

**T.C.**

**AKSARAY ÜNİVERSİTESİ**

**İKTİSADİ VE İDARİ BİLİMLER FAKÜLTESİ**

**YÖNETİM BİLİŞİM SİSTEMLERİ BÖLÜMÜ**

**YAPAY ZEKA**

**DOĞAL DİL İŞLEME ÖDEVİ**

**RAMAZAN DELİGÖZ**

**170104005**

**DANIŞMAN**

**DR. ÖĞR. ÜYESİ TARIK YILMAZ**

**AKSARAY 2021**

Merhabalar.

Bu yazımızda ilk başta size Doğal Dil İşleme Natural Language Processing (NLP) ve Google API (Uygulama Programlama Arayüzü) hakkında bilgi verdikten sonra Python dilini kullanarak “Türkçe metinleri nasıl Türkçe olarak seslendiririz?” yani metni sese çevirme (text to speech) bunu adım adım anlatacağım. Yazdığımız programı uygulamaya dönüştürüp daha kullanışlı hale getirmek için de Python tkinter modülü ile API tasarımı yapacağız. Siz de hazırsanız bekletmeden doğal dil işleme hakkında bilgi vermekle başlayabiliriz.

**Doğal Dil İşleme (Natural Language Processing)**

Doğal dil işleme, İngilizce literatürde Natural Language Processing (NLP) diye geçmektedir. Doğal dil işleme, yapay zekanın bir alt kategorisidir. Bilgisayar dünyasında iki farklı dil vardır bunlardan biri makine dilleri yani programlama dilleri, diğeri ise doğal dillerdir. Doğal dillerden kasıt, insanların konuştuğu doğal dillerdir, bunlara; Türkçe, İngilizce ve Korece gibi diller örnek olarak verilebilir. İnsanların konuştuğu dili makinelerin alıp işlemesine doğal dil işleme denilmiştir.

Dil, insanoğlunun uygarlaşmasını sağlamakla birlikte zekasının giderek parlamasını da sağlamıştır. Kültür dediğimiz kavram dil kullanan ve iletişim kuran insanın sosyalleşme ürünüdür. Doğal dil işleme ile doğal dillerin kurallı yapısının çözümlenmesi veya yeniden üretilmesi amaçlanmıştır. Doğal dilin yapısını çözümlediğimiz ve üretim yapmaya başladığımız zaman, insanların makinelere olan beklentisi yükselmektedir ki keza, son yıllarda insanların makinelerle olan bağının giderek arttığına hepimiz şahit olmaktayız.

**Peki doğal dil işleme için yapılan çalışmalar nelerdir?**

* Metin (Text)
* Ses (Speech)

Metin verilerini doğrudan işleyip analiz yapılırken ses verileri üzerinde doğrudan işlem yapılamıyor, öncelikle ses verileri yazılı metinlere çevriliyor ve bu yazılı metinler doğal dil işleme modüllerine uygun hale getirilip makine tarafından işleniyor.

Doğal dil işleme ile ilgilenirken üzerinde çalışılan dilin, dil morfolojisi seviyelerini bilmek oldukça önemlidir, bunlar;

* Kelime bilimi (Morphological-Lexical)
* Sözdizimi (Syntactic)
* Anlamsal (Semantic)
* Söylevdir (Sragmatic-discourse)

Dil morfolojisinin aklınızda az çok yer edindiğini düşünüyorum. Doğal dil işleme terminolojisinin bir alt kümesi olan NLU, yani doğal dili anlama ile ilgilenecekseniz bu kavramlar üzerinde ciddi çalışmalar yapmanız gerekir çünkü zor bir sorun olarak kabul edilmekte. Makinenin insan gibi davranarak önüne gelen bir metni yorumlama yeteneğini kazandırılmaya çalışılması örnek olarak verilebilir.

* Doğal dil işleme kavramının ne olduğunu anladıktan sonra gelelim uygulama alanlarına.

**Uygulama alanlarına 10 farklı proje örnek verilebilir, bunlar**:

1. Metin Sınıflandırma ve Kategorizasyon (Text Classification and Categorization)
2. Adlandırılmış Varlık Tanıma (Named Entity Recognition (NER) )
3. Konuşma Bölümü Etiketleme (Part-of-Speech Tagging)
4. Anlamsal Ayrıştırma ve Soru Cevaplama (Semantic Parsing and Question Answering)
5. Yorum Bulma (Paraphrase Detection)
6. Dil Üretimi ve Çok Belgeli Özetleme (Language Generation and Multi-document Summarization)
7. Dil Çeviri (Machine Translation)
8. **Ses Tanıma (Speech Recognition)**
   1. **Metni Sese Çevirme (Text To Speech)**
   2. **Sesi Metne Çevirme (Speech To Text)**
9. Karakter Tanıma (Character Recognition)
10. Yazım Denetimi (Spell Checking)

Bütün uygulamaları tek tek açıklamayacağım en başta da dediğim gibi sadece Metni Sese Çevirme (Text To Speech)’yi Python üzerinde uygulayarak anlatacağım. Şimdi gelelim Google API (Uygulama Programlama Arayüzü) hakkında bilgi vermeye…

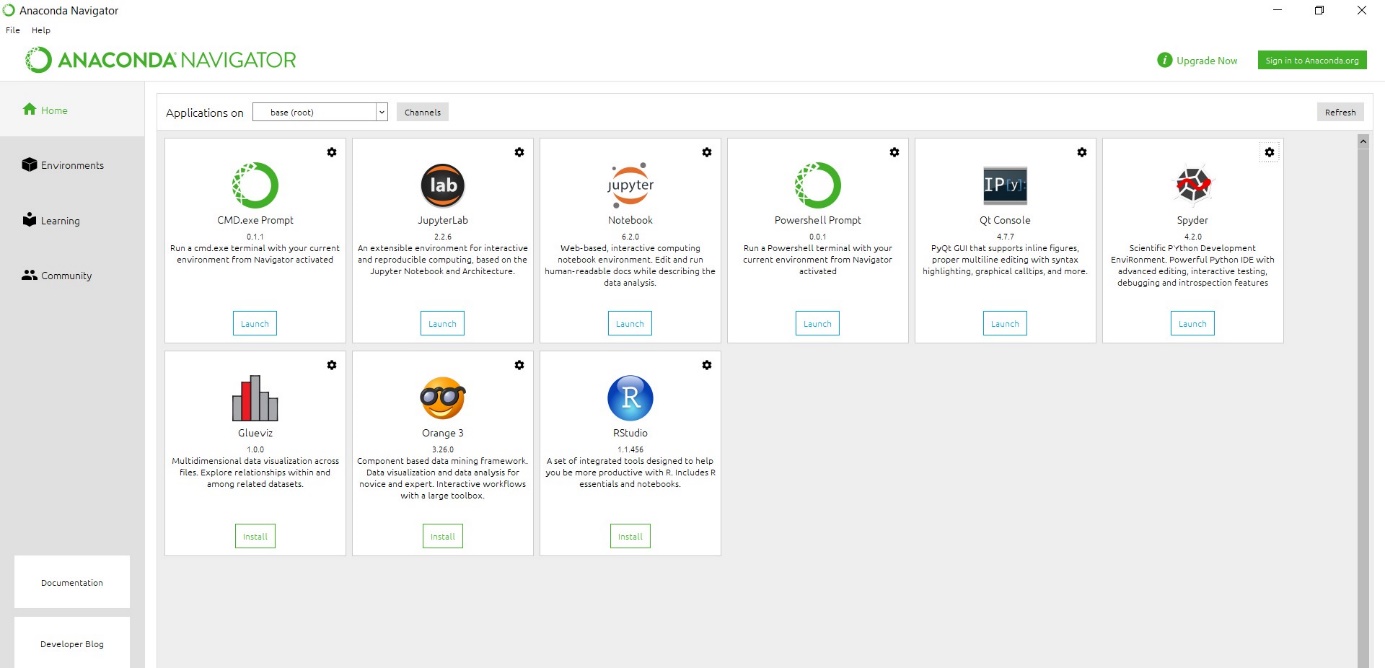
**API (Uygulama Programlama Arayüzü)**

**API,** iki uygulamanın birbiriyle konuşmasına olanak tanıyan bir yazılım aracısı olan Uygulama Programlama Arayüzü’nün kısaltmasıdır. WhatsApp gibi bir uygulamayı her kullandığınızda, her anlık mesaj gönderdiğinizde veya telefonunuzda hava durumunu her kontrol ettiğinizde bir API kullanıyorsunuz. **Google Cloud Speech API** geliştiriciler için metni sese dönüştürerek güçlü sinir ağı modelleri sağlar. ... Bir uygulamanın mikrofonuna dikte eden kullanıcıların metnini yazabilir, komuta ve denetimi sesle etkinleştirebilir veya diğer birçok kullanım durumunda ses dosyalarını kopyalayabilirsiniz.

Gerekli bilgileri tanımladığımıza göre artık ilk adımımız olan kurulumlar ile projemize başlayabiliriz.

* **Öncelikle Python dilinde kodlarımızı yazabilmek için Anaconda kurulumunu yapalım.**

Anaconda, veri bilimi ve benzeri bilimsel uygulamalar için python kullanmak isteyenlere hazırlanmış tümleşik bir python dağıtımıdır. Veri bilimi, yapay zeka vb konularında sıkça kullanılan kütüphanelerin yanı sıra jupyter notebook ve spyder gibi araçları da barındırır. Programı <https://www.anaconda.com/products/individual> adresinden kendi işletim sisteminize uygun versiyonu indirerek kurabilirsiniz. Anacondayı kurduğunuzda sisteminizde python, jupyter notebook ve spyder da kurulmuş olacak.



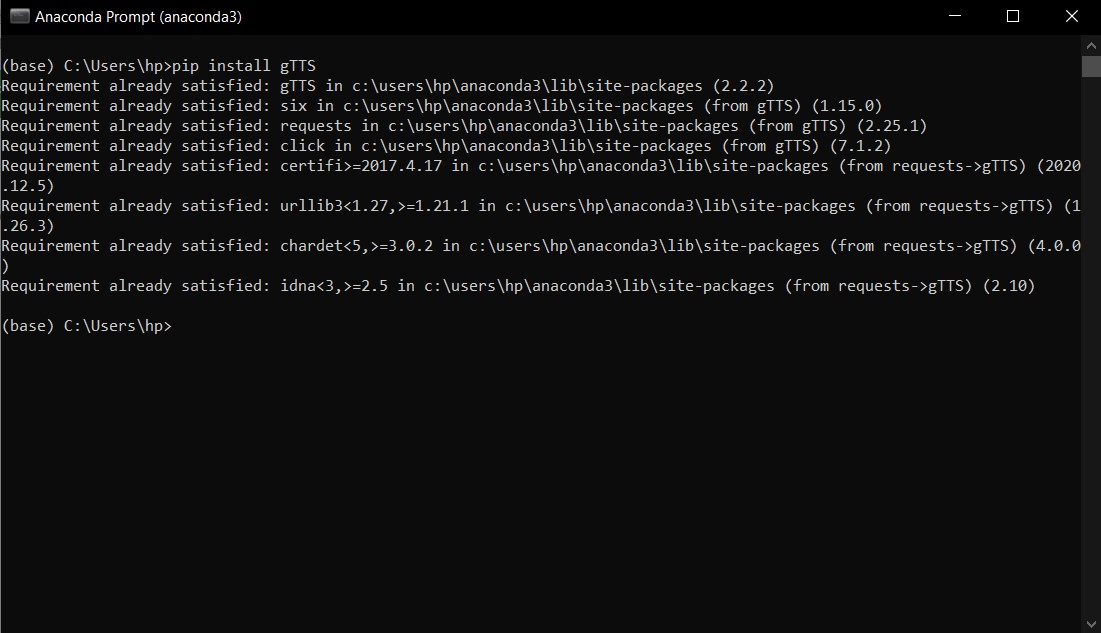
* **Şimdi de kod kısmında kullanacağımız gTTS modülünü kuralım.**

Google tarafından geliştirilen metinlerin seslere dönüştürmesine yardımcı olan gTTS (google Text To Speech) kütüphanesi aynı zamanda CLI (Command Line Interface) olarak da kullanılabilmektedir. Üretilen ses datalarını daha sonra üzerinde işlemler yapabilmek için bir dosyaya veya bytestring veri tipine dönüştürmektedir.

Bilgisayarınızda arama kısmına gelip “Anaconda Prompt” yazdıktan sonra açılan terminalde aşağıdaki komutu yazın.

**pip install gTTS**

Ben daha önceden indirdiğim için yükleme yapmadı, ancak sizde succesfully diyorsa yükleme tamamlanmış demektir.



Yüklemenin başarılı olup olmadığını Spyder dosyanızda test edebilirsiniz.

***from gtts import gTTS***

kod satırını yazdığınızda

***ModuleNotFoundError: No module named 'gtts'***

hatası almıyorsanız kurulum başarılı bir şekilde tamamlanmış demektir.

* **gTTS modülünü kurduktan sonra kod kısmına yavaş yavaş geçebiliriz artık**.

Modülümüzü import ettikten sonra dosya işlemleri için (dosya okuma, dosyaya yazma, dosya kaydetme) 10. ve 11. satırdaki kodlarımızı yazıyoruz.

.txt formatındaki bir metni okuyabilmesi için öncelikle bunu sistemimize okutmamız gerekiyor.



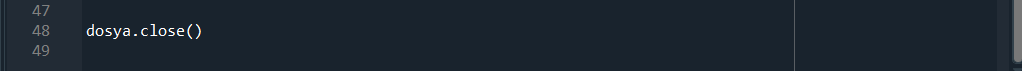
Dosya diye bir değişken oluşturuyorum ve os modülünden gelen open fonksiyonunu kullanıyorum. Bu fonksiyonda ilk parametre olarak dosyanın adını veriyoruz. Ardından düzgün okuyabilmesi için bu fonksiyonun bir de encoding değeri bulunmaktadır. Sistemimiz Türkçe karakterleri de okuyabilsin istediğimiz için bu değeri utf-8 olarak ayarlıyoruz. Bu şekilde dosyamızın yolunu gösterdik şimdi de okuyabiliriz. İçeriğini yazi diye bir değişkene atıyorum. 20. satırdaki kod ile metnin tamamını okuyup yazi değişkeninin içerisine aktaracak. Okuduğunu test etmek için ***print(yazi)*** komutu ile console ekranında gösterebilirsiniz.

******Şimdi bu metni gTTS modülüne aktarmamız gerekiyor. Çünkü sese çevirme işlemini o yapacak.

“cikti” diye bir değişken oluşturuyorum. Bu değişkene gTTS modülünü çağırıyorum. Modülün ilk değeri text, yani neyi çevireceğini soruyor. Biz metnimizi “yazi” değişkeninin içine atadığımız için “yazi” yazdık. İkinci özellik olarak hangi dilde çevireceğini soruyor. Biz Türkçe olmasını istediğimiz için “lang” değerini ‘tr’ olarak belirliyoruz. Son olarak çevirme işlemi sistemde default değeri olarak yani varsayılan değeri olarak slow=True geliyor. Biz o değeri False yaparak okuma işlemini hızlandırmış oluyoruz.



Az önce elde ettiğimiz sesi bir mp3 dosyası olarak kaydedelim. gTTS modülü farklı dosya türlerinde de kaydetmeye imkan sağlıyor.

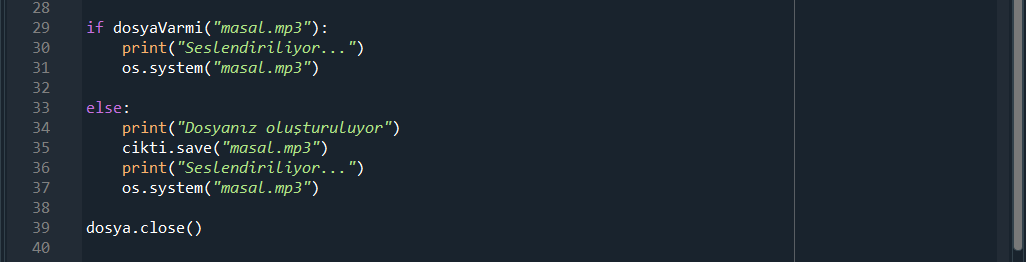
Bu işlemlerin ardından sistem için önemli olan dosya kapatma işlemlerini de unutmayalım.

Kod kısmımız bu kadar ancak bazı küçük geliştirmeler yapalım. Mesela ses dosyası oluşturulduktan sonra dosyayı açmak için çift tıklamamız gerekti. Bunun yerine çevrim tamamlandığında otomatik yürütülmesini istersek ne yapacağız?

Bunun için os modülünden faydalanabiliriz. Dosya kapatma işlemini yapmadan önce mp3 dosyamızın adını yazarak çevrim tamamlandığında otomatik yürütme sağlanabilir.

Şimdi sistemimizi yormamak adına küçük bir iyileştirme daha yapalım. Kodlar bu şekilde kaldığı zaman klasörümüzün içinde daha önce bu mp3 dosyasının olup olmadığını kontrol etmiyor ve böylelikle metinden sese çevirme işlemi en baştan sürekli tek tek tekrar yapılıyor. Bunun yerine daha önce çevirme işleminin olup olmadığını kontrol eden bir karar mekanizması oluşturalım.

Atadığımız bu def ile dosya yolunu kontrol ediyoruz. Daha önce böyle bir dosya var mı? Varsa bu fonksiyon True döndürecek, yoksa False.

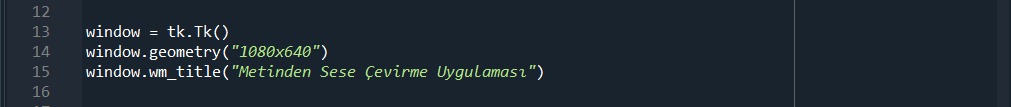
Yazdığımız karar bloğu ile eğer daha önce dosya var ise direkt oku, daha önce böyle bir dosya kaydedilmemişse önce oluştur sonra oku diyoruz.

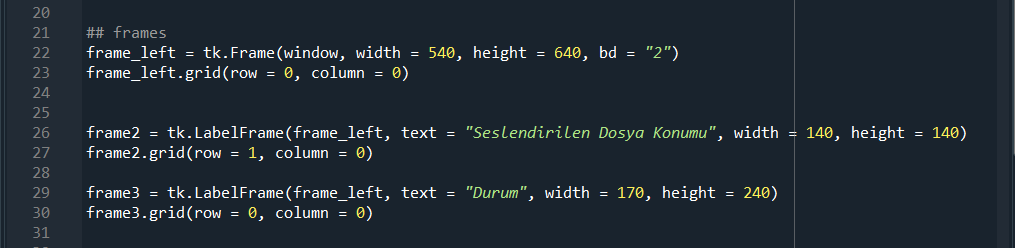
Bu aşamaya kadar hep tek bir masal üzerinden ele aldık. Peki ileride farklı masallar ya da metinler kullanmak istediğimizde kolaylık olması açısından ne yapabiliriz? Sürekli değişken isimlerini değiştirmek zorunda mıyız? Tabi ki hayır 😊 Öyleyse nasıl yapılacağına birlikte bakalım.

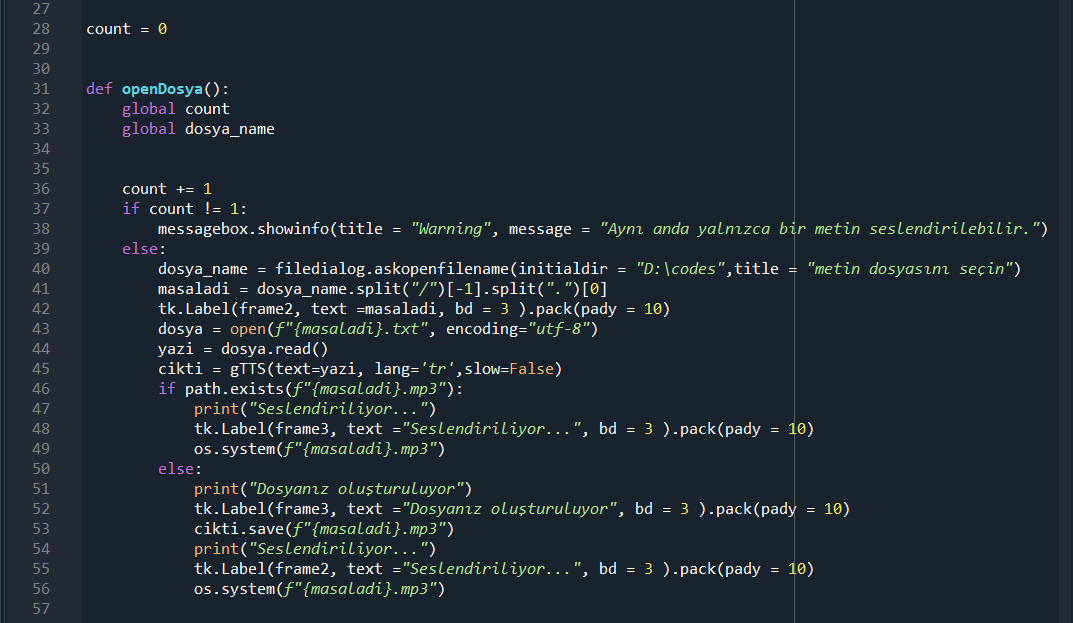
Bunun için yukarıda da görüldüğü gibi “masaladi” şeklinde bir değişken atıyorum. Aşağıda yazdığım kodları da buna göre düzenlemem gerekecek.

Görüldüğü gibi az önce oluşturduğumuz “masaladi” değişkeninindosya isimleri olarak değiştirdik. Ancak süslü parantez kullanıp başına “f” yazmam gerekiyor ki yazdığımız kod format text’i görebilsin.

* **Şimdi yazdığımız program için Python tkinter ile api tasarımını yapalım.**

Kütüphaneleri import ettikten sonra ara yüz tasarımına geçebiliriz.

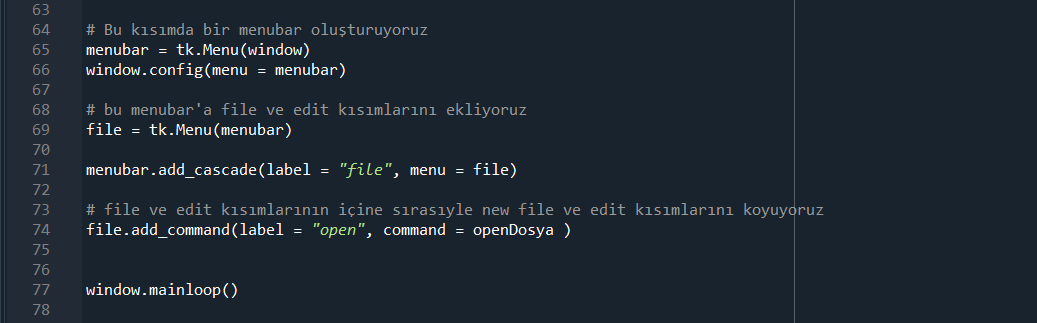
Pencere boyutlarını ve açılacak pencerenin ismini yukarıdaki kod satırları ile ayarlayabiliriz.

Tasarlamak istediğimiz ara yüz için bir adet dosya seçimi bölümü, bir adet dosyanın seslendiriliyor durumu, bir adet de seslendirilen dosyanın konumunu gösteren frame tanımlıyoruz.

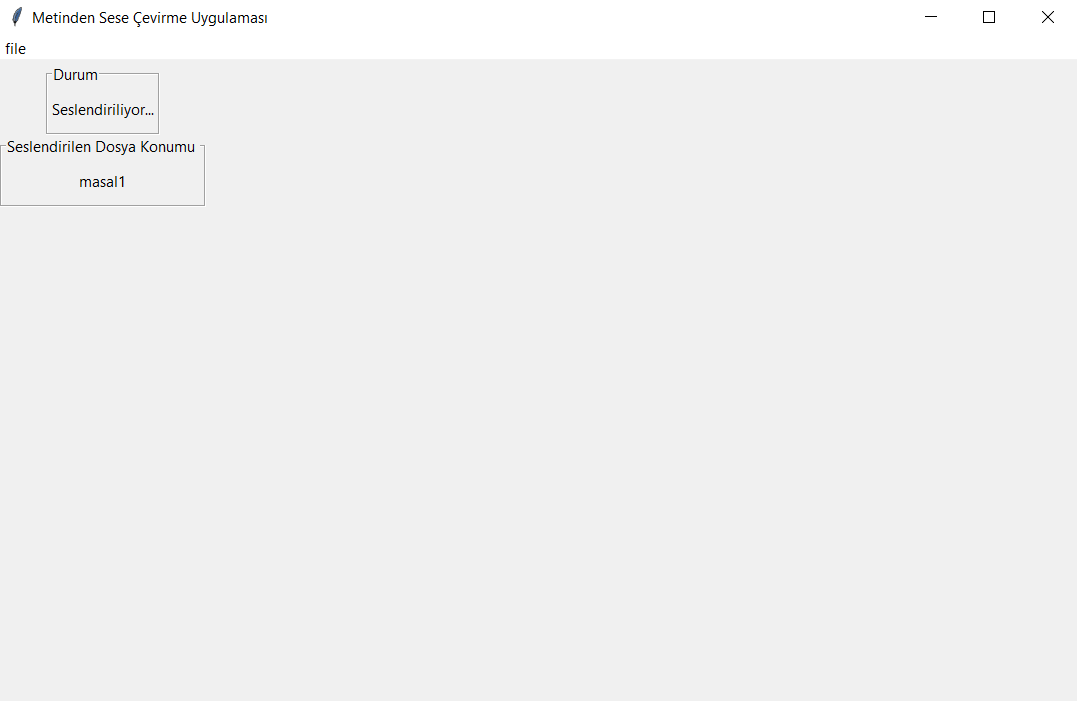
Öncelikle sayaç olarak kullanacağımız “count” değişkenini tanımlıyoruz. Sonrasında bir def atayarak count ve dosya\_name değişkenlerini global değişken olarak ayarlıyoruz.

Kullanıcının aynı anda birden fazla metni okutmak istemesi halinde uyarı verecek kod satırı 38. kod satırıdır.

Sonra sırasıyla ara yüz tasarımından önce yazdığımız kodu, bu kısma framelerin içerisine entegre ediyoruz.



Frame kısımlarını hallettiğimize göre şimdi de üst çubukta yer alacak olan menü kısmına dosyalarımızı seçebilmek için “file” seçeneği ekliyoruz. Bu adımın tamamlanmasının ardından uygulamamız artık çalışmaya hazır durumda olacaktır.

Böylelikle file sekmesinden seçilecek olan .txt formatındaki metin dosyamız .mp3 formatında oynatılacak.

Ara yüz tasarımı bu şekildedir. Sol üstteki file sekmesinden .txt dosyası seçildiği takdirde durumunu ve çalınan dosyanın ismini görüntüleyebiliriz.

Bu yazıda doğal dil işleme, uygulama alanlarını, API (Uygulama Programlama Arayüzü) işledim ve Python dilini kullanarak “Türkçe metinleri nasıl Türkçe olarak seslendiririz?” yani metni sese çevirme (text to speech) anlattım. Yazımı okuduğunuz için teşekkür ederim.