash orqali chiqaruvchi natijaga erishishni ta'minlaydi. Dasturiy ta'minotda algoritmlar turli vazifalarni bajarish, masalan, ma'lumotlarni saralash, qidirish yoki optimallashtirish uchun ishlatiladi.

**Ilova yaratish jarayoni**  
Django yordamida ilova yaratish jarayoni quyidagi bosqichlardan iborat:

1. **Talablarni aniqlash**
   * Muammoni o‘rganish va asosiy funksionallarni aniqlash.
   * Foydalanuvchilar ehtiyojlarini aniqlash va ular uchun qulay interfeys yaratishni rejalashtirish.
2. **Loyihani boshlash**
   * Django loyiha shablonini yaratish uchun terminalda django-admin startproject project\_name buyrug'idan foydalaniladi.
   * Loyiha ichida kerakli dasturlarni yaratish uchun python manage.py startapp app\_name buyrug'i bajariladi.
3. **Ma'lumotlar bazasini sozlash**
   * settings.py faylida ma'lumotlar bazasi parametrlarini o'rnatish (masalan, SQLite, PostgreSQL).
   * ORM (Object-Relational Mapping) vositasi yordamida jadvallarni aniqlash va migratsiyalarni bajarish.
4. **URL marshrutlash va ko'rinishlarni yaratish**
   * Har bir URL uchun marshrutni aniqlash.
   * Foydalanuvchilar tomonidan kiritilgan so'rovlarni qayta ishlash uchun mos ko'rinish (view) funksiyalarini yozish.
5. **Shablon va statik resurslarni yaratish**
   * HTML, CSS, va JavaScript yordamida foydalanuvchi interfeysini ishlab chiqish.
   * Django shablon tizimi yordamida dinamik sahifalarni yaratish.
6. **Sinov va debugging**
   * Django'ning ichki sinov vositalari yordamida ilovadagi xatolarni topish va tuzatish.
   * Birlik testlari va avtomatlashtirilgan sinovlar yordamida kod sifati tekshiriladi.
7. **Ishga tushirish**
   * Loyihani joylashtirish uchun server konfiguratsiyasi (masalan, Apache yoki Nginx).
   * Django’ning manage.py runserver buyrug‘idan foydalanib, lokal serverda ilovani ishlatish va sinash.

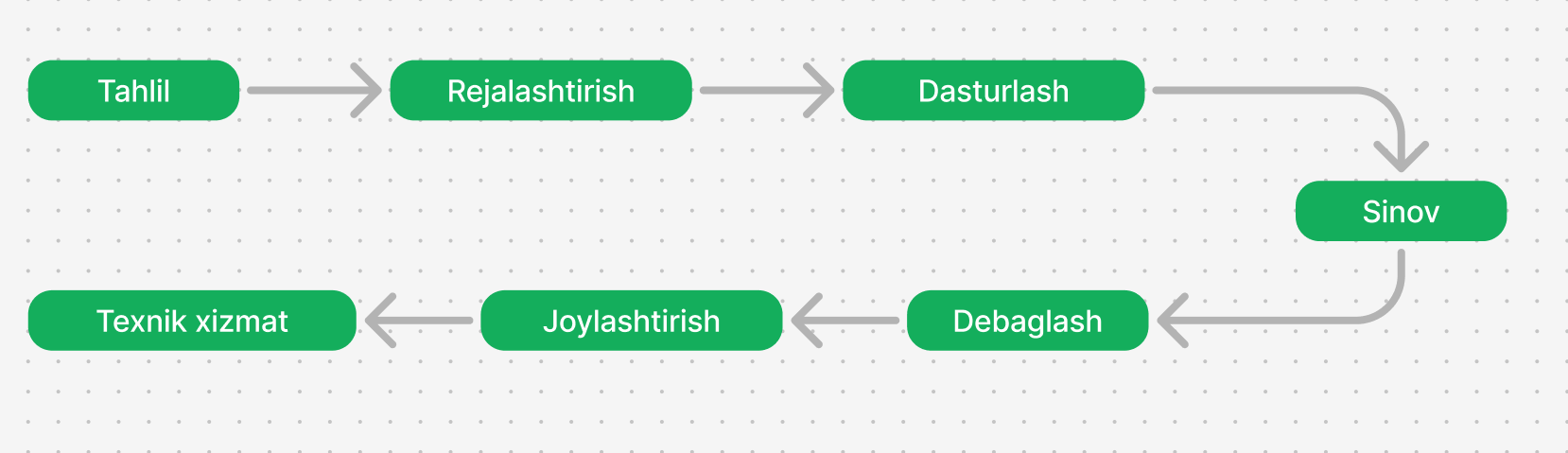
Django yordamida algoritmlarni samarali amalga oshirish mumkin, chunki bu platforma kuchli vositalar, masalan, ORM, keshlash, va xavfsizlik funksiyalari bilan ta'minlaydi.

# A.M1:

Dastur ishlab chiqish jarayoni bir nechta bosqichdan iborat bo'lib, har biri dastur yaratishda muhim ahamiyatga ega. Quyida Django bilan ishlash jarayonini bosqichma-bosqich bayon qilaman:

1. **Tahlil qilish (Analysis)**
   * Talablarni aniqlash: Foydalanuvchilarning muammolari va ehtiyojlarini aniqlash uchun dastlabki tahlil o'tkaziladi.
   * Tizim uchun asosiy funksionallar va imkoniyatlar rejalashtiriladi.
2. **Rejalashtirish (Design)**
   * Arxitektura loyihasi: Django asosida modellar, ko'rinishlar (views), va marshrutlar (URLs) dizayni amalga oshiriladi.
   * Ma'lumotlar bazasi dizayni: ORM yordamida jadvallarni rejalashtirish.
   * Interfeys dizayni: Foydalanuvchilar uchun qulay va samarali interfeysni shakllantirish.
3. **Dasturlash (Implementation)**
   * **Kod yozish**: Django bilan loyiha va dasturlar yaratiladi (startproject va startapp buyrug‘lari ishlatiladi).
   * **Backend funksiyalari**: Ko'rinishlar (views) yordamida foydalanuvchi so'rovlarini qayta ishlash.
   * **Frontend interfeysi**: Django shablonlari va statik resurslar (CSS, JavaScript) yordamida foydalanuvchi interfeysini ishlab chiqish.
4. **Sinov (Testing)**
   * Sinov holatlarini ishlab chiqish: Birlik testlari va integratsiya testlarini yozish (masalan, unittest yoki pytest yordamida).
   * Django sinov vositalari yordamida ilovaning ishlashi va xatolarini tekshirish.
5. **Debaglash (Debugging)**
   * Django’ning xatoliklarni aniqlash vositalaridan foydalanish.
   * Xatolarni topib tuzatish uchun loglarni tahlil qilish.
6. **Joylashtirish (Deployment)**
   * Loyihani ishlab chiqarish muhiti uchun tayyorlash.
   * Server konfiguratsiyasi: Apache yoki Nginx kabi veb-serverlarni sozlash.
   * Docker yoki boshqa konteynerizatsiya vositalaridan foydalanib, dasturiy ta'minotni joylashtirishni osonlashtirish.
7. **Texnik xizmat ko‘rsatish (Maintenance)**
   * Foydalanuvchilardan kelgan fikr-mulohazalar asosida dasturiy ta'minotni yangilash va optimallashtirish.
   * Xavfsizlikni kuchaytirish va yangi funksiyalar qo'shish.

**Diagrammalar**  
Quyida dastur ishlab chiqish jarayonini grafik ravishda tasvirlash uchun blok-sxema keltirilgan



**Django bilan jarayonni boshqarishning afzalliklari:**

* Modulli tuzilma va qayta foydalanish imkoniyati.
* ORM orqali ma'lumotlar bazasi bilan ishlashning qulayligi.
* Tez prototip yaratish imkoniyati.
* Keng doiradagi xavfsizlik funksiyalari.

# **B.D1:**

Algoritm dasturiy ta'minot loyihalarining asosi bo‘lib, dastur qanday vazifalarni qanday ketma-ketlikda bajarishi kerakligini aniqlab beradi. Dasturlash jarayonida algoritmlar yozma ko‘rinishda (matematik formulalar yoki blok-sxemalar shaklida) belgilanadi va ular dasturlash tilida amalga oshiriladi. Ushbu bosqichlar o‘rtasidagi munosabatni tahlil qilish muhimdir. Quyida misol bilan tushuntiriladi:

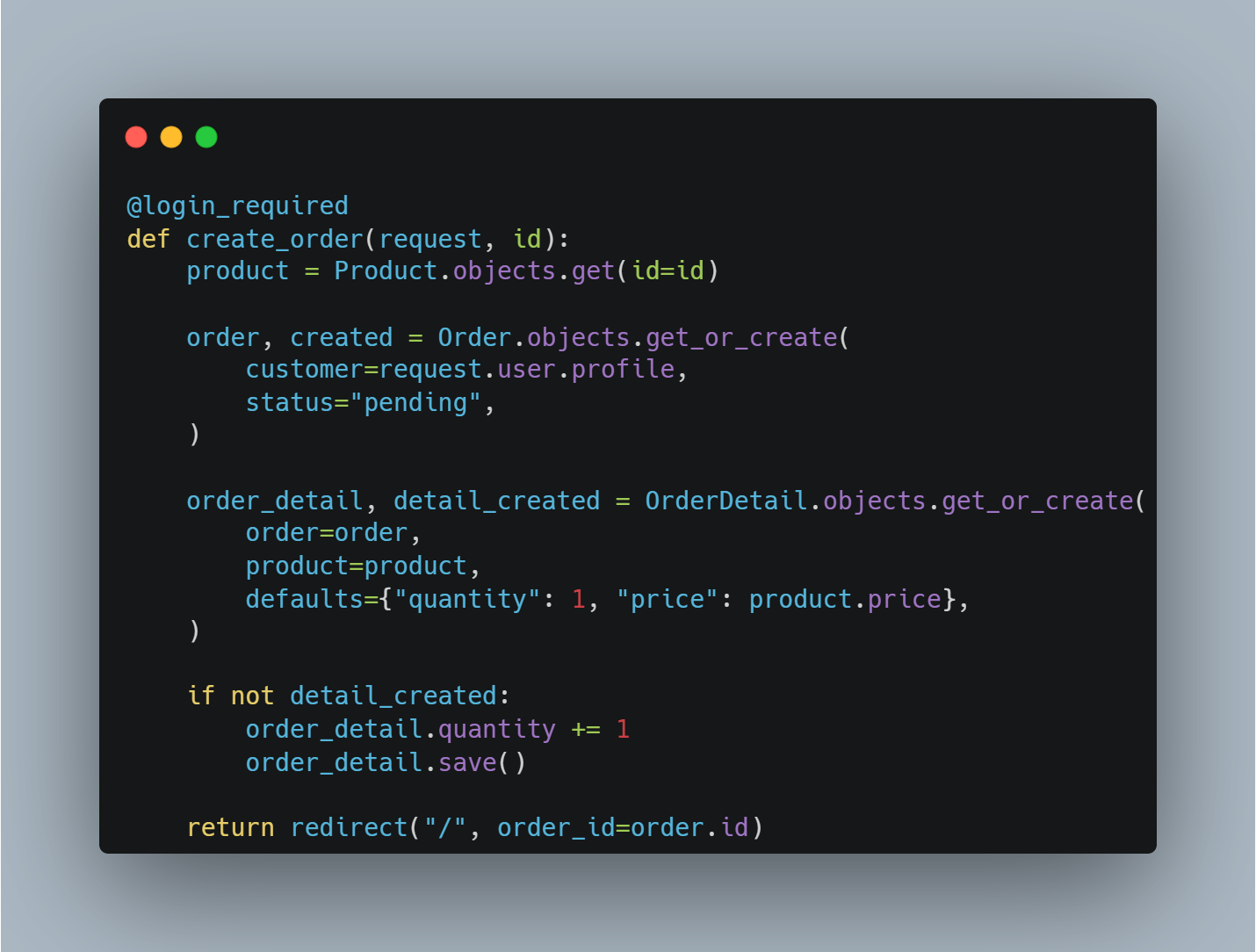
**1. Algoritmning yozma ko‘rinishi (matematik yoki blok-sxema)**

Misol: **Buyurtma jarayonini boshqarish** algoritmi quyidagicha bosqichlarni o‘z ichiga oladi:

1. Foydalanuvchining tizimga kirishini tekshirish (autentifikatsiya).
2. Foydalanuvchining mahsulotlarni ko‘rish va buyurtma yaratish imkoniyatini ta'minlash.
3. Buyurtma yaratilgandan so‘ng, uni ma'lumotlar bazasiga saqlash.
4. Administrator tomonidan buyurtma holatini boshqarish imkoniyatini ta'minlash.

**2. Kod varianti (Django yordamida)**

Yuqoridagi algoritmni Django asosida amalga oshirish uchun quyidagi kod varianti yoziladi:



3. Algoritm va kod o‘rtasidagi munosabat

* Yozma algoritm umumiy ketma-ketlik va vazifalar tartibini belgilaydi, ammo dastur tilining sintaksisi va texnik jihatlari ko‘rsatilmagan.
* Kod varianti esa yozma algoritmning texnik jihatdan amalga oshirilgan shaklidir. Misol uchun, foydalanuvchining buyurtma yaratish jarayoni yozma algoritmdagi barcha bosqichlarni qamrab oladi: ma'lumotlarni tekshirish, buyurtma yaratish, va uni ma'lumotlar bazasiga saqlash.
* Moslashuvchanlik: Yozma algoritmning afzalligi – u ko‘p tillarda amalga oshirilishi mumkin, kod esa biror ma’lum dasturlash tiliga moslashtiriladi (masalan, Python/Django).

Algoritm dastur loyihasining reja bosqichini tashkil etadi va kod ushbu rejani amalga oshirishning vositasidir. Yozma algoritmlar dasturchilarga vazifalarni tizimli ko‘rinishda ko‘rsatib beradi va kod yozishda xatoliklarning oldini olishga yordam beradi. Django asosidagi kod bu algoritmning to‘liq va samarali texnik echimidir.

# A.P2:

Dasturlash paradigmalari – bu dasturiy ta'minotni ishlab chiqishda foydalaniladigan usullar va yondashuvlar to'plami. Ular turli vazifalarni hal qilish uchun ma’lum bir mantiq va strukturalarni taqdim etadi. Quyida ushbu paradigmalarning mohiyati va asosiy xususiyatlari keltirilgan:

**1. Jarayonga asoslangan dasturlash (Procedural Programming)**

Jarayonga asoslangan dasturlash – bu muammoni bir nechta ketma-ket bajariladigan bosqichlarga bo‘lib, har bir bosqichni alohida funksiya yoki protsedura sifatida ifodalash usuli.

**Asosiy xususiyatlari:**

* Kod mantiqiy bloklarga bo‘linadi (funksiyalar/protseduralar).
* Qayta foydalaniladigan kodlar uchun mos.
* Kiruvchi va chiquvchi ma'lumotlarga asoslangan hisoblashni amalga oshiradi.
* Global va lokal o'zgaruvchilar ishlatiladi.

**Afzalliklari:**

* Oddiy va tushunarli tuzilma.
* Boshlang'ich darajadagi dasturlash uchun qulay.

**Kamchiliklari:**

* Murakkab loyihalarda kodni boshqarish qiyinlashadi.
* Ma'lumotlarni himoya qilish va xavfsizlik kam ta'minlangan.

**2. Obyektga yo‘naltirilgan dasturlash (Object-Oriented Programming – OOP)**

Obyektga yo‘naltirilgan dasturlash – bu muammoni real hayotdagi obyektlarga o‘xshash ravishda modellash usuli. Obyektlar xususiyatlar (atributlar) va harakatlardan (metodlar) tashkil topadi.

**Asosiy xususiyatlari:**

* **Kapsulyatsiya:** Ma'lumotlar va funksiyalarni bir obyekt ichida birlashtirish.
* **Merosxo‘rlik:** Bitta klassning xususiyatlarini boshqa klassga o‘tkazish.
* **Polimorfizm:** Bir xil funksiya nomini turli klasslarda turlicha ishlatish.
* **Abstraksiya:** Murakkabliklarni yashirish va faqat muhim ma'lumotlarni taqdim etish.

**Afzalliklari:**

* Kodni qayta foydalanish imkoniyati.
* Loyihalarni tuzilishli boshqarish.
* Katta va murakkab dasturlarni ishlab chiqishga mos.

**Kamchiliklari:**

* O‘rganish va amalga oshirish biroz murakkab.
* Ko'proq resurs talab qiladi.

**3. Hodisaga asoslangan dasturlash (Event-Driven Programming – EDP)**

Hodisaga asoslangan dasturlash – bu foydalanuvchi harakatlariga (masalan, tugma bosilishi yoki klaviaturadan ma’lumot kiritilishi) yoki dasturiy voqealarga javob beradigan kod yozish usuli.

**Asosiy xususiyatlari:**

* Hodisalar va ularning ishlov beruvchilari o'rtasidagi bog'liqlikni ta'minlaydi.
* Qo'shimcha modul va kutubxonalar orqali hodisalarni boshqarish.
* Asosan GUI dasturlar va real vaqtdagi tizimlarda qo'llaniladi.

**Afzalliklari:**

* Foydalanuvchi interfeysli dasturlar uchun mos.
* Moslashuvchan va dinamik.

**Kamchiliklari:**

* Hodisalar o‘rtasidagi murakkab bog‘lanishlar tufayli xatoliklarni aniqlash qiyinlashadi.
* Dastur ishlash tezligiga salbiy ta’sir ko‘rsatishi mumkin.

**Paradigmalarning o‘zaro aloqadorligi**

Bu paradigmalar bir-birini to‘ldirishi yoki birgalikda ishlatilishi mumkin. Masalan:

* Django dasturlashda OOP asosiy paradigma bo‘lsa-da, unda hodisalarga asoslangan elementlar (masalan, signal va callbacklar) ham qo‘llaniladi.
* Django view funksiyalari jarayonga asoslangan paradigmadan foydalanadi.

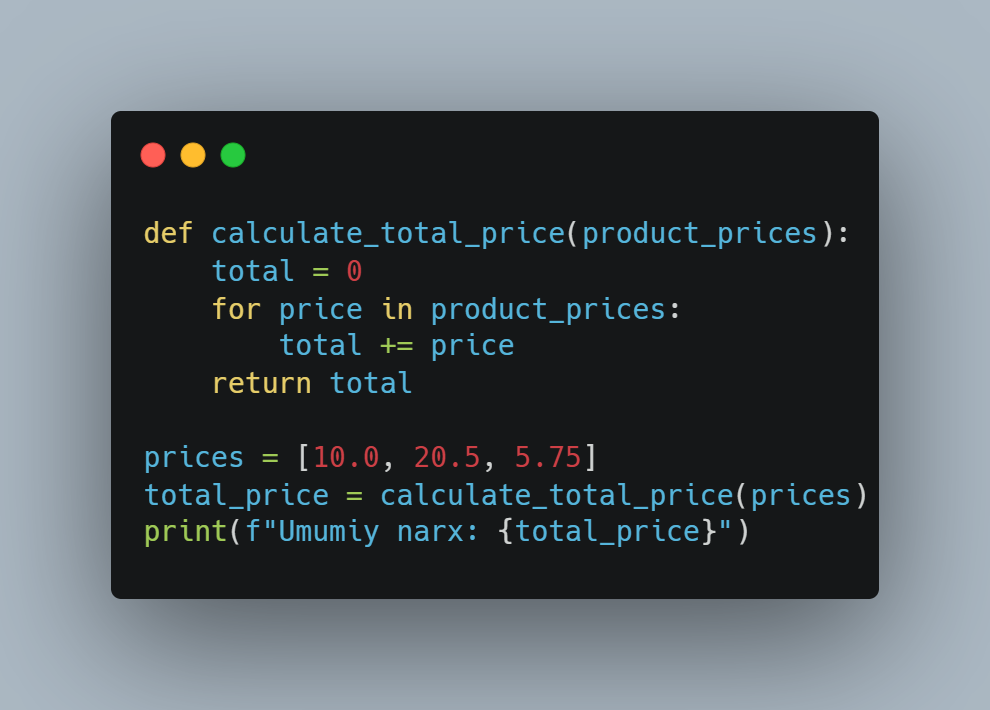
# **B.M2**

Dasturlashda uchta asosiy paradigma mavjud: protsedurali, obyektga yo‘naltirilgan (OOP), va hodisaga asoslangan dasturlash (Event-Driven Programming). Quyida ularning mohiyati, Django frameworkida qo‘llanilishi va farqlari tahlil qilinadi.

**1. Protsedurali dasturlash**

Protsedurali dasturlash asosiy vazifalarni ketma-ket bajaruvchi tartiblangan ko‘rsatmalar to‘plamidir. U dasturiy ta'minotni funksiyalar va protseduralarga bo‘lib tashlash orqali tashkil qiladi. Django’da protsedurali yondashuv ba’zi joylarda ishlatiladi, ayniqsa mustaqil funksiyalarni yozishda.

**Djangodagi protsedurali yondashuv misoli:**



 **Afzalliklari:** Soddalik va to‘g‘ridan-to‘g‘ri vazifa bajarish.

 **Kamchiliklari:** Katta va murakkab loyihalarda kodni boshqarish qiyinlashadi.

**Obyektga yo‘naltirilgan dasturlash (OOP)**

Obyektga yo‘naltirilgan dasturlash ma'lumotlarni sinflar va obyektlar sifatida tashkil etadi. Django OOP paradigmasidan keng foydalanadi, chunki har bir model, view, va boshqa komponentlar obyekt sifatida qaraladi.

**Djangodagi OOP misoli:**

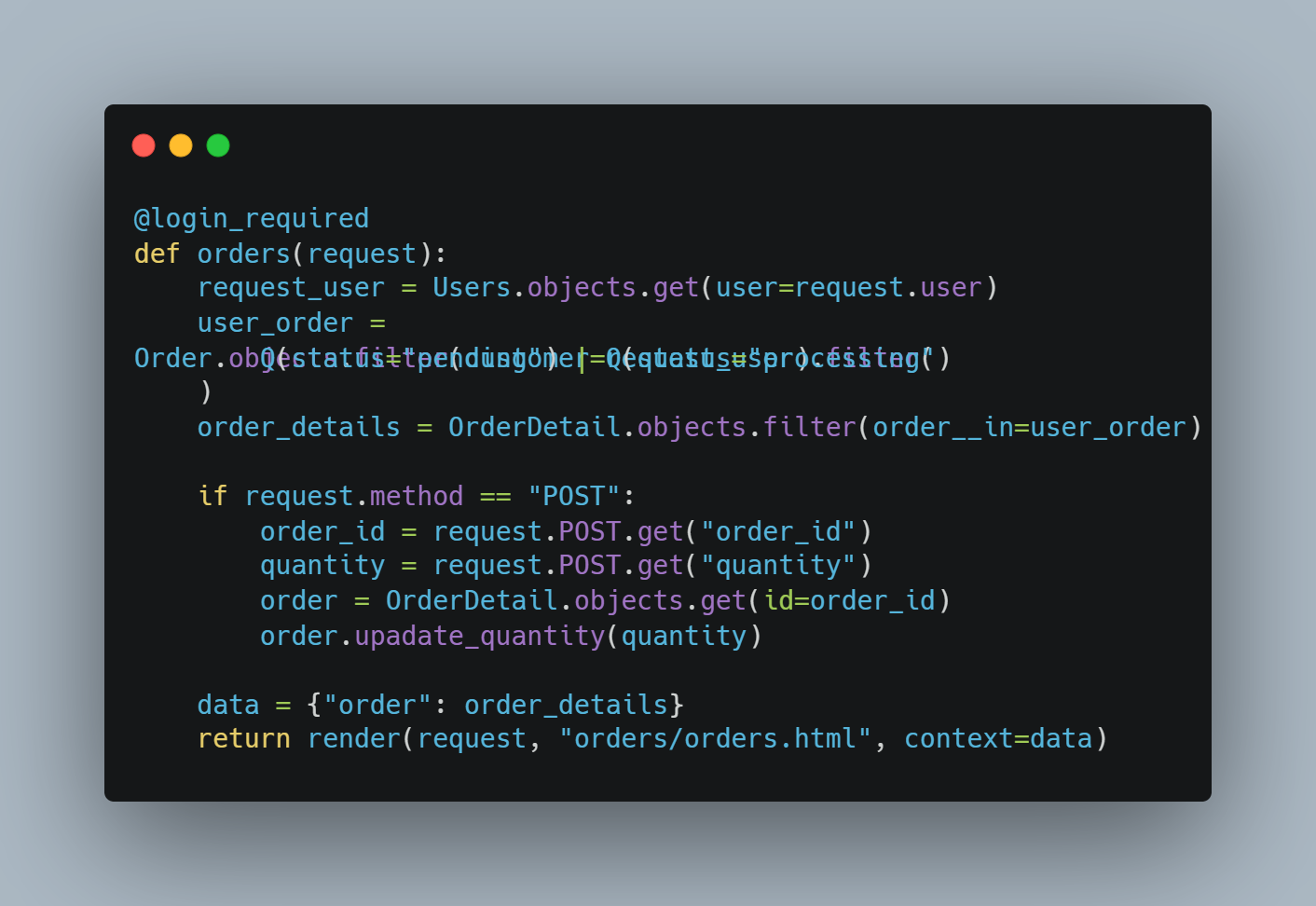


* **Afzalliklari:** Katta tizimlarda kodni modullashtirish va qayta ishlatish imkonini beradi.
* **Kamchiliklari:** OOP o‘rganish va ishlatishda ko‘proq vaqt talab etadi.

Hodisaga asoslangan dasturlash (Event-Driven Programming)

Hodisaga asoslangan dasturlash, dastur voqealarga (masalan, foydalanuvchi amallari yoki tizim voqealari) javob berishga asoslangan. Django’da bu yondashuv asosan signal yoki hook mexanizmlari orqali amalga oshiriladi.

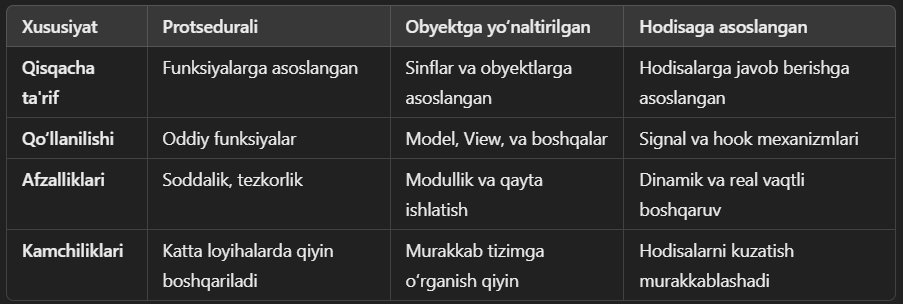
Djangodagi hodisaga asoslangan misol:



 **Afzalliklari:** Asinxron va dinamik hodisalarni boshqarish imkoniyati.

 **Kamchiliklari:** Hodisalar soni ko‘paygan sari dastur murakkablashishi mumkin.

Django’da paradigmalarni taqqoslash



Django barcha uchta paradigmadan foydalanishga imkon beradi, bu esa uni juda moslashuvchan frameworkga aylantiradi. Kichik vazifalar uchun protsedurali yondashuv mos bo‘lsa, katta va murakkab tizimlarda obyektga yo‘naltirilgan va hodisaga asoslangan yondashuvlar afzalroqdir. Bu paradigmalarning birgalikda qo‘llanilishi loyiha muvaffaqiyati uchun muhim rol o‘ynaydi.

# **C.D2:**

Dasturlash paradigmalarining har biri o‘ziga xos yondashuv va afzalliklarga ega. Protsedurali, obyektga yo‘naltirilgan (OOP), va hodisaga asoslangan dasturlash paradigmalari dasturlashning asosiy metodologiyalari bo‘lib, ular bir-biridan sezilarli darajada farq qiladi. Quyida ularning tuzilishi, afzalliklari va kamchiliklari tanqidiy baholanadi.

**1. Protsedurali dasturlash (Procedural Programming)**

**Tuzilishi:**  
Protsedurali dasturlash vazifalarni turli funksiyalarga ajratish orqali amalga oshiriladi. Kodlar yuqoridan pastga qarab bajariladi, va har bir funksiya aniq bir vazifani bajarishga mo‘ljallangan. Django’da bu yondashuv kichikroq va mustaqil funksiyalarni yozishda qo‘llaniladi.

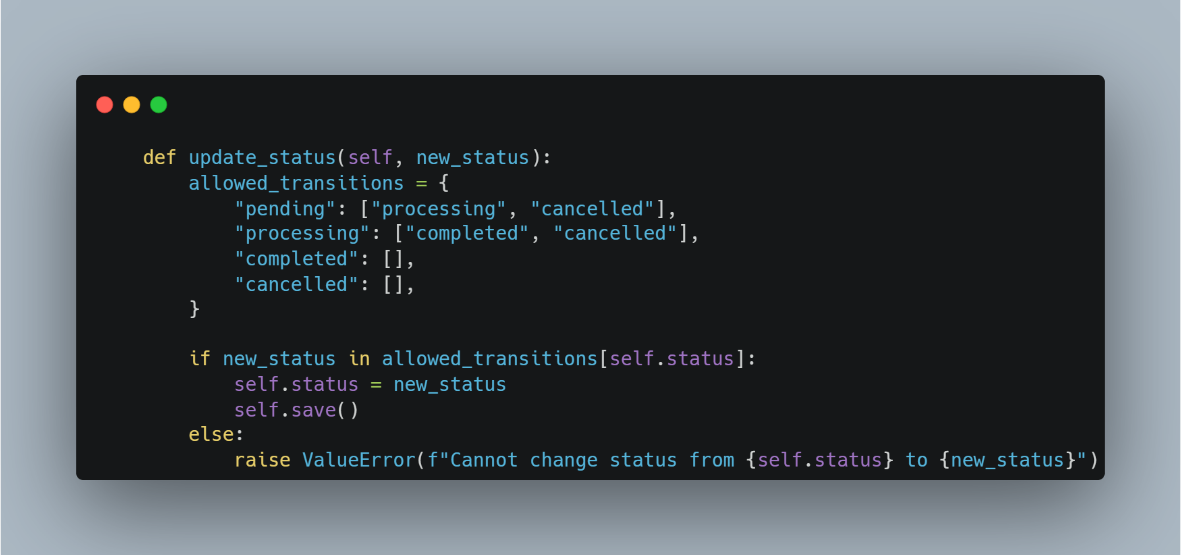
**Afzalliklari:**

* Oson tushuniladi va yangi dasturchilar uchun mos.
* Oddiy va tez bajariladigan vazifalar uchun juda samarali.
* Kodni oddiy shaklda modullashtirish imkonini beradi.

**Kamchiliklari:**

* Katta loyihalarda kodning murakkabligini boshqarish qiyinlashadi.
* Ma’lumotlar va funksiyalarni ajratish darajasi past, bu esa kodni qayta ishlatishda qiyinchilik tug‘diradi.

**Misol:**



**2. Obyektga yo‘naltirilgan dasturlash (Object-Oriented Programming - OOP)**

**Tuzilishi:**  
OOP ma'lumotlar va funksiyalarni sinflar va obyektlar orqali tashkil etadi. Django’da model va viewlarni yaratishda OOP yondashuvi keng qo‘llaniladi.

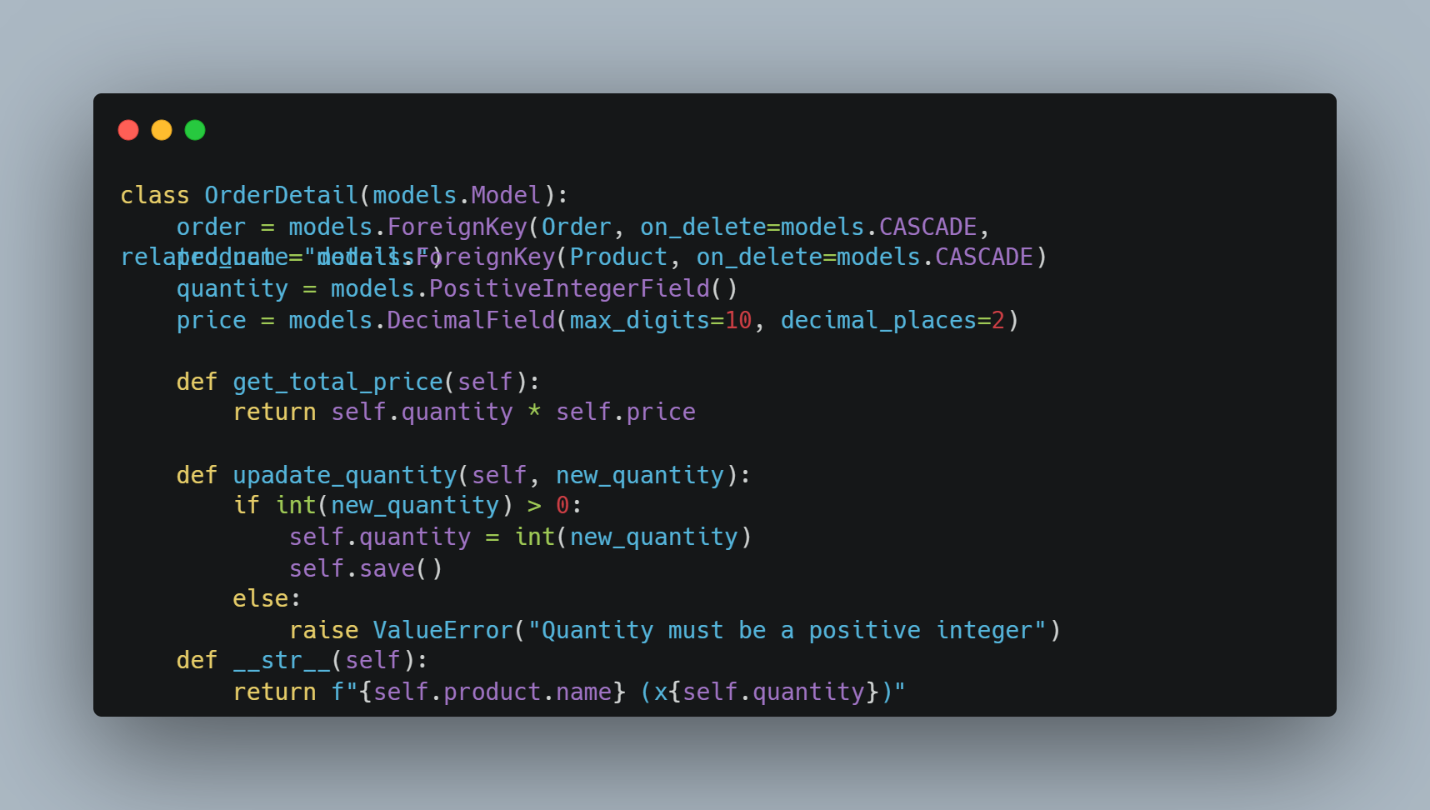
**Afzalliklari:**

* Katta va murakkab tizimlarni modullashtirishga yordam beradi.
* Kodni qayta ishlatish imkonini beradi (meros olish).
* Ma'lumotlarni yashirish va inkapsulyatsiya kabi xavfsizlik funksiyalarini ta'minlaydi.

**Kamchiliklari:**

* Kichik loyihalar uchun ortiqcha murakkablik tug‘diradi.
* Yangi boshlovchilar uchun o‘rganish va ishlatish nisbatan qiyin.

**Misol:**



**3. Hodisaga asoslangan dasturlash (Event-Driven Programming)**

**Tuzilishi:**  
Hodisaga asoslangan dasturlash foydalanuvchi yoki tizim hodisalariga javob berishga asoslangan. Django’da signal mexanizmi ushbu paradigmadan foydalanish uchun ishlatiladi.

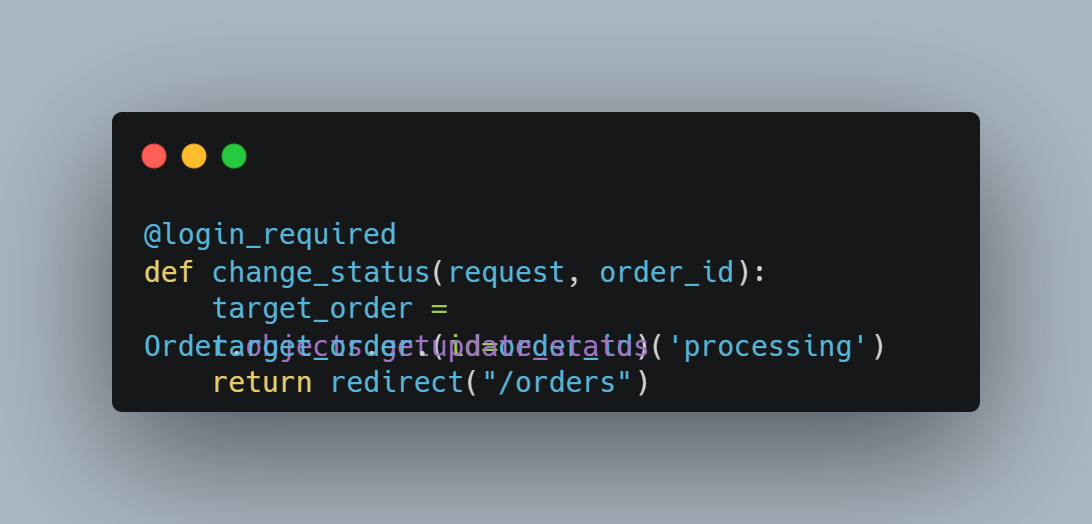
**Afzalliklari:**

* Dinamik va real vaqtli vazifalarni samarali boshqaradi.
* Asinxron hodisalar uchun juda mos.
* Ko‘plab parallel vazifalarni bir vaqtda boshqarish imkoniyatini beradi.

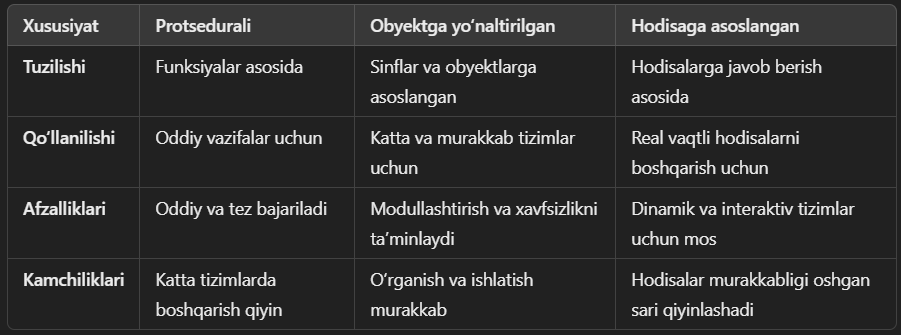
**Kamchiliklari:**

* Murakkab tizimlarda hodisalarni boshqarish qiyinlashishi mumkin.
* Hodisalar qanchalik murakkab bo‘lsa, dastur shunchalik qiyin kuzatiladi va xatoliklar kuzatiladi.

**Misol:**



Paradigmalarning taqqoslanishi



Har bir paradigma o‘ziga xos afzallik va kamchiliklarga ega. Django’da ushbu paradigmalar birgalikda ishlatiladi:

* Protsedurali dasturlash oddiy vazifalarni bajarishda qo‘llaniladi.
* OOP murakkab tizimlarni modullashtirish uchun asosiy paradigma sifatida xizmat qiladi.
* Hodisaga asoslangan yondashuv esa real vaqtli o‘zgarishlarni boshqaradi.

Django bu paradigmalarning har birini samarali birlashtirish imkonini beruvchi kuchli framework hisoblanadi, bu esa uni har xil loyihalar uchun moslashtirilgan qilib ko‘rsatadi.

# **B.P3:**

Django frameworkidan foydalanib, **Chakana Savdo Buyurtma Tizimi** uchun asosiy algoritmlarni amalga oshiruvchi dastur yoziladi. Ushbu dastur mahsulotlarni qo‘shish, buyurtmalarni yaratish va ularni boshqarishni o‘z ichiga oladi. Bu jarayonni samarali amalga oshirish uchun **PyCharm** yoki **VS Code** kabi integratsiyalashgan dasturlash muhiti (IDE)dan foydalaniladi. IDE yozilgan kodni tekshirish, xatolarni aniqlash va debugging jarayonlarini osonlashtiradi. Men bu loyihani tayyorlash uchun Vs Code dan foydalandim sababi bu satur PyCharm dan ko’ra yengilroq mening kompyuterim uchun juda qulay.

**IDEdan foydalanishning afzalliklari**

* **Kod yozish:** Kodni avtomatik to‘ldirish va syntax highlighting imkoniyati.
* **Xatolarni aniqlash:** IDE sizga koddagi xatolarni darhol ko‘rsatadi.
* **Debugging:** Kodni real vaqt rejimida tekshirish va nosozliklarni bartaraf etish imkonini beradi.

**Loyihaning asosiy maqsadi va funksionalligi**

Ushbu loyiha quyidagi algoritmlarni bajaradigan dastur yozishni talab qiladi:

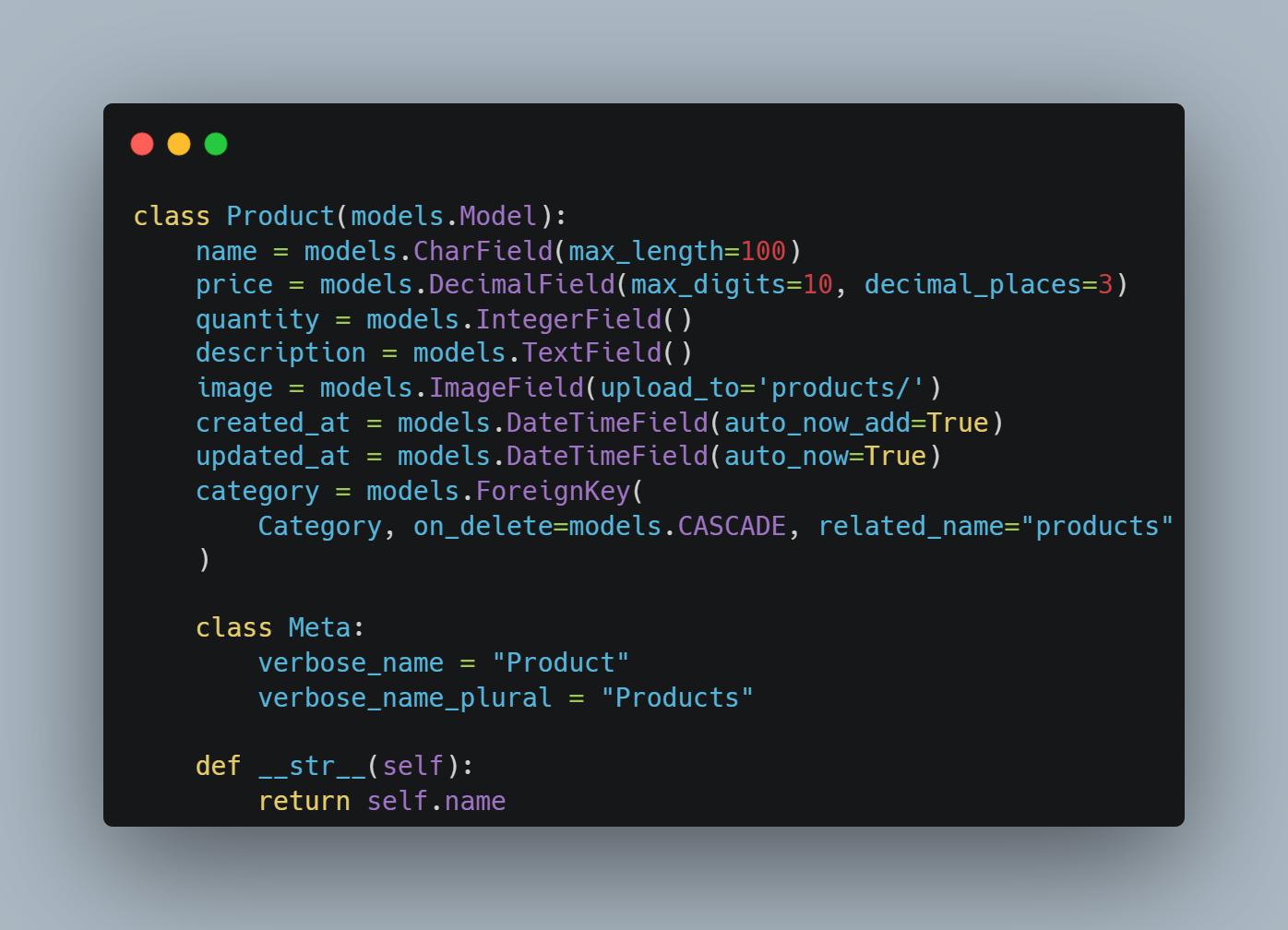
1. Mahsulotlarni CRUD amallari yordamida boshqarish (Yaratish, O‘qish, Yangilash, O‘chirish).
2. Mijozlarning buyurtma yaratish jarayonini boshqarish.
3. Buyurtmalar haqida batafsil ma’lumot olish imkoniyati.

Men Django lohinai tuzish uchun alohida environment yaratdim bu keyinchalik versiya bilan bog’liq bo’lishi mumkin bo’lgan xatoliklarni oldini oladi. Loyixani asosini 3ga bo’ldim ular:

1. Products
   * Masulotlarni asosiy oynada chiqarish



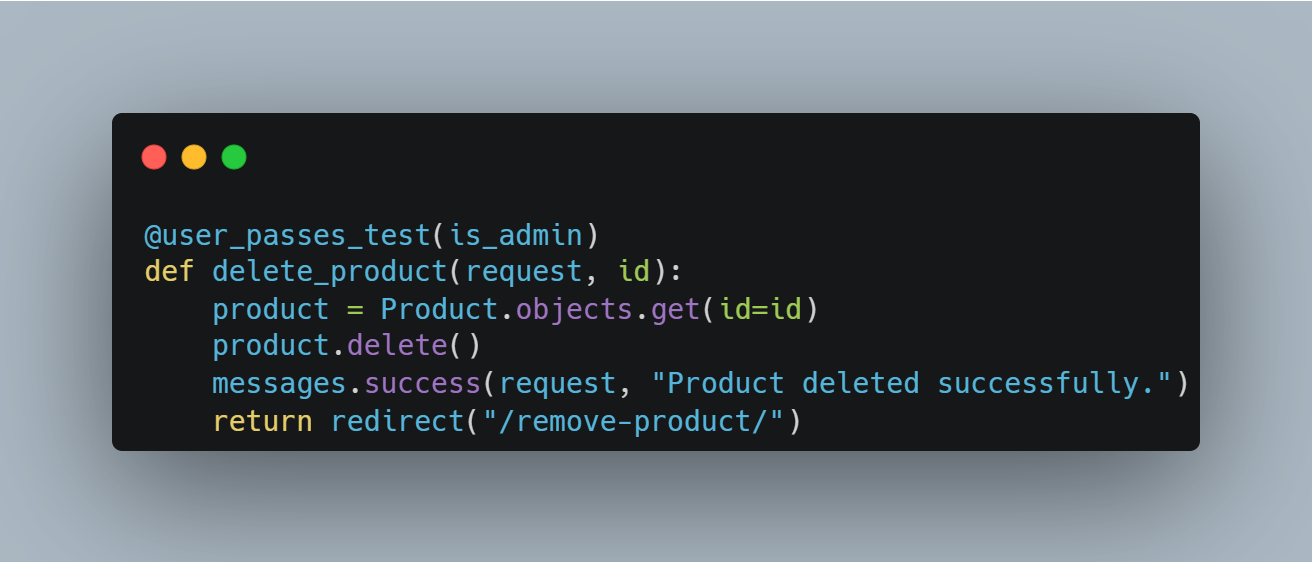
* + Maxulot haqida to’liq malumot saqlash



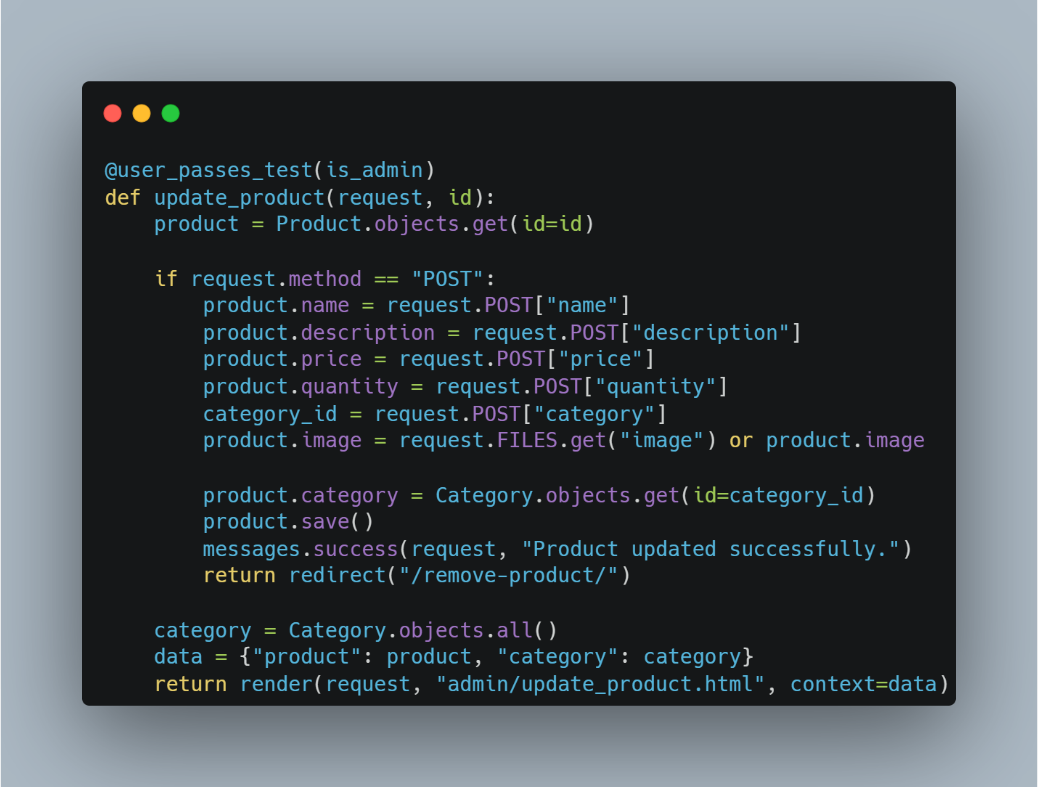
* + Yangi maxsulot qo’shish



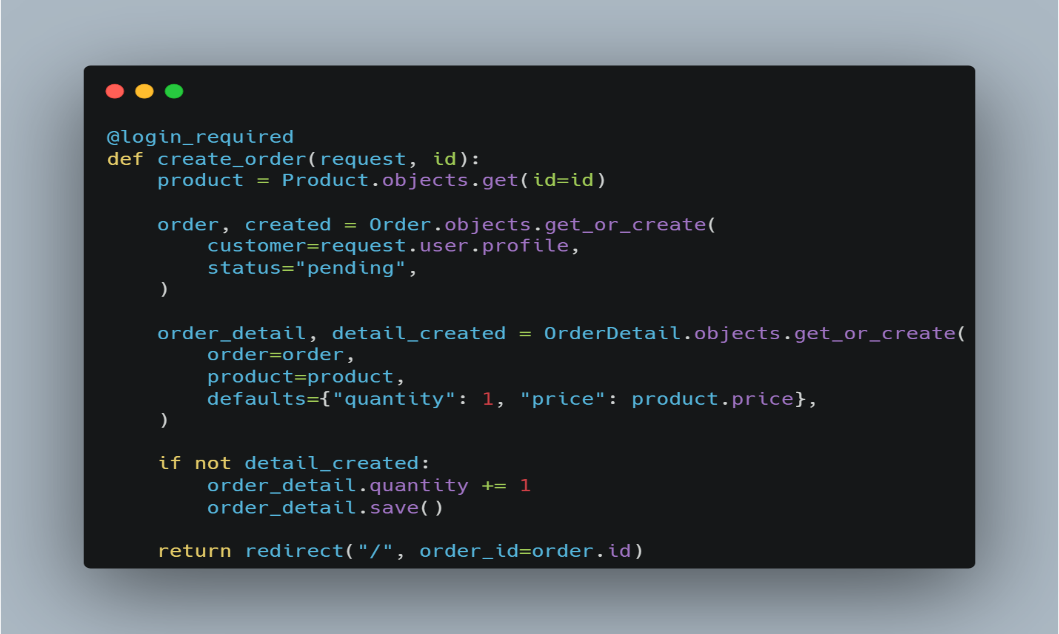
* + O’chirish



* + Yangilash



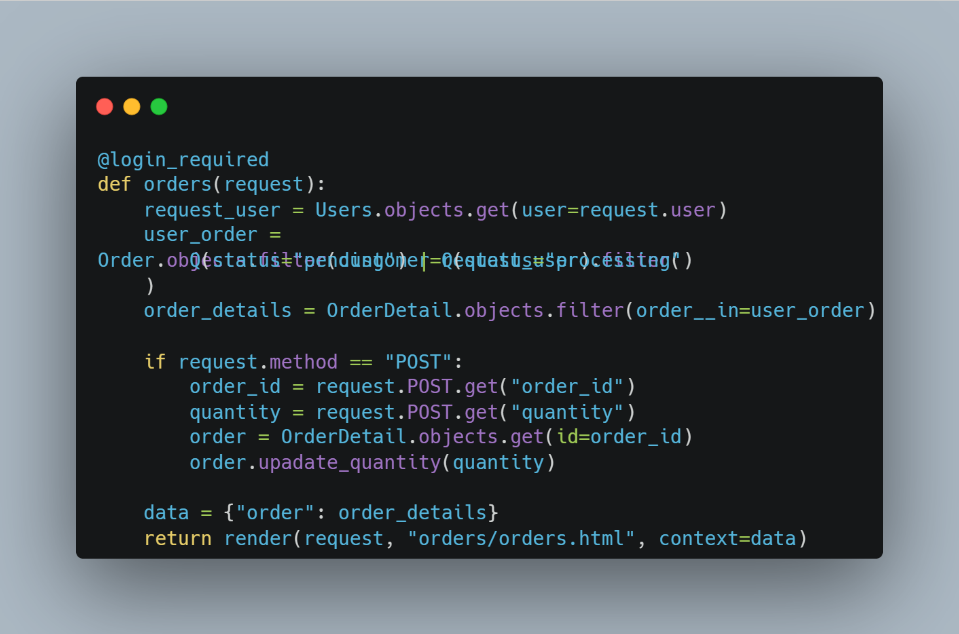
1. Orders
   * Foydalanuvchidan buyurtmalarni qabul qilish



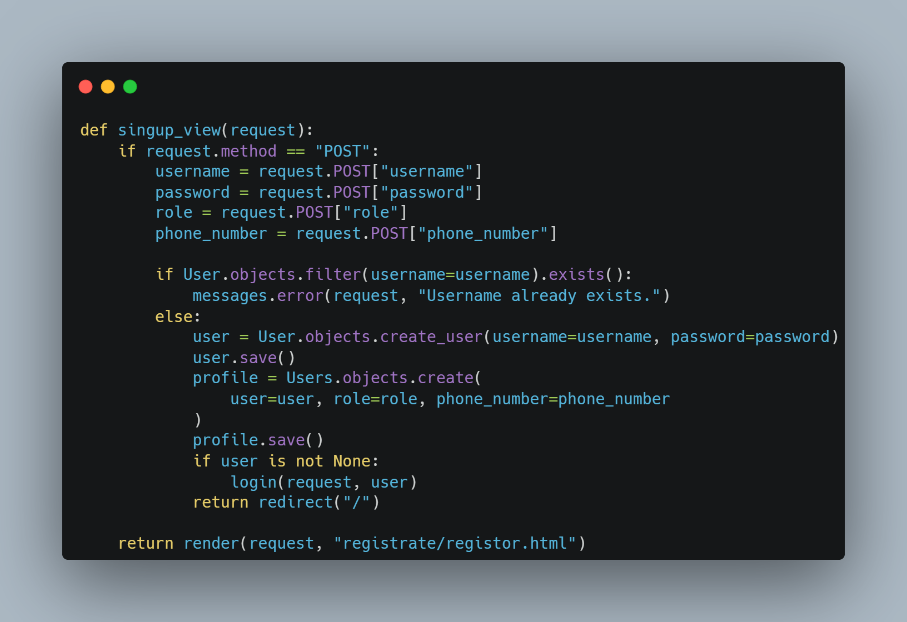
* + Buyurtma syatusini o’zgartirish



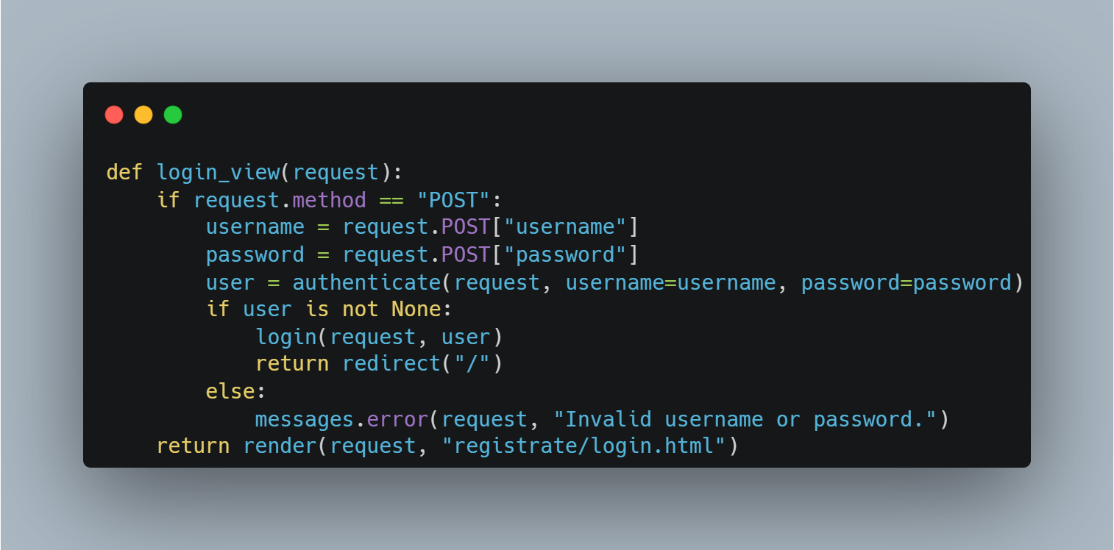
* + Barcha oldin qilingan buyurtmalarni chiqarish



1. Users
   * Yangi foydalanuvchini ro’yxatga olish



* + Tizimdan chiqib ketganlarini qayta tizimga qo’shish



* + Tizimdan chiqarish



Django va IDE yordamida asosiy algoritmlarni amalga oshirish jarayoni kodni modullashtirishni, xatolarni tezkor aniqlashni va samarali dastur yaratishni ta’minlaydi. Ushbu yondashuv loyihani muvaffaqiyatli yakunlash uchun muhim rol o‘ynaydi.

# **C.M3:**

Django loyihalarini ishlab chiqish jarayonida **VS Code** samarali vosita hisoblanadi. U kodni yozish, debugging qilish, testlash va ma’lumotlar bazasi bilan ishlashni qulay va tezkor qiladi. Quyida VS Code yordamida ishlab chiqish jarayonini boshqarishning asosiy jihatlari yoritiladi.

**Loyiha tuzilmasini boshqarish**

VS Code’ning qulay interfeysi loyihaning barcha papkalari va fayllarini tartibda saqlashga imkon beradi. Masalan, Django loyihasida quyidagi tuzilma ishlatiladi:

* **myproject/** – asosiy loyiha papkasi.
* **orders/** – ilova papkasi.
* **db.sqlite3** – ma’lumotlar bazasi fayli.

Kodni samarali yozish

VS Code Python kengaytmasi yordamida kodni avtomatik to‘ldirishni, sintaksis xatolarini tekshirishni va PEP8 standartlariga muvofiq formatlashni ta’minlaydi.

**Debugging va nosozliklarni bartaraf etish**

VS Code debugging vositasi koddagi xatolarni tezkor aniqlash imkonini beradi. Debug rejimida dastur ishlashini to‘xtatib, o‘zgaruvchilar qiymatini kuzatish mumkin.

**Ma’lumotlar bazasi boshqaruvi**

Ma’lumotlar bazasi bilan ishlash uchun terminal orqali quyidagi buyruqlardan foydalaniladi:

1. Migratsiyalarni yaratish:
   * python manage.py makemigrations
2. Migratsiyalarni qo‘llash:
   * python manage.py migrate
3. Admin panel uchun superfoydalanuvchi yaratish:
   * python manage.py createsuperuser

Testlash jarayoni

Django test sinflari yordamida kodni sinovdan o‘tkazish mumkin.



VS Code yordamida Django loyihasini boshqarish ishlab chiqish jarayonini sezilarli darajada osonlashtiradi. IntelliSense, debugging vositalari va terminal integratsiyasi kabi funksiyalar kodni samarali yozishni, xatolarni tezkor aniqlashni va loyihani sifatli rivojlantirishni ta’minlaydi. Django loyihalarini yaratishda VS Code’ning imkoniyatlaridan foydalanish ishlab chiqish jarayonini tezlashtiradi va sifatini oshiradi.

# **D.D3:**

Dasturiy ta’minot ishlab chiqishda IDE’dan foydalanish yoki foydalanmaslikning turli afzalliklari va cheklovlari mavjud. Quyida bu jihatlar tanqidiy baholangan.

**1. IDE yordamida ishlab chiqish jarayoni**

**IDE (masalan, VS Code)** dasturiy ta’minot yaratishda quyidagi imkoniyatlarni taqdim etadi:

1. **Kod yozish samaradorligi:**  
   IDE avtomatik to‘ldirish funksiyasi (IntelliSense) tufayli kod yozish tezligi oshadi. Masalan, Django’da models.Model sinfini yozishda tegishli maydonlarni tavsiya etadi.
2. **Debugging (nosozliklarni aniqlash):**  
   Debug vositasi yordamida kodni to‘xtash nuqtalarida (breakpoints) to‘xtatib, o‘zgaruvchilarni tekshirish mumkin.
3. **Sintaksisni tekshirish:**  
   IDE yozilgan koddagi sintaksis va mantiqiy xatolarni real vaqt rejimida ko‘rsatadi, bu xatolarni tuzatishni osonlashtiradi.
4. **Loyiha tuzilmasini boshqarish:**  
   Loyiha papkalarini tartibga solish va ko‘p fayllar bilan ishlash osonlashadi.

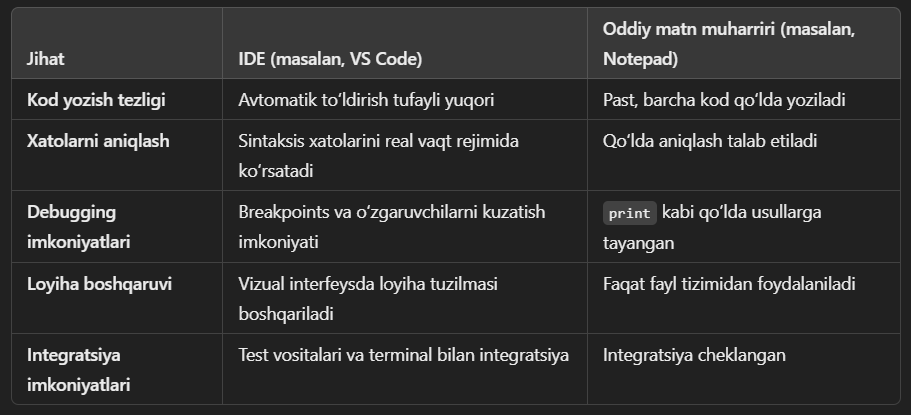
**Misol:**  
VS Code yordamida debugging jarayoni:



**IDE’dan foydalanmaslik (faqat oddiy matn muharriri bilan ishlash)**

**Oddiy matn muharriri (masalan, Notepad)** dan foydalanish quyidagi cheklovlarni keltirib chiqaradi:

1. **Kod yozish murakkabligi:**  
   Kodni avtomatik to‘ldirish yo‘qligi sababli, dasturchi barcha kodni qo‘lda yozadi, bu xatolik ehtimolini oshiradi.
2. **Xatolarni aniqlash qiyinligi:**  
   Sintaksis yoki mantiqiy xatolarni aniqlash qo‘lda bajariladi, bu esa vaqtni sezilarli darajada oladi.
3. **Debuggingning murakkabligi:**  
   Oddiy matn muharririda debugging uchun hech qanday vosita mavjud emas, bu esa dasturchiga xatolarni izlashda qiyinchilik tug‘diradi.



IDE yordamida ishlab chiqish jarayoni kodni samarali va tez yozishni, debuggingni osonlashtirishni va sintaksis xatolarini kamaytirishni ta’minlaydi. Oddiy matn muharririda ishlash esa dasturchidan ko‘proq vaqt va kuch talab qiladi. Django kabi murakkab tizimlarda IDE’dan foydalanish ayniqsa zarur, chunki u loyihani boshqarishni va ishlab chiqish jarayonini sezilarli darajada yaxshilaydi.

# B.P4

Debaglash (nosozliklarni tuzatish) dasturiy ta'minotni ishlab chiqishda muhim bosqichlardan biridir. Django frameworki nosozliklarni aniqlash va bartaraf etish uchun bir qancha qulay vositalarni taqdim etadi. Quyida debaglash jarayoni bosqichlari va Django’da mavjud vositalar bayoni keltirilgan:

1. **Django DEBUG Rejimi**
   * Django’da settings.py faylida DEBUG parametri mavjud. Bu parametrni True qiymatiga o‘rnatilganda, xatolik yuz berganda batafsil xatolik haqida ma'lumot ko‘rsatiladi. Bu rejim ilovani ishlab chiqish (development) davrida nosozliklarni tezroq aniqlashga yordam beradi.
   * Ammo bu rejimni ishlab chiqarish (production) muhiti uchun o‘chirish kerak, aks holda tizim xavfsizligiga putur yetishi mumkin.
2. **Django Nosozlik Xabarlarini Ko‘rish**

Agar kodda xatolik yuz bersa, Django avtomatik ravishda HTML sahifada batafsil xatolik hisobotini ko‘rsatadi. Bu hisobot quyidagilarni o‘z ichiga oladi:

* + Xatolik turi (Exception type)
  + Xatolik haqida ma'lumot (Exception message)
  + Xatolik qatori (Traceback)
  + So‘rov ma'lumotlari (Request data)

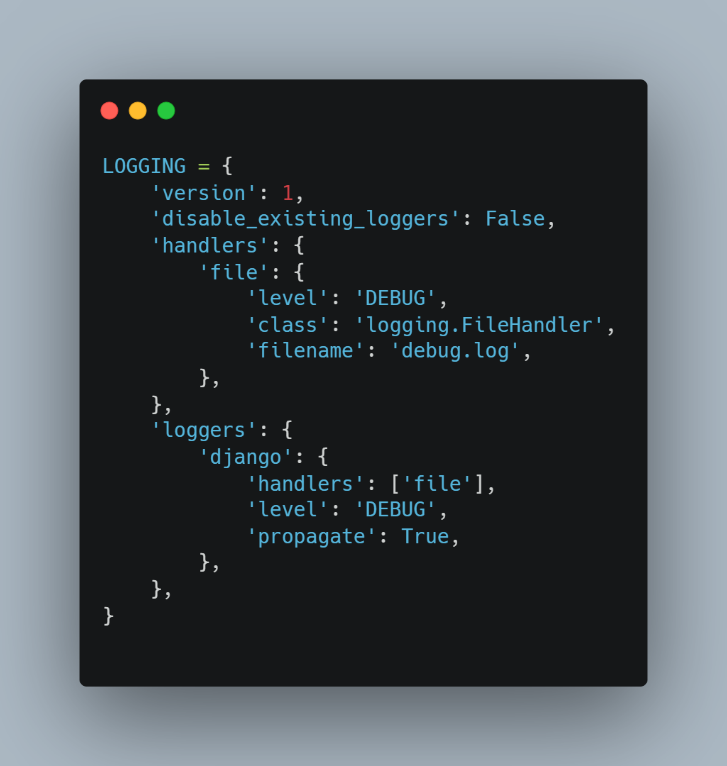
Ushbu ma'lumotlar muammo sababini aniqlashda juda foydali.

1. **Django Debug Toolbar**

django-debug-toolbar — ishlab chiqish jarayonida nosozliklarni aniqlash uchun kuchli vosita. Bu vosita ma'lumotlar bazasi so‘rovlarini, shablonlar, ko‘rinishlar va boshqa jarayonlar haqida ma'lumot beradi. Bu vosita bilan ilovaning turli jihatlarini tahlil qilish va optimallashtirishtirish mumkin.

1. **Loglash (Logging)**

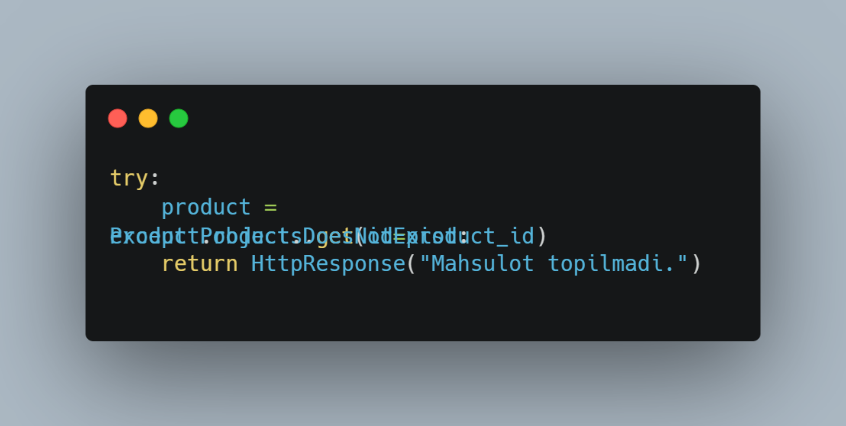
Django nosozliklarni kuzatish uchun loglash mexanizmini taklif etadi. settings.py faylida loglash sozlamalarini o‘rnatish mumkin:



Bu konfiguratsiya yordamida barcha xatoliklar debug.log fayliga yoziladi va ularni kuzatib borish mumkin.

**Nosozliklarni Tahlil Qilish va Bartaraf Etish**

* try-except bloklaridan foydalanish: Kodda kutilmagan xatoliklar yuz berganda, try-except bloklari yordamida xatolikni ushlash va foydalanuvchiga ma'lumotli xabar ko‘rsatish mumkin.



**Unit Testlar**

* Django’dagi unittest moduli yordamida dasturiy ta'minotning har bir qismi sinovdan o‘tkazilishi mumkin. Bu jarayon nosozliklarni dastlabki bosqichlarda aniqlashga yordam beradi.

# C.P5:

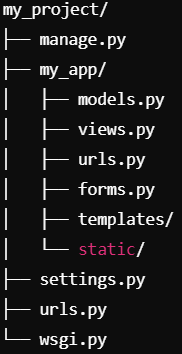
Dasturiy ta'minot ishlab chiqishda kodlash standartlariga rioya qilish juda muhim. Kodlash standartlari dasturiy ta'minotning sifati, o‘qilishi va boshqaruvchanligini oshiradi, ayniqsa jamoaviy loyihalarda. Django framework’dan foydalanishda quyidagi kodlash standartlariga amal qildim:

**PEP 8 – Python Kodlash Standarti**  
Django ilovasidagi barcha kodlarni Python uchun standart kodlash qoidalariga moslashtirishga harakat qildim. PEP 8 quyidagi qoidalarga urg‘u beradi:

1. **Indentatsiya (bo‘sh joylar):** Har bir ichki qator 4 ta bo‘sh joy bilan boshlangan.
2. Qator uzunligi: Qator uzunligi 79 belgidan oshmasligi kerak.
3. Bo‘sh qatorlar: Funksiyalar va sinflar orasida kamida 2 ta bo‘sh qator qoldirilgan.
4. Nomlash: O‘zgaruvchi va funksiyalar nomi kichik harflarda, kerak bo‘lsa, pastki chiziq (snake\_case) bilan ajratilgan.

Django Loyihasi Tuzilishi

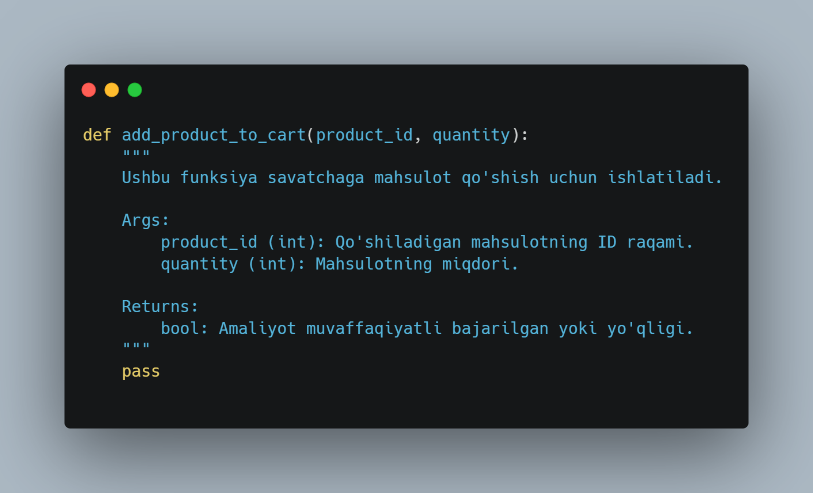
Django ilovasining kodlari standart tuzilma asosida joylashtirilgan:



Bu tuzilma kodni modulga ajratishni osonlashtiradi va loyiha bo‘limlarini boshqarishni soddalashtiradi.

**Kommentariyalar va Hujjatlash**

* Har bir funksiyaning vazifasi izohlar bilan tushuntirilgan.
* Django’da ishlatilgan murakkab funksiyalar uchun docstring yozilgan:



**Kodning Tashqi Kutubxonalar bilan Uyg‘unligi**

* Django o‘zining ichki kutubxonalariga asoslangan kodlash qoidalariga rioya qilindi.

**Errorlar va Loggerlar**

Xatoliklarni boshqarish uchun try-except bloklari va loglash mexanizmlaridan foydalandim.

Loglash uchun PEP 8 standartiga mos log formatlari ishlatilgan.

**Versionlash**

Kodda o‘zgartirishlar kiritilganda git orqali versiya nazorati boshqarilib, har bir muhim o‘zgarish tavsifi bilan hujjatlashtirilgan.

**Testlash**

Kodni to‘g‘ri ishlashini ta’minlash uchun unittest yordamida asosiy funksiyalarni sinovdan o‘tkazdim. Bu sifatni ta’minlash uchun muhimdir.

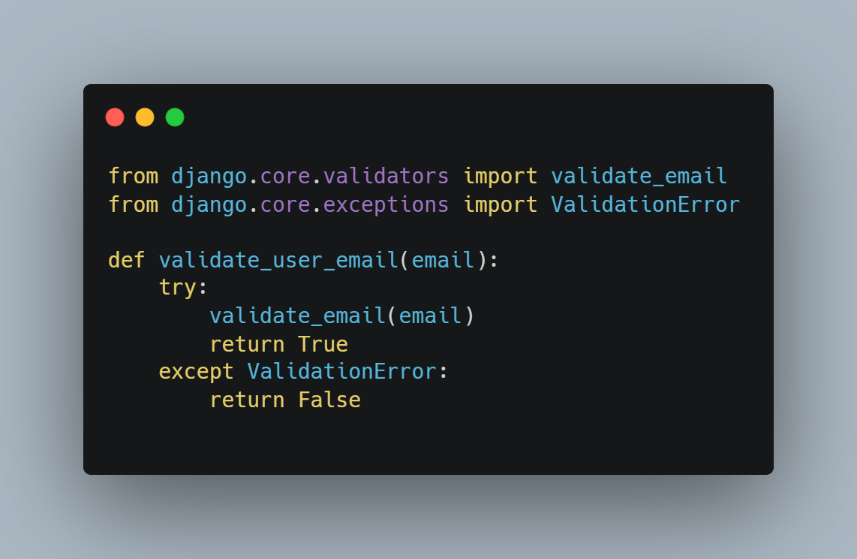
Ushbu kodlash standartlariga rioya qilish natijasida loyiha sifati yaxshilandi va boshqarilishi oson bo‘ldi. Bundan tashqari, kodni o‘qish va tushunish jarayoni soddalashtirildi.

# D.M4:

Nosozliklarni bartaraf etish (debugging) jarayoni dasturiy ta'minotni ishlab chiqishda xavfsizlik va mustahkamlikni ta'minlashning ajralmas qismidir. Bu jarayon kodda xatolarni topish, tahlil qilish va tuzatish bilan shug‘ullanadi. Quyida bu jarayonning xavfsiz va mustahkam dasturlarni yaratishga qanday hissa qo‘shishi haqida izoh berilgan:

1. **Kodning Mustahkamligi**
   * Nosozliklarni bartaraf etish jarayonida dasturiy kodda yuzaga keladigan barcha kutilmagan xatolar aniqlanadi va tuzatiladi. Masalan:
     + Xatoliklar dastur ishlashini to‘xtatib qo‘ymasligi uchun try-except bloklaridan foydalanish.
     + Bu yondashuv dasturiy ta'minotning uzluksiz ishlashini ta'minlaydi.
2. **Ma'lumotlarning Xavfsizligi**

* Debugging yordamida xavfsizlikka tahdid soluvchi muammolar (masalan, SQL Injection yoki XSS xatoliklari) aniqlanadi va bartaraf qilinadi.
* Masalan, Django’da foydalanuvchi kiritmalarini validatsiya qilish orqali kutilmagan xatolarni oldini olish mumkin:



1. **Resurslardan Samarali Foydalanish**
   * Debugging jarayoni orqali resurslarni (masalan, ma'lumotlar bazasi so‘rovlari, keshlash) ortiqcha ishlatadigan joylar aniqlanadi va optimallashtiriladi. Bu, ayniqsa, katta hajmdagi ma'lumotlar bilan ishlashda muhimdir.
2. **Foydalanuvchi Tajribasini Yaxshilash**
   * Debugging foydalanuvchiga aniq va foydali xatolik xabarlarini taqdim etishga yordam beradi. Masalan, serverda muammo yuz berganda foydalanuvchini chalkashtirmaslik uchun foydali xabarlar ko‘rsatiladi.

Nosozliklarni bartaraf etish jarayoni nafaqat dasturdagi xatolarni tuzatish bilan cheklanmay, balki dastur xavfsizligi va mustahkamligini ta'minlash uchun ham muhimdir. Ushbu jarayon dasturiy ta'minotning ishonchliligini oshiradi, foydalanuvchilarning qoniqishini ta'minlaydi va xavfsizlikka oid muammolarni oldini olishga yordam beradi

# D.D4:

Kodlash standartlari dasturiy ta'minot ishlab chiqish jarayonida samaradorlikni oshirish va dastur sifatini ta'minlashda juda muhim rol o‘ynaydi. Quyida kodlash standartlarining jamoa va individual dasturchi uchun afzalliklari va ahamiyati tanqidiy baholanadi:

**1. Jamoaviy Ishlashda Kodlash Standartlarining Ahamiyati**

* **Bir xil kod uslubi:** Jamoa a'zolari yagona kodlash qoidalariga rioya qilsa, kodni tushunish va tahrirlash osonlashadi. Har bir dasturchi bir xil formatda yozgan kod boshqalar uchun tushunarli va boshqarilishi qulay bo‘ladi.
  + Masalan, barcha funksiya nomlarini snake\_case usulida yozish, sinflarni esa CamelCase usulida nomlash qoidalari jamoaviy ishda tartibni saqlaydi.
* **Murakkablikni kamaytirish:** Bir xil standartlar qo‘llanganda murakkab jamoaviy loyihalarni boshqarish osonlashadi.  
  Misol: Django’da views.py, models.py kabi fayllarda turli jarayonlarni tartibga solish.
* **O‘quvchanlik:** Har bir dasturchi boshqa a'zo yozgan kodni tezda o‘qib, tahlil qilishi va kerakli o‘zgartirishlarni amalga oshirishi mumkin. Standartlar yo‘q bo‘lsa, kodni tushunish uchun ko‘proq vaqt sarflanadi.

**2. Individual Dasturchilar Uchun Afzalliklari**

* **Qayta foydalanish:** Standart kodlash individual dasturchiga yozilgan kodni qayta ishlatishda yordam beradi.
  + Masalan, Django’dagi Model va Form sinflari o‘rtasida moslikni saqlash.
* **Xatoliklarni kamaytirish:** Kodlash standartlariga amal qilish dasturchiga xatoliklarni oldindan aniqlash va tuzatishda yordam beradi.
* **Professional rivojlanish:** Standartlarga mos kod yozish individual dasturchining texnik malakasini oshiradi va uni jamoaviy loyihalarda ishlashga tayyorlaydi.

**3. Kodlash Standartlariga Amal Qilmaslikning Salbiy Tomonlari**

* **Qiyin tushunish:** Har bir dasturchi o‘z uslubida kod yozsa, bu boshqa dasturchilar uchun kodni o‘qishni qiyinlashtiradi. Bu jamoaviy ishda kechikishlarga olib kelishi mumkin.
* **Xatolarning ko‘payishi:** Standartlarga amal qilmaslik xatoliklarni aniqlash va debugging jarayonini murakkablashtiradi.
* **Noto‘g‘ri hujjatlashtirish:** Yagona format bo‘lmasa, kodga tegishli hujjatlar bir-biriga mos kelmasligi mumkin.

**4. Kodlash Standartlarining Texnik Tomondan Afzalliklari**

* **Testlash:** Yaxshi tuzilgan kodlarni testlash osonroq bo‘ladi. Django’dagi unittest yordamida test yozish uchun kodning to‘g‘ri tuzilishi zarur.
* **Xavfsizlik:** Kodni xavfsizlik qoidalariga mos ravishda yozish (masalan, foydalanuvchi kiritmalarini to‘g‘ri validatsiya qilish) ilovaning zaifliklarini kamaytiradi.

Kodlash standartlari jamoa va individual dasturchilar uchun quyidagi afzalliklarni taqdim etadi:

* Jamoaviy ishlashni samarali qiladi.
* Kodni tushunishni va boshqarishni osonlashtiradi.
* Xatoliklarni kamaytiradi va xavfsizlikni oshiradi.
* Loyihaning mustahkamligi va kelajakda kengaytirilishi uchun poydevor yaratadi.

Kodlash standartlariga rioya qilish dasturiy ta'minotni ishlab chiqish jarayonida eng yaxshi amaliyot hisoblanadi. Ushbu standartlarga amal qilmaslik loyihaning sifati va boshqariluvchanligiga salbiy ta'sir ko‘rsatishi mumkin.

# Xulosa

Yuqoridagi loyiha doirasida NewEra Cash & Carry chakana savdo do‘koni uchun avtomatlashtirilgan buyurtma boshqaruv tizimi ishlab chiqildi. Ushbu tizim, Python va Django asosida yaratilgani sababli, samarali ma'lumotlar boshqaruvi va foydalanuvchilar uchun qulay interfeysni ta'minlaydi. Loyihaning maqsadi – do‘konning kundalik operatsiyalarini optimallashtirish va ma'lumotlarning aniqligi, xavfsizligi hamda uzluksiz ishlashini ta'minlashdir.

Tizim quyidagi afzalliklarni taqdim etadi:

1. Administrator uchun qulaylik: Mahsulotlarni qo‘shish, o‘zgartirish va o‘chirish kabi funksiyalarni bajarish imkoniyati.
2. Mijozlar uchun qulaylik: Mahsulotlar katalogini ko‘rish va buyurtma berish jarayonining soddaligi.
3. Ma'lumotlar xavfsizligi va yaxlitligi: Har bir jarayon real vaqt rejimida kuzatiladi va tizimdagi ma'lumotlarning ishonchliligi kafolatlanadi.
4. Loyiha davomida zamonaviy dasturlash usullari qo‘llanildi, shuningdek, kodni optimallashtirish va xatolarni oldini olish bo‘yicha ilg‘or amaliyotlar amalga oshirildi. Tizim mustahkam va kengaytiriluvchan bo‘lib, kelajakdagi ehtiyojlarni qondirish uchun moslashuvchanlikni ta'minlaydi.

Ushbu loyiha nafaqat ma'lumotlar boshqaruvi samaradorligini oshiradi, balki mijozlarga sifatli xizmat ko‘rsatishga ham yordam beradi. Shu sababli, avtomatlashtirilgan buyurtma tizimi NewEra Cash & Carry do‘koni uchun innovatsion va samarali yechim bo‘lib xizmat qiladi.

adabiyotlar ro'yxati

1. Dennis, A., & Haley, W. (2009). Systems Analysis and Design. John Wiley & Sons Ltd.
2. Ferguson, J. (2014). BDD in Action: Behavior-Driven Development for the Whole Software Lifecycle. Manning.
3. Lejk, M., & Deeks, D. (2002). An Introduction to Systems Analysis Techniques. 2nd Edition. Addison-Wesley.
4. Murch, R. (2012). The Software Development Lifecycle: A Complete Guide. Kindle.
5. Albahari, J., & Albahari, B. (2021). C# 10.0 in a Nutshell: The Definitive Reference. O'Reilly Media.
6. Freeman, A., & Sanderson, A. (2021). Pro ASP.NET Core 5: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. Apress.
7. Microsoft Documentation. (n.d.). Entity Framework Core. Retrieved from https://learn.microsoft.com.
8. Microsoft Documentation. (n.d.). ASP.NET Core Authentication and Authorization. Retrieved from https://learn.microsoft.com.
9. Fowler, M. (2019). Refactoring: Improving the Design of Existing Code. Addison-Wesley.
10. Thomas, D., & Hunt, A. (1999). The Pragmatic Programmer: Your Journey to Mastery. Addison-Wesley.