Python bilginizi geliştirmek için güzel bir proje fikrim var! Bu projede, kullanıcıların yapılacak görevleri (To-Do List) ekleyebileceği, düzenleyebileceği, silebileceği ve görevlerin durumunu güncelleyebileceği bir uygulama oluşturacağız. Bu uygulama aşağıdaki özellikleri içerecek:

1. **Class ve Fonksiyon Kullanımı**: Projeyi, nesne yönelimli programlama (OOP) ilkelerine göre yapılandırarak class ve fonksiyonları etkin şekilde kullanacağız.
2. **JSON İşlemleri**: Görevleri kalıcı hale getirmek için görevleri JSON dosyasına kaydedecek ve dosyadan okuyacağız.
3. **Dinamik Kodlama**: Görev yönetimini esnek ve dinamik hale getirmek için sınıfları ve fonksiyonları ayrıntılı bir şekilde yapılandıracağız.

**Proje Adımları**

1. **Görev Sınıfı (Task Class)**: Her bir görev için bir sınıf tanımlayacağız. Bu sınıf, görev adı, açıklaması ve durum bilgisi (tamamlandı veya tamamlanmadı) gibi özelliklere sahip olacak.
2. **Görev Yönetimi Sınıfı (TaskManager Class)**: Bu sınıf, görevleri ekleme, silme, güncelleme ve kaydetme gibi işlemleri yönetecek.
3. **JSON Okuma ve Yazma**: Görevlerin JSON dosyasına kaydedilmesi ve dosyadan okunması sağlanacak.

Aşağıda projenin temel kod yapısını veriyorum:

python

Kodu kopyala

import json

# 1. Görev Sınıfı

class Task:

def \_\_init\_\_(self, title, description, completed=False):

self.title = title

self.description = description

self.completed = completed

def to\_dict(self):

"""Görev nesnesini sözlüğe dönüştür."""

return {

'title': self.title,

'description': self.description,

'completed': self.completed

}

@classmethod

def from\_dict(cls, data):

"""Sözlükten bir görev nesnesi oluştur."""

return cls(data['title'], data['description'], data.get('completed', False))

# 2. Görev Yönetimi Sınıfı

class TaskManager:

def \_\_init\_\_(self, file\_path='tasks.json'):

self.file\_path = file\_path

self.tasks = self.load\_tasks()

def add\_task(self, task):

"""Yeni bir görev ekle."""

self.tasks.append(task)

self.save\_tasks()

def remove\_task(self, title):

"""Görev sil."""

self.tasks = [task for task in self.tasks if task.title != title]

self.save\_tasks()

def update\_task(self, title, \*\*updates):

"""Görevi güncelle."""

for task in self.tasks:

if task.title == title:

task.title = updates.get('title', task.title)

task.description = updates.get('description', task.description)

task.completed = updates.get('completed', task.completed)

self.save\_tasks()

def list\_tasks(self):

"""Görevleri listele."""

for task in self.tasks:

status = "Tamamlandı" if task.completed else "Tamamlanmadı"

print(f"{task.title} - {task.description} [{status}]")

def save\_tasks(self):

"""Görevleri JSON dosyasına kaydet."""

with open(self.file\_path, 'w') as file:

json.dump([task.to\_dict() for task in self.tasks], file, indent=4)

def load\_tasks(self):

"""JSON dosyasından görevleri yükle."""

try:

with open(self.file\_path, 'r') as file:

return [Task.from\_dict(task) for task in json.load(file)]

except (FileNotFoundError, json.JSONDecodeError):

return []

# Kullanıcı arayüzü (Basit Konsol)

def main():

manager = TaskManager()

while True:

print("\n1. Görev Ekle\n2. Görev Sil\n3. Görev Güncelle\n4. Görevleri Listele\n5. Çıkış")

choice = input("Bir seçenek girin: ")

if choice == '1':

title = input("Görev başlığı: ")

description = input("Görev açıklaması: ")

manager.add\_task(Task(title, description))

elif choice == '2':

title = input("Silinecek görevin başlığı: ")

manager.remove\_task(title)

elif choice == '3':

title = input("Güncellenecek görevin başlığı: ")

new\_title = input("Yeni başlık (boş bırakılırsa değişmez): ")

description = input("Yeni açıklama (boş bırakılırsa değişmez): ")

completed = input("Tamamlandı mı? (evet/hayır): ").strip().lower() == 'evet'

manager.update\_task(title, title=new\_title or title, description=description, completed=completed)

elif choice == '4':

manager.list\_tasks()

elif choice == '5':

break

else:

print("Geçersiz seçim. Lütfen tekrar deneyin.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Açıklama**

1. **Task Sınıfı**: Task sınıfı, her bir görevin başlık, açıklama ve tamamlanma durumu özelliklerine sahiptir. Ayrıca to\_dict ve from\_dict metodları sayesinde, görev nesnesini JSON formatında kolayca saklayabiliriz.
2. **TaskManager Sınıfı**: TaskManager sınıfı, görevleri ekleme, silme, güncelleme ve listeleme işlemlerini içerir. Ayrıca görevleri tasks.json dosyasına kaydeder ve gerektiğinde bu dosyadan yükler.
3. **JSON Okuma ve Yazma**: save\_tasks metodu görevleri JSON formatında kaydederken, load\_tasks metodu dosyayı okuyup görevlere dönüştürür.
4. **Basit Konsol Arayüzü**: main fonksiyonunda kullanıcıdan giriş alarak yukarıdaki işlemler yapılır.

Bu projeyi genişletebilir, daha fazla özellik ekleyebilir veya bir GUI arayüzü oluşturabilirsiniz. Özellikle sınıfları kullanarak görevlerin nasıl yönetildiğini incelemek, dinamik kodlama açısından yararlı olacaktır.

4o

**Şunu dedin:**

yukarida belirttigin projeyi bana adim adim yapmam gerekenleri gorev olarak yaz. ben de bu gorevleri takip ederek projeyi tamamalamaya calisayim

**ChatGPT:**

ChatGPT

Projeyi adım adım takip edebilmeniz için görevleri listeledim. Bu görevleri sırayla yaparak projeyi tamamlayabilirsiniz.

**Görev Listesi**

**1. Adım: Proje Klasörünü Oluştur**

* **Görev:** Yeni bir klasör oluşturun ve adını todo\_list\_project olarak belirleyin.
* **Görev:** tasks.json adlı bir dosya oluşturun. (Bu dosya, görevlerinizin JSON formatında kaydedileceği dosya olacak.)
* **Görev:** Proje klasörünüzde todo\_app.py adında bir Python dosyası oluşturun. Bu dosya, tüm kodları içerecek.

**2. Adım: Task Sınıfını Tanımla**

* **Görev:** todo\_app.py dosyasını açın ve Task sınıfını tanımlayın.
* **Görev:** \_\_init\_\_ metodunu yazın ve görev başlığı, açıklama ve tamamlanma durumu (default olarak False) için özellikler ekleyin.
* **Görev:** Görev nesnesini JSON'a dönüştürmek için to\_dict adında bir metod ekleyin. Bu metod, görev nesnesini bir sözlük formatına çevirsin.
* **Görev:** JSON'dan göreve dönüştürmek için from\_dict adlı bir sınıf metodunu ekleyin. Bu metod, JSON'daki veriyi bir Task nesnesine çevirsin.

**3. Adım: TaskManager Sınıfını Tanımla**

* **Görev:** TaskManager adında bir sınıf oluşturun.
* **Görev:** Bu sınıfa \_\_init\_\_ metodunu ekleyin ve file\_path adında bir parametre alın (default olarak 'tasks.json'), böylece JSON dosyasının yolunu belirtmiş olun.
* **Görev:** load\_tasks metodunu yazın, JSON dosyasındaki görevleri yükleyip self.tasks listesine eklesin. JSON dosyası yoksa veya okunamıyorsa boş bir liste döndürsün.
* **Görev:** save\_tasks adında bir metod yazın ve görev listesini JSON formatında tasks.json dosyasına kaydedin.

**4. Adım: Görev Ekleme İşlemi**

* **Görev:** add\_task adında bir metod oluşturun.
* **Görev:** Bu metod bir Task nesnesi alacak ve bu görevi self.tasks listesine ekleyecek.
* **Görev:** Yeni bir görev eklendiğinde save\_tasks metodunu çağırarak dosyaya kaydedin.

**5. Adım: Görev Silme İşlemi**

* **Görev:** remove\_task adında bir metod ekleyin.
* **Görev:** Bu metod, görev başlığına göre bir görevi self.tasks listesinden silecek.
* **Görev:** Görev silindikten sonra save\_tasks metodunu çağırarak dosyayı güncelleyin.

**6. Adım: Görev Güncelleme İşlemi**

* **Görev:** update\_task adında bir metod oluşturun.
* **Görev:** Bu metod, verilen başlığa göre bir görevi bulacak ve güncelleme işlemlerini gerçekleştirecek.
* **Görev:** Güncellemeler yapılınca save\_tasks metodunu çağırarak dosyaya kaydedin.

**7. Adım: Görev Listeleme İşlemi**

* **Görev:** list\_tasks adında bir metod oluşturun.
* **Görev:** Bu metod self.tasks listesindeki tüm görevleri sırayla yazdıracak.
* **Görev:** Her görev için başlık, açıklama ve tamamlanma durumunu konsola yazdırın.

**8. Adım: Basit Bir Kullanıcı Arayüzü (Console) Oluştur**

* **Görev:** main adında bir fonksiyon oluşturun.
* **Görev:** Kullanıcıdan giriş alarak görevleri ekleme, silme, güncelleme ve listeleme işlemlerini yönetmesini sağlayın.
* **Görev:** main fonksiyonunda, kullanıcı seçimlerine göre add\_task, remove\_task, update\_task ve list\_tasks metodlarını çağırın.
* **Görev:** Kullanıcı 5 girdiğinde programdan çıkmasını sağlayın.

**9. Adım: Uygulamayı Çalıştır ve Test Et**

* **Görev:** Konsolda uygulamayı çalıştırmak için python todo\_app.py komutunu kullanın.
* **Görev:** Yeni görevler ekleyin, görevleri listeleyin, güncelleyin ve silin. Tüm işlemlerin düzgün çalıştığını kontrol edin.
* **Görev:** JSON dosyasına kaydedildiğinden emin olun. tasks.json dosyasını açarak görevlerin kaydedilip kaydedilmediğini kontrol edin.

**10. Adım: Hataları Gözden Geçir ve İyileştirme Yap**

* **Görev:** Kodda hata varsa düzeltin, gerekirse try-except blokları ekleyerek hata yönetimi yapın.
* **Görev:** Kod yapısını ve dosya düzenini kontrol ederek projeyi tamamlayın.

Bu adımları takip ederek projeyi adım adım tamamlayabilirsiniz. Her adımı ayrı ayrı tamamladıktan sonra bir sonraki adıma geçebilirsiniz. Takıldığınız noktalarda bana sorabilirsiniz!