Python kütüphaneleri ve framework'leri ile uygulama geliştirme konularını ele almaktadır.

Python Kütüphaneleri

- Python kütüphaneleri, belirli programlama görevlerini basitleştiren ve hızlandıran araçlar içerir. Örneğin, NumPy gelişmiş matematiksel hesaplamalar için, Pandas veri manipülasyonu ve analizi için, Matplotlib ise veri görselleştirmesi için kullanılır.
- Web uygulama geliştirme için önemli kütüphaneler arasında Requests (HTTP isteklerini kolaylaştırır), BeautifulSoup (web sayfalarından bilgi kazıma için) ve SQLAlchemy (SQL araç seti ve ORM aracı) bulunmaktadır.

Framework'ler

- Framework'ler, uygulama geliştirme için önceden tanımlanmış yapılar sunar ve geliştiricilere kod yazma ve düzenleme konusunda rehberlik eder.
- Django, Flask ve Web2Py gibi framework'ler, geliştiricilerin daha az kod yazarak daha fazla işlevsellik eklemelerine olanak tanır ve güvenlik özellikleri ile veritabanı yönetimini kolaylaştırır.

Kütüphaneler ve Framework'ler Arasındaki Farklar

- Framework, bir uygulamanın temel akışını ve mimarisini içerirken, Python kütüphanesi yalnızca belirli işlevsellikleri yerine getiren paketler içerir.

Flask Nedir?

- Flask, web uygulamaları oluşturmak için kullanılan bir mikro framework'tür ve kullanıcıyı belirli bir araç setine bağlamaz.
- 2004 yılında Armin Ronacher tarafından yaratılmıştır ve kolay kullanımı ile popülerlik kazanmıştır.

Flask'ın Temel Özellikleri

- Geliştirme modunda çalışan bir web sunucusu ve hata ayıklama aracı sunar.
- Uygulama günlükleri için standart Python logging kullanır ve test etme özellikleri ile test odaklı bir yaklaşım benimsemeye olanak tanır.

Flask'ın Ek Özellikleri

- Statik varlıkları (CSS, JavaScript, resimler) destekler ve dinamik sayfalar oluşturmak için Jinja şablonlama framework'ünü kullanır.
- RESTful hizmetler için dinamik URL'ler ve yönlendirme desteği sağlar.

Flask Uygulaması Oluşturma

- Flask sınıfını içe aktararak bir uygulama nesnesi oluşturun ve bu nesneyi app.py dosyasında tanımlayın.
- İlk yönlendirmeyi tanımlamak için @app dekoratörünü kullanarak bir yol belirleyin ve istemciye bir mesaj döndürün.

Uygulamanın Çalıştırılması

- Uygulamayı çalıştırmadan önce FLASK_APP ve FLASK_ENV gibi sistem ortam değişkenlerini ayarlayın.
- Uygulamayı çalıştırmak için Flask run komutunu kullanın ve tarayıcıda http://localhost:5000 adresine giderek yanıtı kontrol edin.

JSON Yanıtları Döndürme

- Flask uygulamanızdan JSON döndürmek için bir sözlük veya liste gibi serileştirilebilir bir nesne döndürebilirsiniz.
- jsonify yöntemini kullanarak anahtar-değer çiftlerini JSON formatında döndürmek mümkündür.

Flask Request Object

- **Request Nesnesi**: Flask sunucusuna yapılan her HTTP isteği, Flask.Request sınıfından oluşturulan bir request nesnesi ile işlenir. Bu nesne, istemcinin gönderdiği başlıklar, URL ve sunucu adresi gibi bilgileri içerir.
- **Veri Erişimi**: Request nesnesi üzerinden, GET ve POST isteklerinden veri almak için çeşitli yöntemler kullanılabilir. Örneğin, args ile sorgu parametrelerine, json ile JSON verilerine, form ile form verilerine erişim sağlanabilir.

Flask Response Object

- **Response Nesnesi**: Flask, her istemci isteğine yanıt vermek için bir response nesnesi sağlar. Bu nesne, yanıtın durum kodunu, başlıklarını ve içerik türünü içerir.
- **Yanıt Oluşturma**: make_response ile özel bir yanıt oluşturulabilir.
  Ayrıca, jsonify metodu ile otomatik olarak JSON formatında bir yanıt oluşturulabilir.

Dış API'lere Erişim

- Python requests kütüphanesi kullanılarak dış API'lere istek yapılabilir.
- API'den alınan JSON verisi, istemciye geri döndürülebilir; ayrıca sonuçlar işlenebilir.

Dinamik Yollar

- Flask, URL'ye dinamik parametreler eklemeye olanak tanır.
- Örneğin, ISBN numarasını URL'den alarak kitap bilgilerini döndüren bir endpoint oluşturulabilir.

Parametre Türleri

- Flask, parametre türlerini ayarlamaya izin verir; bu, gelen isteklerin doğrulanmasında kullanılır.
- Basit türler (string, int, float) ve karmaşık türler (path, UUID) gibi farklı parametre türleri tanımlanabilir.

HTTP Durum Kodları

- Her HTTP yanıtı, API hizmetlerinin döndürebileceği farklı hata ve başarı durumlarını belirten üç haneli bir kod içerir. Bu kodlar 100 ile 599 arasında değişir ve 100'lük gruplar halinde kategorize edilmiştir.
- 200 ile 299 arasındaki kodlar, isteğin alındığını ve işlemin başarılı olduğunu gösterirken, 400 ile 499 arasındaki kodlar istekte bir hata olduğunu belirtir.

Flask'ta Hata Yönetimi

- Flask, varsayılan olarak 200 OK durum kodunu döndürür. Ancak, geliştiriciler yanıtla birlikte durum kodunu bir demet içinde gönderebilirler.
- Hata durumları için özel hata işleyicileri oluşturmak mümkündür. Örneğin, 404 hatası için "API not found" mesajı döndürebiliriz.

Örnekler ve Uygulama

- Örnek bir search_response metodu, veritabanındaki bir kaynak için sorgu parametresini arar. Kaynak mevcutsa 200, mevcut değilse 404 durum kodu döner.
- Hata mesajlarını uygulama düzeyinde işlemek için Flask, 404 ve 500 hataları için özel işleyiciler sağlar.

Flask Nedir?

- Flask, Python ile hızlı ve kolay bir şekilde web uygulamaları oluşturmak için kullanılan bir mikro çerçevedir.
- CRUD işlemlerini destekler; bu, POST, PUT, GET, PATCH ve DELETE istekleri ile veri oluşturma, okuma, güncelleme ve silme anlamına gelir.

Flask ile Web Uygulaması Oluşturma

- Flask paketini yüklemek için pip install flask komutu kullanılır.
- Flask sınıfı örneği oluşturulur ve uygulama başlatılır. Örneğin, app = Flask(__name__) ile bir uygulama nesnesi oluşturulur.
- Temel bir GET isteği ile "hello world" yanıtı döndüren bir uygulama örneği verilmektedir.

Şablonlar ve Statik Dosyalar

- Flask, statik ve dinamik HTML sayfalarını sunmak için şablonlar kullanır. Statik dosyalar "static" klasöründe, dinamik sayfalar ise "templates" klasöründe bulunur.
- Dinamik sayfalar, URL veya istek parametreleri aracılığıyla geçirilen değerlerle doldurulur.