İzmir,
Pescara
2026

# Açık Kaynak Kodlu Simülatör Yazılımı iVEBSIM+

**PROGRAMLAMA DİLİ**

**SİMÜLATÖR YAZILIMI**

**YAPAY ZEKA EKLENTİSİ**

**SİMÜLATÖR KULLANIMI**

**TEMRİNLER**

## KA220VET Erasmus+ Projesi

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY

Avrupa Birliği tarafından finanse edilmektedir

“Erasmus+ Programı kapsamında Avrupa Komisyonu tarafından desteklenmektedir. Burada yer alan içerik yazarların görüşlerini yansıtmaktadır ve bu görüşlerden Avrupa Komisyonu ve Türkiye Ulusal Ajansı sorumlu tutulamaz.”


Avrupa Birliği tarafından finanse edilmektedir

**İÇİNDEKİLER**

# ÖNSÖZ

Eğitim sistemleri, dijital dönüşümün etkisiyle yeniden şekillenmekte; özellikle mesleki ve teknik eğitim alanında yenilikçi, esnek ve erişilebilir öğrenme ortamlarına duyulan ihtiyaç her geçen gün artmaktadır. Erasmus+ KA220-VET iş birliği ortaklıkları kapsamında yürütülen "Mesleki Eğitimin Yapay Zeka Destekli Açık Kaynak Kodlu Simülatör ile Geliştirilmesi" başlıklı bu proje, söz konusu ihtiyaca sürdürülebilir ve yenilikçi bir yanıt üretme amacı taşımaktadır.

Türkiye koordinatörlüğünde; Polonya, İspanya, İtalya, Portekiz ve Romanya ortaklığında hayata geçirilen bu çok uluslu çalışma, mesleki eğitimin dijital araçlarla güçlendirilmesini, öğretim süreçlerinin daha etkileşimli ve uygulama temelli hâle getirilmesini hedeflemektedir. Projenin temel çıktısı olan iVEBSIM+ simülatörü, endüstriyel elektronik ve kontrol alanına yönelik olarak tasarlanmış, tamamen açık kaynak kodlu, yapay zekâ destekli bir yazılımdır. Bu yönüyle proje, yalnızca bir eğitim materyali üretmekle kalmayıp; aynı zamanda paylaşılabilir, geliştirilebilir ve sürdürülebilir bir dijital öğrenme ekosistemi oluşturmayı amaçlamaktadır.

Elinizde bulunan bu kitap, projenin çıktılarından biri olarak kurgulanmıştır. Kitabın ilk bölümünde, simülatör yazılımının geliştirilmesinde temel programlama dili olarak kullanılan Python'a ilişkin kuramsal ve uygulamalı bilgiler sunulmuştur. Programlama temelleri, değişken yapıları, karar ve döngü mekanizmaları gibi konular; mesleki eğitim öğrencilerinin anlayabileceği açıklıkta ve uygulama odaklı bir yaklaşımla ele alınmıştır.

Devam eden bölümlerde;

Simülatör yazılımının kuramsal arka planı,

Grafik kullanıcı arayüzünün oluşturulması,

Endüstriyel elektronik bileşenlerin dijital ortama entegrasyonu,

Yapay zekâ eklentisinin geliştirilmesi ve kullanımı,

Mesleki eğitimde kullanılmak üzere hazırlanan uygulama temrinleri,

Öğrencilerle gerçekleştirilen pilot uygulama süreçleri

ayrıntılı biçimde ele alınmaktadır.

Proje süreci, her bir hareketlilikte belirli bir teknik aşamanın tamamlanması esasına dayalı olarak yapılandırılmıştır. Python temellerinin oluşturulmasından simülatör çekirdeğinin geliştirilmesine; endüstriyel komponent kütüphanesinin genişletilmesinden yapay zekâ entegrasyonuna ve pedagojik temrinlerin hazırlanmasına kadar tüm aşamalar uluslararası iş birliği içerisinde gerçekleştirilmiştir.

Böylece hem teknik kapasite geliştirilmiş hem de iyi uygulama örneklerinin ülkeler arası paylaşımı sağlanmıştır.

Bu çalışmanın en önemli özelliklerinden biri, geliştirilen simülatörün açık kaynak kodlu olmasıdır. iVEBSIM+ Simülatörü'nün tüm kaynak kodları proje web sitesi aracılığıyla kamuoyu ile paylaşılmakta; eğitimcilerin, öğrencilerin ve geliştiricilerin katkısına açık bir yapı sunulmaktadır. Bu yaklaşım; şeffaflık, erişilebilirlik ve kolektif üretim ilkelerini temel almakta; mesleki eğitimde dijital eşitliği desteklemektedir. Ayrıca kitap içeriğinde yazılımın geliştirme aşamalarının ayrıntılı biçimde paylaşılması, öğrenenlerin yalnızca kullanıcı değil aynı zamanda geliştirici bireyler olmalarını teşvik etmektedir.

Mesleki eğitimde simülasyon temelli öğrenme; güvenli deneyim alanı sunması, maliyetleri azaltması ve tekrar edilebilir uygulama imkânı sağlaması bakımından büyük önem taşımaktadır. Yapay zekâ desteği ise bireyselleştirilmiş geri bildirim, hata analizi ve akıllı yönlendirme gibi olanaklarla öğrenme sürecini daha verimli ve etkileşimli hâle getirmektedir. Bu kitap ve beraberindeki yazılım, söz konusu iki yaklaşımı bütünleştirerek çağın gerektirdiği dijital yeterliliklerin kazandırılmasına katkı sunmayı hedeflemektedir.

Bu eserin; mesleki ve teknik eğitim kurumlarında görev yapan öğretmenlere, alanda öğrenim gören öğrencilere, yazılım geliştiricilere ve araştırmacılara yol gösterici olması temennisiyle hazırlanmıştır. Uluslararası ortaklığın bilgi birikimi ve deneyimiyle ortaya çıkan bu çalışmanın, açık bilim ve açık eğitim kaynakları anlayışı doğrultusunda yaygınlaşması en büyük dileğimizdir.

Bu proje, Avrupa Birliği tarafından desteklenmiş olup içerikte yer alan görüşler yazarlarına aittir.

Katkı sunan tüm proje ortaklarına ve emeği geçen eğitimcilere teşekkür ederiz.

İTO Vakfı Süleyman Taştekin MTAL

Zespol Szkol Nr1 im. prof. B. Krupinski ego

I. I. S. A. VOLTA

Liceul Tehnologic "Anghel Saligny"

Instituto Nossa Senhora Da Encarnacao Cooperativa De Ensino E Cultura CRL

Asocicion de profesores Tecnicos y de Ensenanza Secundar ia de Automocion de Galicia

"Erasmus+ Programı kapsamında Avrupa Komisyonu tarafından desteklenmektedir. Burada yer alan içerik yazarların görüşlerini yansıtmaktadır ve bu görüşlerden Avrupa Komisyonu ve Türkiye Ulusal Ajansı sorumlu tutulamaz."

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY

Avrupa Birliği tarafından finanse edilmektedir

# PYTHON PROGRAMLAMA DİLİ

## 1. PYTHON PROGRAMLAMA DİLİ VE ÖNEMİ

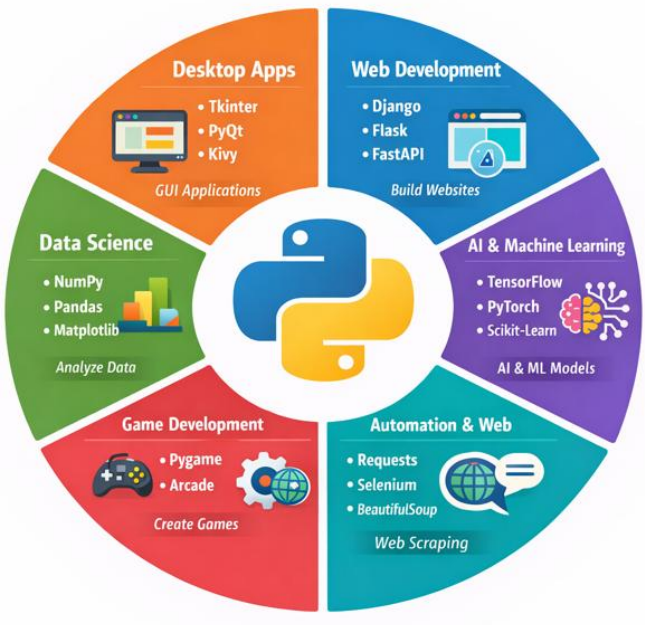
Python, günümüzde en çok kullanılan programlama dillerinden biridir. Guido van Rossum tarafından geliştirimiş ve 1991 yılında yayınlanmıştır. Yüksek seviyeli bir dil olup hemen hemen her alanda uygulama geliştirebileceğiniz, geniş bir kütüphane desteğine sahiptir.

Basit söz dizimine sahip olması, dilin öğrenilmesini kolaylaştırmaktadır. Söz Dizimi diğer diller ile karşılaştırıldığında; İngilizce diline benzemesi, noktalı virgül ve parantezler yerine satırı tamamlamak için alt satıra geçmesi, yazım sırasında gerektiğinde girintiler verilerek döngülerin, fonksiyonların, sınıfların kapsama alanı belirlenmektedir. Bu nedenle programlamaya yeni başlayanlar tarafından da tercih edilmektedir.

Unix, Linux, Mac, Windows, .. vb. her türlü platformda çalışabilir. Kurulumu kolay ve ücretsizdir.

Python, çeşitli uygulamalarda kullanılmaktadır. Uygulama geliştirebileceğiniz alanlardan bazıları aşağıdaki gibidir.

- Sistem programlama
- Kullanıcı arabirimi programlama
- Web geliştirme
- Ağ programlama
- Oyun geliştirme
- Veri Bilimi Makine öğrenimi
- Veri analizi
- Yapay Zeka (AI)
- Komut dosyası oluşturma ve araçlar




Şekil 1. Python Kütüphaneleri

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Avrupa Birliği tarafından finanse edilmektedir

## 2. PYTHON PROGRAMLAMA DİLİNİN YÜKLENMESİ

Python programlama dili ile kod yazabilmek için bilgisayarınıza kodları çalışabileceğiniz bir editor yüklemeniz gerekmektedir.

### 2.1 Python Yüklenmesi

Windows için:

- Python'ın resmi web sitesine gidiniz. ( https://www.python.org/)
- Downloads menüsüne tıklayınız. Downloads to Windows butonuna tıklayarak en son sürümünü indiriniz. ( Python 3.14.0)
- Kurulum dosyasını çalıştırınız. Add python.exe to PATH seçeneğini seçerek Install Now butonuna tıklayınız.


Şekil 2. Kurulum Penceresi

- Kurulum tamamlanınca "Setup was successful" mesajını aldığınızda "Close" butonuna tıklayınız.

macOS için :

- .pkg dosyasını aç, klasik macOS kurulum adımlarını takip et.
- Python, Applications → Python x.x klasörüne yüklenir.

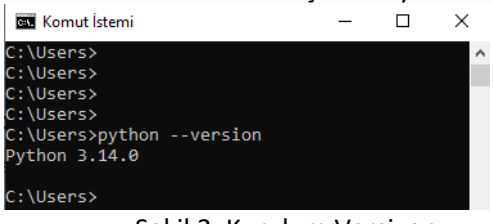
Linux için:

Terminal aç →

- sudo apt update
- sudo apt install python3 python3-pip

### 2.2 Kurulumun Doğrulanması

Microsoft Windows işletim sisteminde Komut İstemini açınız ve yandaki kodu yazınız. python --version



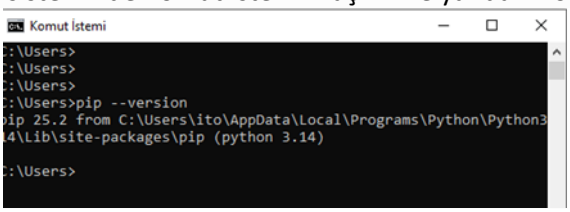

Şekil 3. Kurulum Versiyonu

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Avrupa Birliği tarafından finanse edilmektedir

Şekil 3'teki gibi python sürüm numarası görünüyorsa doğru bir şekilde kurulmuştur. Göremediyseniz, kurulum aşamalarını yenileyiniz.

### 2.3 Paket Yöneticisinin Kontrolü

Microsoft Windows işletim sisteminde Komut İstemini açınız ve yandaki kodu yazınız. pip --version

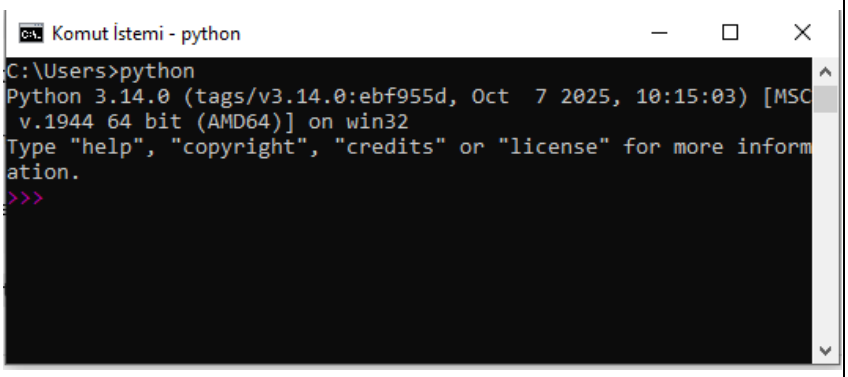
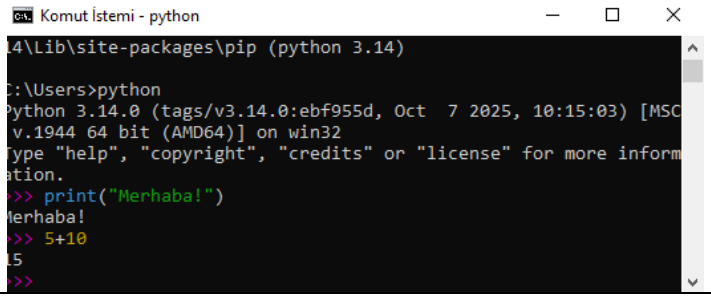



Şekil 4. Paket Yöneticisi Versiyonu

Pip (Python Package Installer) Python Paket Yükleyici, Python dilinde kütüphane yönetimi ve paket kurulumu için kullanılan bir araçtır. Python kütüphanelerini kurmak, güncellemek ve yönetmek için kullanılır. Şekil 4'teki gibi pip sürümü görünüyorsa doğru bir şekilde kurulmuştur.

### 2.4 İlk Python Programının Çalıştırılması

Python'da komut kodlarını yazmak için bir editöre ihtiyaç duyulmaz. Komut İstemci kullanılarak direkt olarak kodlar çalıştırılabilir. Tablo 1.'de ekrana "Merhaba" yazan kod parçacığının örneği verilmiştir. Program kodları çoğaldıkça, kütüphane kullanımları devreye girdikçe düzen ve erişilebilirlik bakımından bu yöntem bir programcı için pek geçerli olmayabilir.

| **Kodlar:** | Ekran çıktılarınız aşağıdaki gibi olacaktır. |
|---|---|
| Pyhton |  |
| **Kodlar:** | Ekran çıktılarınız aşağıdaki gibi olacaktır. |
| print("Merhaba!")<br>5+10 |  |

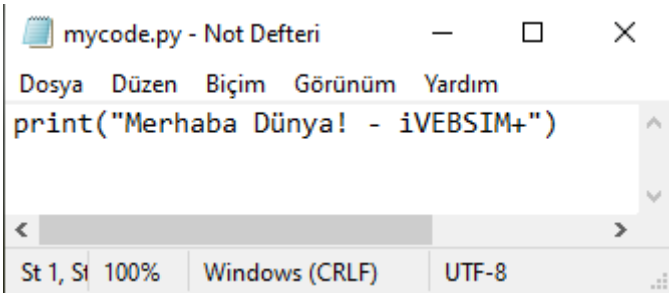




Tablo 1. Komut İstemi İle Program Yazılması

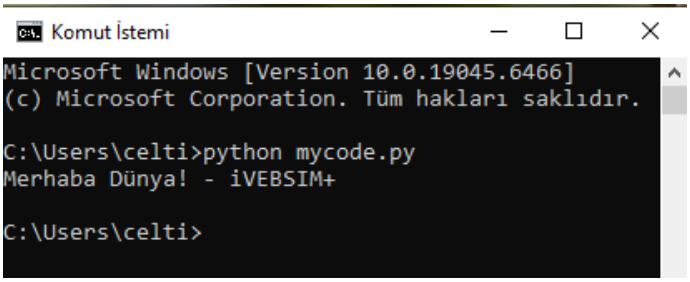
## 3. EDİTÖR KULLANIMI

Eğer istenirse tercih edilen bir metin editöründe kod parçacıkları yazılarak uzantısı .py olan bir dosya oluşturulabilir. Şekli 5.'te mycode.py adında yeni bir dosya oluşturuldu.




Şekil 5. Metin Editör Kullanımı

Komut İstemcide python mycode.py komutu yazıldığında program çalışacaktır ve konsolumuzda aşağıdaki şekilde gözükecektir.



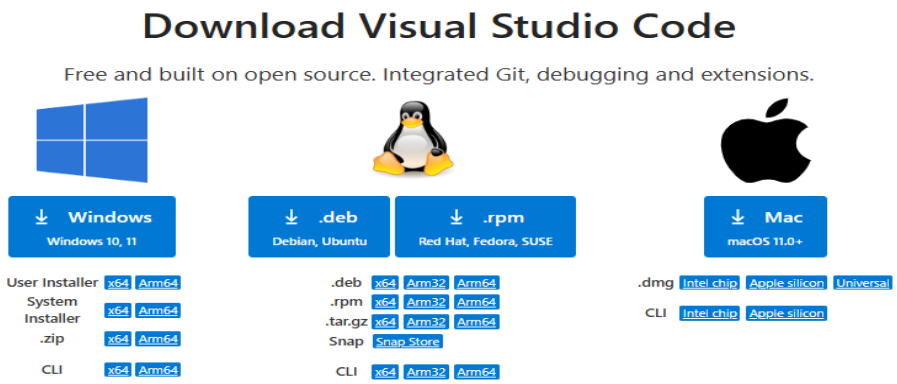

Şekil 6. Konsol Program Çıktısı

Kodlarımızı daha düzenli görmek ve daha Rahat yazabilmek için editörlerden de yararlanabiliriz. Visual Studio Code, Phycharm, Sublime Text bunlardan bazılarıdır.

### 3.1. Visual Studio Code Kurulumu

Visual Studio (VS) Code, macOS, Linux ve Windows işletim sistemlerinde çalışan ücretsiz bir kod düzenleyicisidir. VS Code'un küçük bir indirme dosyası olduğundan, birkaç dakika içinde kurulur ve hemen hemen tüm donanımlarda sorunsuz çalışmaktadır.

Visual Studio Code kurulumu için https://code.visualstudio.com/Download adresinden ilgili yönergeleri izleyerek kurabilirsiniz.

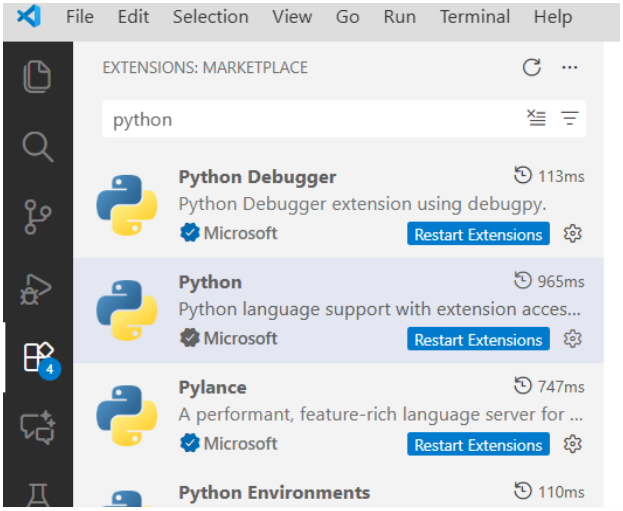



Şekil 7. VS Code İndirme Sayfası

### 3.2. Visual Studio Code Python Eklentisinin Kurulumu

VS Code editöründe Python kodlarını yazarken kod tamamlama, hata gösterimi, Hata ayıklama ve çalıştırma işlemlerini yapabilmek için Microsoft tarafından yayınlanan Python eklentisini kurmamız gerekiyor. Bunun için:

1. Wiew menüsünde Extensions seçiniz veya sol açılır menü kısmından  Extensions butonuna tıklayınız.

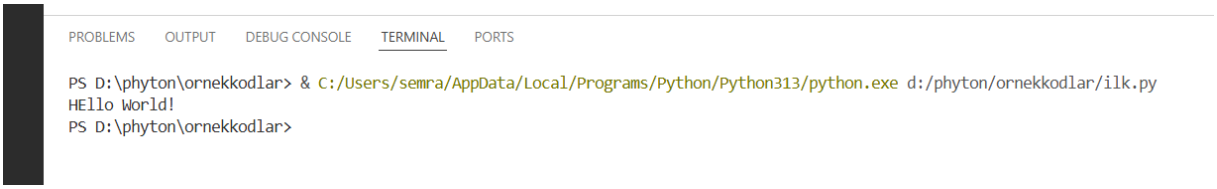



Şekil 8. VS Code Python Kurulumu

2. Arama kutusuna Python yazıp, Microsoft tarafından yayınlanan eklentiyi kurunuz.

### 3.3. İlk Kodun Çalıştırılması

- File → New File → Açılan pencereden python File'ı seçiniz
- Editör içerisine ; print("HEllo World!") kodlarını yazınız.
- Run Python File komutunu vererek kodlarınızı çalıştırınız.




Şekil 9. Ekran Çıktısı

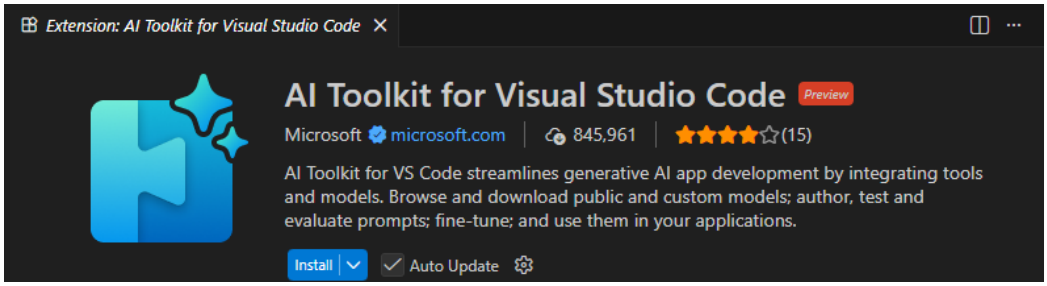
### 3.4. Visual Studio Code Yapay Zeka Araç Seti

VS Code için Yapay Zeka Araç Seti (AI Toolkit), programcıların yapay zeka modellerini kullanarak akıllı uygulamalar oluşturmasına, test etmesine ve geliştirmesine imkan sağlayan bir eklentidir. AI Toolkit, OpenAI, Anthropic, Google ve GitHub gibi popüler yapay zeka modelleriyle sorunsuz entegrasyon sunar.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Avrupa Birliği tarafından finanse edilmektedir

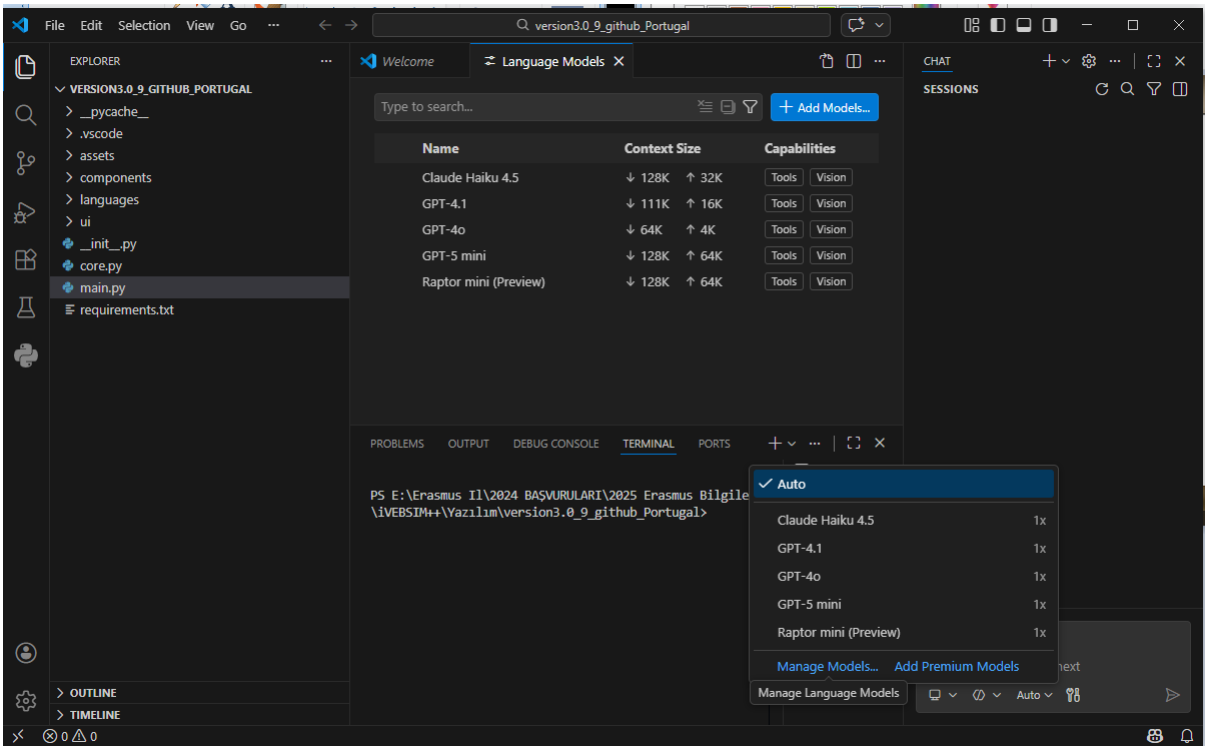
### 3.4.1. Kurulum

Wiew menüsünde Extensions seçiniz veya sol açılır menü kısmından Extensions butonuna tıklayınız. Arama kutusuna AI Toolkit for Visual Studio Code yazıp, Microsoft tarafından yayınlanan eklentiyi kurunuz.




Şekil 10. AI Toolkit Kurulumu

Şekil 11.'de VS Code Editörünün genel yapısı gözükmektedir. Sağ alt köşede bulunan Auto menüsü açıldığında editördeki Yapay Zeka Modelleri gözükmekte ve bu menüden yapay zeka modellerinin yönetimi yapılmaktadır. Auto seçeneği seçildiğinde tüm modeller devreye alınır ve kullanılır.

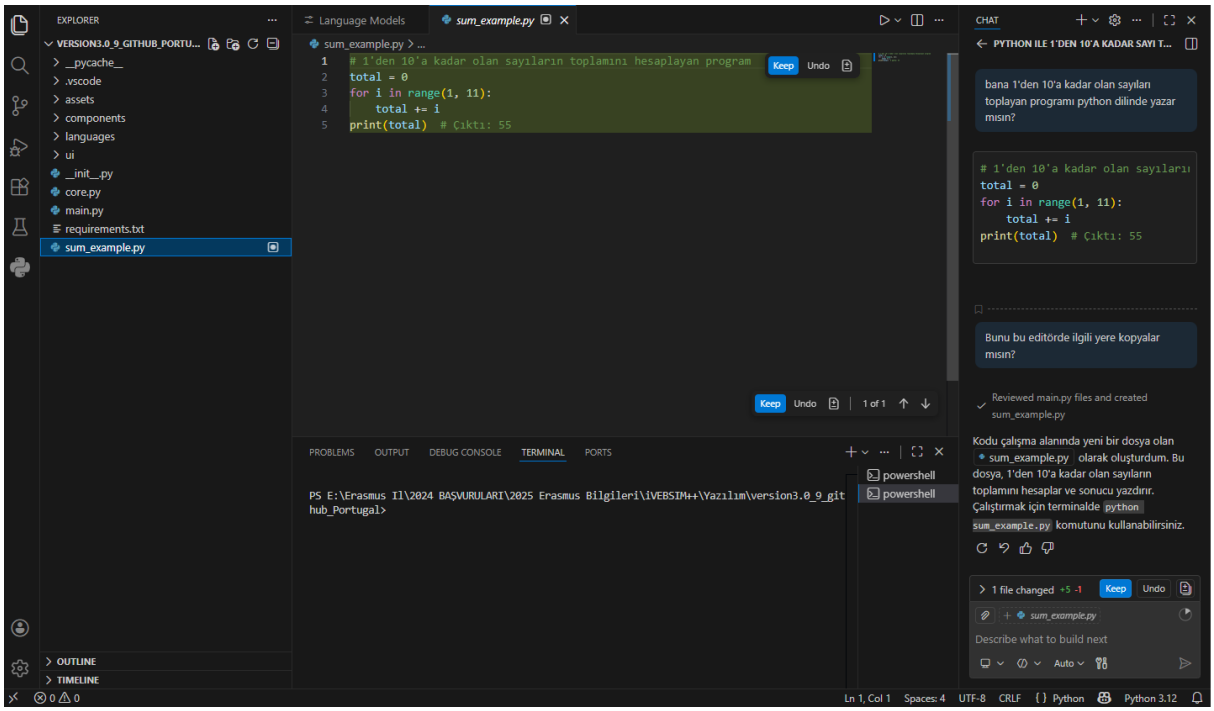



Şekil 11. VS Code Yapay Zeka Destek Yönetimi

### 3.4.2. Çalıştırılması

Şekil 11.'de Sağ alt köşede bulunan menüde Yapay Zekaya 1'den 10'a kadar olan sayıları toplayan bir program yazması istenmiştir. Bunu gerçekleştirildikten sonra Yapay Zekaya bu programı Python dilinde Editörde yazması istenmiştir. Yapay Zeka istenilenleri gerçekleştirerek sum_example.py isimli bir

pytohn dosyası açmış, programı buraya kaydetmiş, çalıştırmış ve sonucu ekranda yazmıştır. Bu projede bu gibi yapay zeka kullanımı oldukça fazla kullanılmıştır.




Şekil 12. VS Code Yapay Zeka Kullanımı

## 4.PYTHON DEĞİŞKENLERİ VE DEĞİŞKEN KURALLARI

Python'da değişken tanımlamak için bir komutu yoktur. Değişken adını bildirip, içerisine değer atamanız yeterlidir. İçerisine aktarılan değere göre değişken türü belirlenir.

Fakat değişken adı verirken dikkat etmemiz gereken bazı kurallar vardır. Bunlar:

a) Değişken isimleri büyük küçük harf duyarlıdır.
Örneğin; değişken isminin **adres** ya da **Adres** olması bu değişkenlerin farklı iki değişken olduğunu gösterir.

b) Değişken isimlendirilirken hem harfler hem de sayılar kullanılabilir fakat sayılar başa gelemez. Örneğin sayi1 doğru bir isimlendirmeyken **1sayi** doğru bir isimlendirme değildir.

c) Değişken isimlendirilirken alt tire (_) kullanılabilir. Ancak boşluk ve diğer özel karakterler (?,%,!, ., + vb.) kullanılamaz.
Örneğin; **ev_adresi** doğru bir isimlendirmeyken **ev adresi, ev-adresi** ya da **ev%adresi** gibi değişken isimleri kurallara aykırı olduğundan hataya neden olacaktır.

d) Değişken isimlendirilirken phyton programlama dilindeki komutları örneğin if, for, true vb. Ifadeleri kullanılmamalıdır.

Bu kurallar çerçevesinde aşağıda doğru tanımlanmış bazı değişken örnekleri görülmektedir:

yasadigi_sehir =“Izmir”

sinav_notu =80

oran3=5.7

Yukarıda tanımlanan; yasadigi_sehir, sinav_notu, oran3 birer değişkendirler ve kendilerine birer değer atanmıştır.

### 4.1. Python Operatörleri

Operatörler, değişkenler ve veriler üzerinde işlem yaparak yeni değerler üretilmesini sağlayan sembollerdir. Python programlama diline yeni başlayanlar için aritmetiksel, atama, karşılaştırma, mantıksal ve kimlik operatörleri öğrenmek son derece önemlidir.

#### 4.1.1.Aritmetik Operatörler

Değişkenler içerisinde saklanılan değerler üzerinde matematiksel işlem yapmak için kullanılan operatörlerdir.

| Operatör | Tanımı | Örnek |
|---|---|---|
| + | Toplama | a+b |
| - | Çıkarma | a-b |
| * | Çarpma | a*b |
| / | Bölme | a/b |
| % | Mod alma (Bir sayının diğer sayıya bölümünden kalan) | a%b |
| ** | Kuvvet alma (ab) | a**b |
| // | Tam sayı bölme (Bölme işleminde sadece tam kısım alınır.) | a// |

| Örnek Kodlar | Ekran Çıktısı |
|---|---|
| a = 10<br>b = 5<br>print(a)<br>print(b) | 10<br>5 |
| sonuc = a + b<br>print("Sonuç:", sonuc) | Sonuç: 15 |
| sonuc = a - b<br>print("Sonuç:", sonuc) | Sonuç: 5 |
| sonuc = a * b<br>print("Sonuç:", sonuc) | Sonuç: 50 |
| sonuc = a / b<br>print("Sonuç:", sonuc) | Sonuç: 2.0 |
| sonuc = a % b<br>print("Sonuç:", sonuc) | Sonuç: 0 |
| sonuc = a ** b<br>print("Sonuç:", sonuc) | Sonuç: 100000 |
| sonuc = a // b<br>print("Sonuç:", sonuc) | Sonuç: 2 |

### 4.1.2. Atama Operatörleri

Bir değişkendeki değerin başka bir değişkene aktarılabilmesi için ya da bir işlem sonucunun başka bir değişkene aktarılabilmesi için kullanılan operatörlerdir.

| Örnek Kodlar | Ekran Çıktısı |
|---|---|
| x = 10<br>print(x) | 10 |
| x = 10<br>x += 5  print(x) | 15 |
| x = 10<br>x -= 3<br>print(x) | 7 |
| x = 4<br>x *= 3<br>print(x) | 12 |
| x = 10<br>x /= 4<br>print(x) | 2.5 |
| x = 10<br>x //= 4<br>print(x) | 2 |
| x = 10<br>x %= 3<br>print(x) | 1 |
| x = 2<br>x **= 3<br>print(x) | 8 |

| Operatör | Örnek | Açıklama |
|---|---|---|
| = | a=2 | a değişkenine 2 değeri atanmıştır. |
| += | a+=2 | a değişkenine 2 değerini ekleyerek yine a değişkenine atanmıştır. a=a+2 anlamına gelmektedir. |
| -= | a-=2 | a değişkeninden 2 değeri çıkarılarak yine a değişkenine atanmıştır. a=a-2 anlamına gelmektedir. |
| *= | a*=2 | a değişkeni 2 ile çarpılarak yine a değişkenine atanmıştır. a=a*2 anlamına gelmektedir. |
| /= | a/=2 | a değişkeni 2 değerine bölünerek yine a değişkenine atanmıştır. a=a/2 anlamına gelmektedir. |
| %= | a%=2 | a değişkenin 2 değeri ile modu alınarak yine a değişkenine atanmıştır. a=a%2 anlamına gelmektedir. |

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Avrupa Birliği tarafından finanse edilmektedir

### 4.1.3. Karşılaştırma Operatörleri

İki değişkenin değerini karşılaştırıp ona göre işlem yaptırmak istendiğinde kullanılan operatörlerdir.

| Operatör | Tanımı | Örnek |
|---|---|---|
| == | Eşittir | a==b |
| != | Eşit değildir | a!=b |
| < | Küçüktür | a<b |
| > | Büyüktür | a>b |
| <= | Küçük eşittir | a<=b |
| >= | Büyük eşittir | a>= |

| Örnek Kodlar | Ekran Çıktısı |
|---|---|
| a = 10<br>b = 10<br>print(a == b) | True |
| a = 10<br>b = 5<br>print(a != b) | True |
| a = 7<br>b = 10<br>print(a > b) | False |
| a = 5<br>b = 8<br>print(a < b) | True |
| a = 10<br>b = 10<br>print(a >= b) | True |
| a = 12<br>b = 10<br>print(a <= b) | False |

### 4.1.4. Mantıksal Operatörler

İki ya da daha fazla değişkenin değerini kontrol ettirerek program akışına karar vermek için kullanılan operatörlerdir.

| Operatör | Örnek | Açıklama |
|---|---|---|
| and | a<3 and b>=5 | İki veya daha fazla şartın tamamının doğru olması durumunda True değerini döndürür. Buradaki örnekte a değişkeni 3'ten küçük ve b değişkeni 5'e eşit ya da 5'ten büyük olursa True değeri döndürülür. |
| or | a<3 or b>4 | İki veya daha fazla şartın en az birinin doğru olması durumunda True değerini döndürür. Buradaki örnekte a değişkeninin 3'ten küçük olması ya da b değişkenin 4'ten büyük olması True değeri döndürmek için yeterlidir. |
| not | not(a<3) | Durumu tersine çevirmek (True ise False; False ise True) için kullanılır. Buradaki örnekte parantez içindeki mantıksal sınamanın sonucu tersine çevrilir. İfadenin not komutu olmadan yazıldığında true döndüreceği varsayıldığında bu haliyle false döndürecektir. |

| Örnek Kodlar | Ekran Çıktısı |
|---|---|
| a = 10<br>b = 5<br>print(a<3 and b>=5) | False |
| a = 10<br>b = 5<br>print(a<3 and b>=5) | True |
| durum = True<br>print(not durum) | False |

### 4.2. Python Veri Türleri

Python'da genel olarak string (metinsel), numbers (sayısal),boolean, list (liste), tuple (demet), dictionary (sözlük) ve set (küme) gibi veri tipleri bulunmaktadır. Başlangıç aşamasında bunlardan en fazla kullanacağımız değişken türleri aşağıda yer almaktadır.

#### 4.2.1. String (Metinsel) Veri Tipi

Tek ya da çift tırnak içlerine yazılan karakter dizileridir. Burada karakter harf (t,c), rakam (1,9,2,3) ya da özel semboller (&,/) olabilir. String veri tipleri tek ya da çift tırnak içinde yazılır

```
name="Mert"
surname="Yılmaz"
```

#### 4.2.2. Numbers (Sayısal) Veri Tipi

Sayısal verileri tutan veri tiplerine verilen addır. Python'da sayısal veri tipleri genel olarak int, float ve complex veri tipleridir.

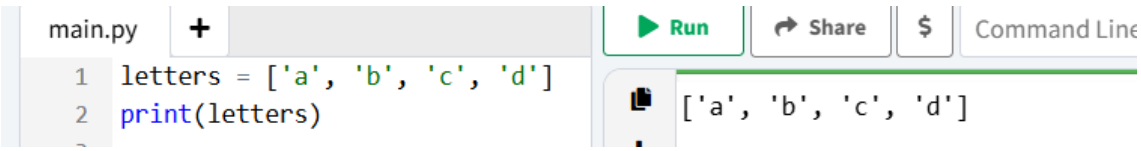
```
sayi=1919
pi_degeri=3.14
```

#### 4.2.3. Python Listeleri

Farklı verilerin bir dizi hâlinde tutulduğu koleksiyonlara liste adı verilir. Python programlama dilinde listeler iki köşeli parantez ile tanımlanmaktadır.

```
sayı=[10,20,30,40,50,60,70]

sehir=["İzmir", "İstanbul", "Ankara", "Bolu"]
```

Örnek Kullanım



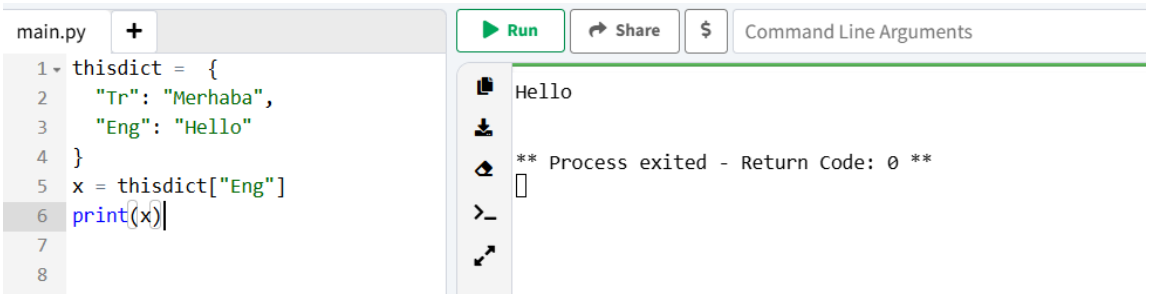

. int, float, string gibi farklı veri tiplerini tek bir listede tutabilirsiniz. Birden fazla veriyi sıralı ve değiştirilebilen bir yapıda tutmak için listeler kullanılır.

```
ilk_liste=["Ankara", 312, 0.6]
```

### 4.2.4 Sözlük (dict)

Anahtar-değer çiftlerini saklar. Anahtarlar benzersiz olmalıdır. {} içinde anahtar:değer şeklinde tanımlanır.

```
class = {
  "Name": "Gizem",
  "SurName": "Yılmaz",
  "Age": 18
}
```




## 5. KARAR YAPILARI

Bir program akışında programın çalışma yönünü belirlemek için gerektiği yerlerde döngülerden ve karar yapılarından faydalanılmaktadır.

### 5.1. Karar Yapısı – IF..ELIF

İki ya da daha fazla değişkenin değerlerini kontrol ettirip kodların istenilen şartlara uygun çalışmasını sağlamak için kullanılır.

- IF YAPISI
  If Kullanımı **:** Kontrol etmemiz gereken tek şart durumlarında kullanılır.

```
if ( Şart ):
    Doğru ise çalışan kodlar
```

Örnek:

```
a = 33
b = 200
if b > a:
  print("b a'dan büyüktür")
```

- IF … ELIF KULLANIMI : Birden fazla şarta göre işlem yaptıracağımız durumlarda kullanılır. Birden fazla elif komutu ile şart tanımlayabilirsiniz.

**Kullanımı**

if( Şart 1):

Şart 1'imiz doğru ise çalışmasını istediğimiz kolardır. Şart 1 doğru değil ise Şart 2'yi kontrol eder.

elif(Şart 2):

Şart 2'imiz doğru ise çalışmasını istediğimiz kolardır.

Örnek:

```
a = 33
b = 33
if b > a:
  print("b a'dan büyüktür")
elif a == b:
  print("a and b eşittir")
```

➢ ELSE KULLANIMI

if …. elif yapısında verilen tüm şartların doğru olmaması durumunda çalışır. Else bloğunda koşul yazılmaz.

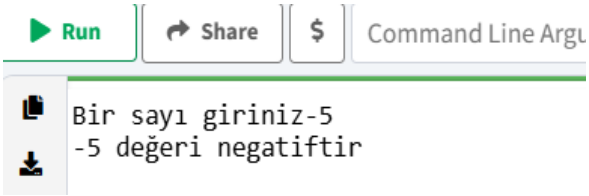
**Örnek**

```
a=int(input("Bir sayı giriniz"))
if(a>0):
    print(str(a)+" değeri pozitiftir")
elif a < 0:
    print (str(a)+" değeri negatiftir")
else:
    print (str(a)+" değeri sıfırdır")
```

Programın Çalıştırılması

Klavyeden bir sayı girilmesi istenmektedir.

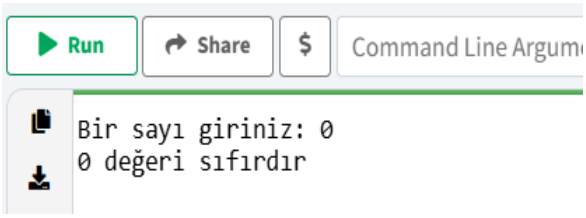
Örneğin a değişkenine -5 değerinin girildiği düşünüldüğünde if bloğunda kontrol etmeye başlar.




- A değişkeninin değeri 0'dan büyük mü? -5>0 'dan büyük olmadığı için diğer şart ifadesine geçer.
- A<0 , A değişkeninin değeri 0'dan küçük mü? Evet, şart doğru olduğu için print("..") komutu çalışır. Ekrana -5 değeri negatiftir diye yazar.

Örneğin a değişkenine 0 değerinin girildiği düşünüldüğünde

- A değişkeninin değeri 0'dan büyük mü? 0>0 'dan büyük olmadığı için diğer şart ifadesine geçer.




- A<0 , A değişkeninin değeri 0'dan küçük mü? Hayır, 0<0 değildir. O yüzden tüm şartlar sağlanmadığı için else bloğuna geçer
- print("..") komutu çalışır. Ekrana "0 değeri sıfırdır" diye yazar.

## 6. DÖNGÜLER

Belli bir şart sağlanana kadar tekrarlanmasını istediğimiz komutları yazarken döngülerden yararlanırız. Döngüler, kodlarımızın daha sade, anlaşılır ve kısa olmasını sağlar. İçerisinde birden fazla değer bulunduran yapılarla ( Dizi,liste, sözlükler ) daha kolay çalışmamızı sağlar. For Döngüsü, While Döngüsü ve Do While döngü yapılarını kullanırız.

### 6.1. For Döngüsü

Bir for döngüsü, bir dizi (yani bir liste, bir tanımlama grubu, bir sözlük, bir küme veya bir dize) üzerinde yineleme yapmak için kullanılır. For Döngüsü genellikle, çalışacak olan komutların, tekrar sayılarının belli olduğu durumlarda kullanılır.

**For Döngüsünün Range Fonksiyonu ile birlikte kullanımı**

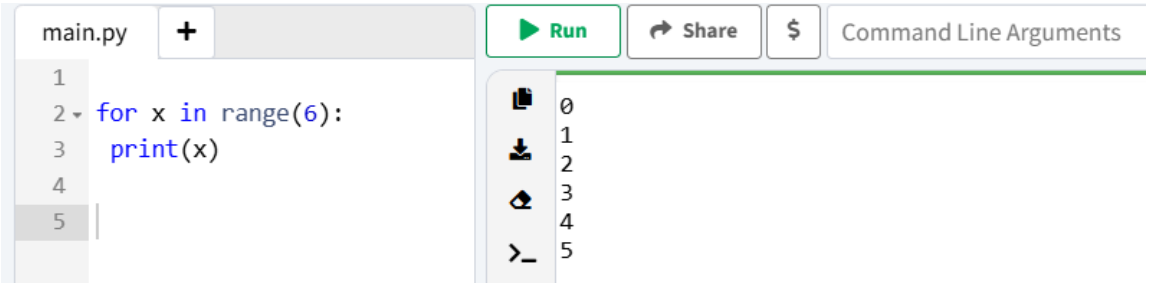
Döngüyü belirtilen sayıda döndürebilmek için Range fonksiyonundan yararlanılır.

**Kullanımı:**

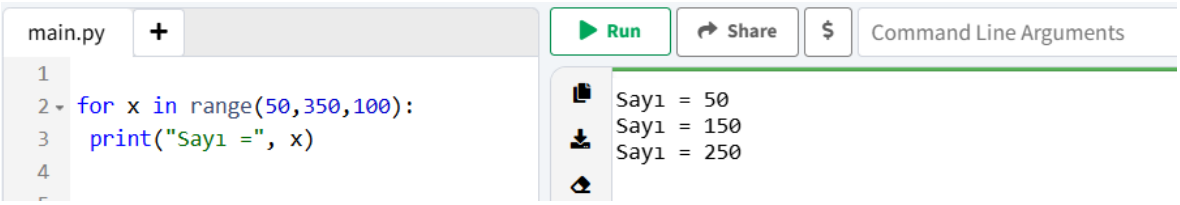
Range(Başlangıç Değeri, Bitiş Değeri, Artış Miktarı) şeklinde tanımlanır.

Range(sayi) şeklinde tanımlandığında varsayılan olarak 0'dan başlar, belirtilen sayıya kadar birer birer artarak ilerlememizi sağlar.

**Örnek 1:** Aşağıdaki kod ekranında 1'den 6'ya kadarki sayıların ekrana yazdırılması sağlanmıştır.



**Örnek 2:** Aşağıdaki kod ekranında, 50'den 350 değerine kadar 100'er artırarak yazdırılması sağlanmıştır.




Adım Adım çalıştırırsak;

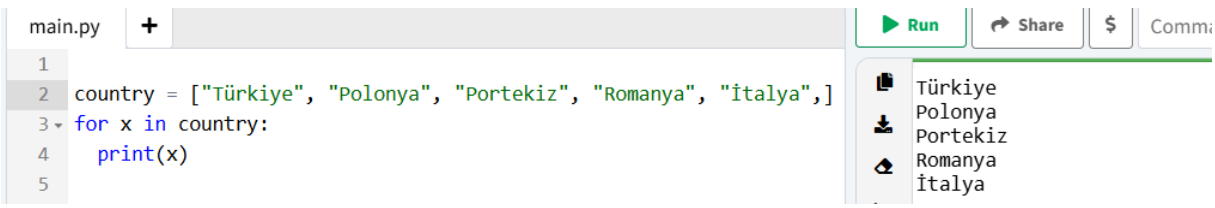
- X sayaç değişkenine 50 değeri aktarılır, x değişkeninin değeri kontrol edilir. x<=350 mi? x 350'den küçük mü? Evet şartımızı sağlar. Ve döngü içerisine girer. Ekrana "Sayı =" stringi ile x değişkeninin değerini birleştirir ve Sayı =50 şeklinde yazılmasını sağlar.Döngü başına yeniden döner.
- x sayaç değişkeninin değerini 100 artırır. x değişkeninin değeri 150 olur. x<=350 mi? yani 150<350 mi? Evet şartımız sağlanır. Döngü içerisine girer. Yeniden ekrana yazdırma komutu çalışır x değişkeninin değeri ile sabit metin ifadesini birleştirerek ekrana Sayı =150 şeklinde yazdırmamızı sağlar. Döngü başına yeniden döner.
- x sayaç değişkeninin değerini 100 artırır. x değişkeninin değeri 250 olur. x<350 mi? yani 250<350 mi? Evet şartımız sağlanır. Döngü içerisine girer. Yeniden ekrana yazdırma komutu çalışır x değişkeninin değeri ile sabit metin ifadesini birleştirerek ekrana Sayı =250 şeklinde yazdırmamızı sağlar. Döngü başına yeniden döner.
- x sayaç değişkeninin değerini 100 artırır. x değişkeninin değeri 350 olur. x<350 mi? yani 350<350 mi? Hayır şartımız sağlanmaz, çünkü eşittir. Döngünün dışına çıkarız. Döngünün dışında da çalıştırılacak bir komut olmadığı için programımız sonlanmış olur. Ekran görüntümüzde yukarıdaki gibi olur.

Böylece şartımız sağlanana kadar 3 kere for döngüsü içerisindeki komutları çalıştırmış oluruz. Sonuç olarak For döngüsünün tekrar sayısı başlangıç değeri ve koşula bağlı olarak değişir. Koşul sağlanana kadar döngü içerisindeki komutlar çalışır.

**For Döngüsünün In Komutu ile Kullanımı**

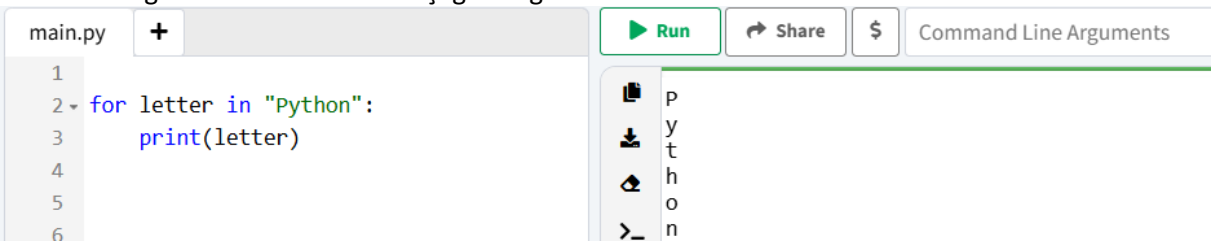
In operatörü belirlenmiş bir değerin listede olup olmadığını kontrol etmek için kullanılır.

Örnek: Tanımlanmış bir ülke dizisini yazdırılması aşağıdaki gibidir.

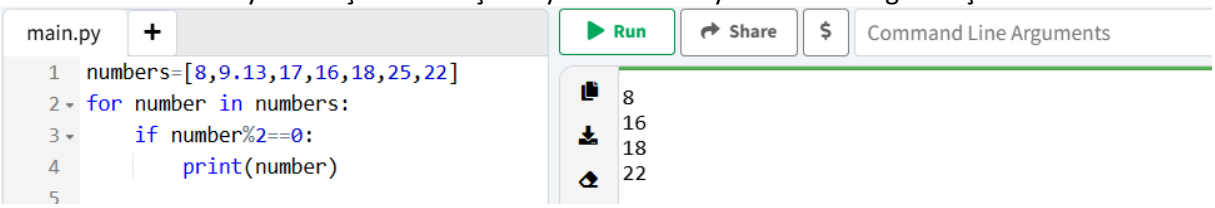



TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Avrupa Birliği tarafından finanse edilmektedir

Örnek: String ifadeler ile kullanımı aşağıdaki gibidir.




Örnek: Verilen bir sayı dizisi içerisinden çift sayıların ekrana yazdırılması sağlanmıştır




### 6.2. While Döngüsü

Verilen koşul doğru olduğu sürece bir dizi ifadeyi çalıştırabiliriz. While döngüsünde tekrar sayısı belli değildir. Verilen şart sağlandığı sürece döngü çalışır. Şart sağlanmadığı anda döngüden çıkar, program döngünün bittiği yerden çalışmaya devam eder.

Kullanımı

```
While(şart)
{
    //İşlemler
}
……
```

Şartınız doğru olduğu sürece parantezler içerisindeki komutların çalıştırılmasını sağlamaktadır.

O yüzden while döngüsünde şartınızı belirlerken çok dikkatli olmanız gerekmektedir.

Örneğin; Aşağıdaki kod ekranında 1’den 4’e kadarki sayıların ekrana yazdırılması sağlanmıştır.


Açıklama:

Sayaç değişkenimizi while şartını sağlayabilmesi için döngüye girmeden önce tanımlamamız gerekir.

i=1 şeklinde sayaç değişkeni tanımlanmıştır. Bu değişken sayesinde döngümüzün kaç kez çalışacağı belirlenmiş olur.

Adım adım çalıştırdığımızda;

- i değişkenine 1 değeri aktarılır,
- While döngüsüne girer ve i'yi kontrol eder i=1 için i<4 mü? Evet şartı sağladığı için döngünün içerisine girer ve döngü içerisindeki komutları çalıştırır, Ekrana i değişkeninin değeri olarak 1 değerini yazar ve i'nin değerini 1 artırır. i'nin yeni değeri böylece 2 olur.
- Döngünün başına yeniden dönüp i değerini kontrol eder. i=2 için i<4 mü? Evet, şartını sağlar ve yeniden döngüye girer Ekrana i değişkeninin değeri olarak 2 değerini yazar, i değerini 1 artırır ve i'nin yeni değeri 3 olur.
- Döngü başına yeniden döner ve i değerini kontrol eder. i=3 için i<4 mü? Evet, i'nin değeri 4'den küçük olduğu için şartı sağlar ve yeniden döngüye girer. Ekrana 3 değerini yazdırır. İ değişkeninin değerini 1 artırır. Ve i'nin yeni değeri 4 olur.
- Döngü başına yeniden döner ve i değerini yeniden kontrol eder. i=4 için i<4 mü? Hayır, i değişkeninin değeri 4'ten küçük değildir, eşittir şartı sağlamadığından dolayı döngüden çıkar.

Böylece şartımız sağlanana kadar döngü içerisindeki komutlar 3 kere çalıştırmış oluruz.

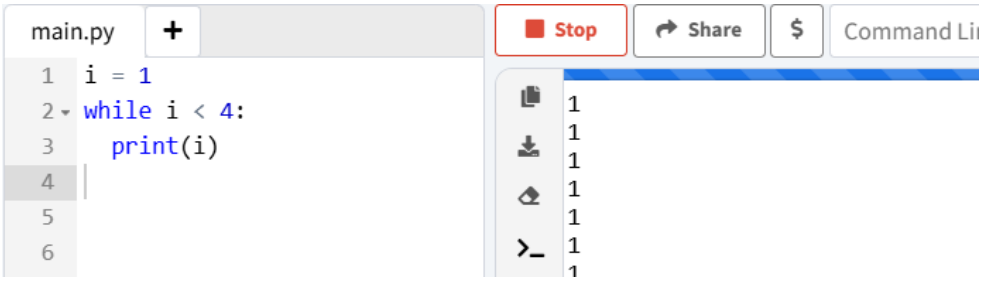
i=1 başlangıç değerine göre, buradaki şart değerimizi artırırsak;

- i<=10 için döngü içerisindeki komutlarımız 10 kere çalışacaktır,
- İ<=100 için döngü içerisindeki komutlarımız 100 kere çalışacaktık.
- i<5 için döngü içerisindeki komutlarımız 4 kere çalışacaktır. (Çünkü burada <= ifadesi olmadığı için i=5 için döngümüz çalışmayacaktır.

Sonsuz Döngü –While Döngüsünde sayaç değişkeninin değerinin artırılması çok önemlidir.

Eğer sayaç değişkeninin değer artım işlemini yaptırmazsanız, döngümüz sonsuz döngüye girer.

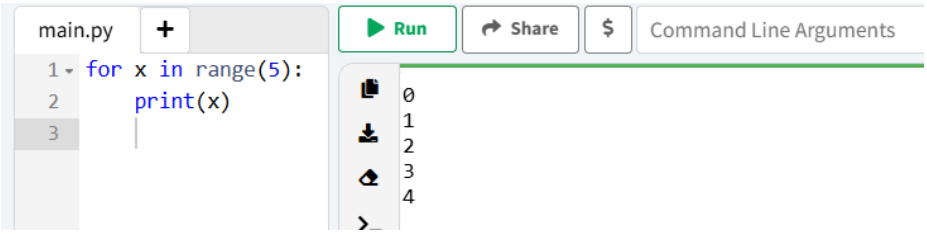
Aşağıdaki örnekteki gibi sayaç değişkeni olarak kullanılan i değişkeninin değeri artırılmadığı için ekrana sürekli "1" şeklinde yazacaktır. Programın durmadığına dikkat ediniz. Stop düğmesi aktif.



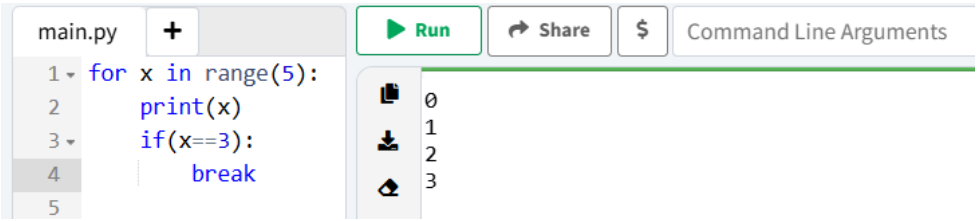
### 6.3. Break ve Continue Komutları

- BREAK; KOMUTU

Döngü içerisindeyken, bu komut çalıştırıldığında hemen döngüden çıkmamızı sağlar. Tüm döngü yapıları ile birlikte kullanabiliriz.




Break komutu ile birlikte aşağıdaki örnekte x değeri 3'e eşit ise döngüden çıkmamızı sağlar

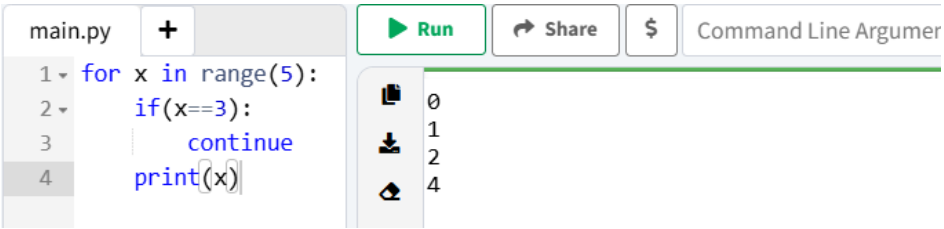



- CONTINUE KOMUTU

Döngü içerisindeyken, bu komut çalıştırıldığında döngü başına dönmemizi sağlar. Tüm döngü yapıları ile birlikte kullanabiliriz.

Örnek 1:

Aşağıdaki kodlarda x değeri kontrol edilir ve eğer 3'e eşit ise continue komutu çalıştırılarak döngü başına gitmesi sağlanır. Kodun çıktısına baktığınızda o yüzden 3 değerinin ekrana yazdırılamadığını görürüz.



# SİMÜLATÖR YAZILIMININ OLUŞTURULMASI

## 7. SİMÜLATÖR YAZILIMLARI

Simülatör yazılımları, gerçek dünyadaki fiziksel bir sistemi, süreci veya cihazı dijital bir ortamda taklit ederek kullanıcıya gerçeğe yakın bir deneyim sunan bilgisayar programlarıdır. Bu yazılımlar; karmaşık matematiksel modeller ve algoritmalar kullanarak, belirli girdilere karşı sistemin nasıl tepki vereceğini önceden görmemizi sağlar. Genellikle eğitim ve mühendislik amacıyla kullanılırlar; böylece yüksek maliyetli, tehlikeli veya erişimi zor olan senaryolar, güvenli ve kontrollü bir sanal alanda defalarca deneyimlenebilir.

Simülatör Yazılımlarının Temel Faydaları

- Güvenlik: Gerçek hayatta hayati risk taşıyan durumların (yüksek gerilim, yüksek basınç vb.) risksiz bir ortamda çalışılmasını sağlar.
- Maliyet Tasarrufu: Gerçek ekipmanların yıpranmasını önler ve enerji gibi operasyonel giderleri ortadan kaldırır.
- Hata Analizi: Yapılan hataların sonuçları anlık olarak izlenebilir ve süreç başa sarılarak tekrar denenebilir.
- Veri Toplama: Sistemin sınırlarını zorlayarak uç durumlarda nasıl performans gösterdiğine dair detaylı veriler sunar.

### 7.1 Endüstriyel Kontrol Simülatör Yazılımları

Endüstriyel elektronik kontrol simülasyon sistemleri, karmaşık otomasyon süreçlerinin ve elektrik devrelerinin fiziksel kuruluma gerek kalmadan dijital ortamda tasarlanıp test edilmesini sağlayan yazılımlardır.

Aşağıda belirtildiği gibi birçok avantaja sahiptir;

- Güvenlik: Gerçek panolarda oluşabilecek çarpılma, patlama veya malzeme bozma riskini sıfıra indirir. Hata yapmaktan korkmadan deneme yapmanıza olanak tanır.
- Maliyet Tasarrufu: Binlerce liralık motor, şalter ve kontaktörü fiziksel olarak satın almadan karmaşık devreler kurmanıza imkan sağlar.
- Öğrenme Hızı: Teorik bilginin pratikte nasıl sonuç verdiğini anında gösterdiği için öğrenme sürecini ciddi şekilde hızlandırır.
- Düşük Sistem Gereksinimi: Çok eski bilgisayarlarda bile kasmadan, akıcı bir şekilde çalışabilir.
- Mantık Geliştirme: Bu programlar hata paylarını minimize eder. PLC programlamaya geçmeden önce "kumanda mantığını" oturtmak için en iyi başlangıç noktasıdır.

Avantajlarının yanısıra teknik adamlar bilmelidir ki;

- Güncellik Sorunu: Yazılım oldukça eskidir ve modern otomasyon sistemlerindeki (modern PLC'ler, servo sürücüler vb.) gelişmeleri tam olarak kapsamayabilir.
- Sınırlı Eleman Sayısı: Çok profesyonel projeler için kütüphanesi yetersiz kalabilir.

- Görsel Basitlik: Daha gelişmiş rakipleri (©TIA Portal, etc) kadar detaylı dinamik görseller sunmayabilir.
- Fiziksel Hataları Kapsamaz: Kablo gevşekliği, oksitlenme veya manyetik etkilenme gibi gerçek hayattaki fiziksel arızaları simüle edemez.

Tüm bunlar ışığında bir simülatör yazılımı oluşturulurken ki biz bundan sonra bu yazılıma iVEBSIM+ diyeceğiz;

- Geniş Kütüphane: Kontaktörler, zaman röleleri, butonlar, asenkron motorlar ve çeşitli sensörler gibi klasik kumanda elemanlarını içerir.
- Dinamik Simülasyon: Devreyi kurduktan sonra "Play" tuşuna basarak sistemin nasıl çalıştığını gerçek zamanlı olarak görebilirsiniz.
- Hata Denetimi: Yanlış bağlantı yapıldığında veya kısa devre oluştuğunda yazılım sizi uyarır.
- Çoklu Dil Desteği: Yerli bir yazılım olmasının avantajıyla tamamen Türkçe arayüz sunar. Çeşitli dilleri destekler.
- Sürükle-Bırak Mantığı: Karmaşık kodlama dilleri yerine görsel bir arayüzle devre tasarımı yapılır.

| **Özellik** | **iVEBSIM+** | **Profesyonel Yazılımlar (TIA Portal vb.)** |
|---|---|---|
| **Kullanım Amacı** | Eğitim ve Temel Mantık | Endüstriyel Uygulama |
| **Zorluk Seviyesi** | Çok Kolay | Zor / Uzmanlık Gerektirir |
| **Maliyet** | Ücretsiz / Erişilebilir | Çok Yüksek |
| **Donanım İhtiyacı** | Düşük | Yüksek |

Tablo 1. Endüstriyel Yazılım Karşılaştırması

## 8. TKINTER – GÖRSEL PROGRAMLAMA

### 8.1 Tkinter Nedir?

Bir yazılımda arayüz oluştururken o programlama diline ait olan grafik kullanıcı arayüz (GUI) kütüphaneleri kullanılır. Tkinter, Python Programlama Dili'nin standart grafik kullanıcı arayüzüdür. Tcl/Tk araç seti üzerine inşa edilmiştir ve Python Programlama Dili ile birlikte otomatik kurulur. Küçük boyutlu ve hızlıdır. Platform bağımsızlığının yanısıra (Windows, macOS, Linux) en önemlisi, kolay öğrenilir ve kullanılır. Kaynak Kodu: Lib/tkinter/__init__.py

### 8.1.1 Mimari

Tcl/Tk tek bir kütüphane değil, her biri ayrı işlevselliğe ve kendi resmi dokümantasyonuna sahip birkaç farklı modülden oluşmaktadır. Python'ın ikili sürümleri de beraberinde bir eklenti modülü sunmaktadır.

Tcl

Tcl, tıpkı Python gibi dinamik yorumlayıcı bir programlama dilidir. Genel amaçlı bir programlama dili olarak kendi başına kullanılabilse de, en yaygın olarak C uygulamalarına bir betik motoru veya Tk araç setine bir arayüz olarak gömülür. Python'dan farklı olarak, Tcl'nin yürütme modeli işbirlikçi çoklu görevlendirme etrafında tasarlanmıştır ve Tkinter bu farkı kapatır (ayrıntılar için İş Parçacığı modeli bölümüne bakın).

Tk

Tk, GUI widget'ları oluşturmak ve manipüle etmek için özel komutlar ekleyen C'de uygulanan bir Tcl paketidir.

Ttk

Temalı Tk (Ttk), birçok klasik Tk widget'ına göre farklı platformlarda çok daha iyi bir görünüm sağlayan daha yeni bir Tk widget ailesidir. Ttk, Tk sürüm 8.5'ten itibaren Tk'nin bir parçası olarak dağıtılır. Python bağlamaları ayrı bir modül olan tkinter.ttk'de sağlanır.

İçsel olarak, Tk ve Ttk, temel işletim sisteminin olanaklarını kullanır, yani Unix/X11'de Xlib, macOS'ta Cocoa, Windows'ta GDI.

### 8.1.2 Tkinter Temel Bileşenleri (Widget)

Widget temel olarak bileşen olarak adlandırılabilir. Belli bir görevi yerine getirirler. Bir Tkinter kullanıcı arayüzü, ayrı ayrı widget'lardan oluşur. Her widget, ttk.Frame, ttk.Label ve ttk.Button gibi sınıflardan örneklenen bir Python nesnesi olarak temsil edilir.Temel Widgetlar:

- **Frame**: Diğer widget'ları gruplamak için çerçeve
- **Label**: Metin veya resim göstermek için etiket
- **Button**: Tıklanabilir buton
- **Entry**: Tek satırlık metin girişi
- **Text**: Çok satırlık metin alanı
- **Canvas**: Çizim ve grafik alan
- **Menu**: Menü çubukları
- **Scrollbar**: Kaydırma çubukları
- **Checkbutton**: Onay kutuları
- **Radiobutton**: Seçim butonları

- **Listbox**: Liste kutuları
- **Scale**: Kaydırıcılar
- **Spinbox**: Sayı seçiciler

### 8.2 Pencere Oluşturma

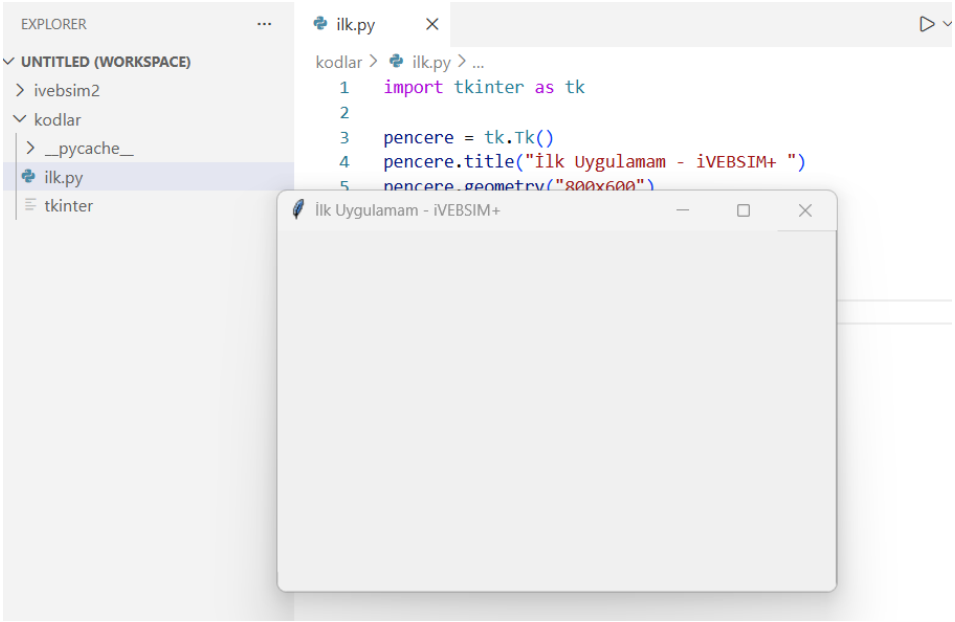
Tkinter'da bir pencere oluşturmak için şu adımları izlersiniz:

1. **Tkinter'ı içe aktarın**
2. **Ana pencere nesnesi oluşturun** (tk.Tk())
3. **Pencere özelliklerini ayarlayın** (başlık, boyut, konum)
4. **Widget'ları ekleyin** (butonlar, etiketler, giriş alanları)
5. **Olayları bağlayın** (tıklama, tuş basma gibi)
6. **Ana döngüyü başlatın** (mainloop())

**Örnek Uygulama**

```
1   import tkinter as tk
2
3   pencere = tk.Tk()
4   pencere.title("İlk Uygulamam - iVEBSIM+ ")
5   pencere.geometry("800x600")
6   pencere.mainloop()
7
8
9
```

Kodlarını çalıştırdığımızda Form sayfamızı aşağıdaki şekilde görürüz.




**İlk Uygulama Örneği - Detaylı İnceleme**
Şimdi ilk Tkinter uygulamamızı adım adım inceleyelim:

**1. İçe Aktarma:**

```
import tkinter as tk
```

Tkinter modülünü "tk" takma adıyla içe aktarır.

**2. Ana Pencere:**

```
pencere = tk.Tk()
```

- tk.Tk() ana pencereyi oluşturur
- Bu pencere tüm diğer widget'ların ana kabıdır
- Her uygulama sadece bir Tk() nesnesi olabilir

**3. Pencere Özellikleri:**

```
pencere.title("İlk Uygulamam - iVEBSIM+ ")
pencere.geometry("800x600")
```

- title(): Başlık çubuğunda görünen metin
- geometry(): Pencere boyutunu piksel olarak ayarlar

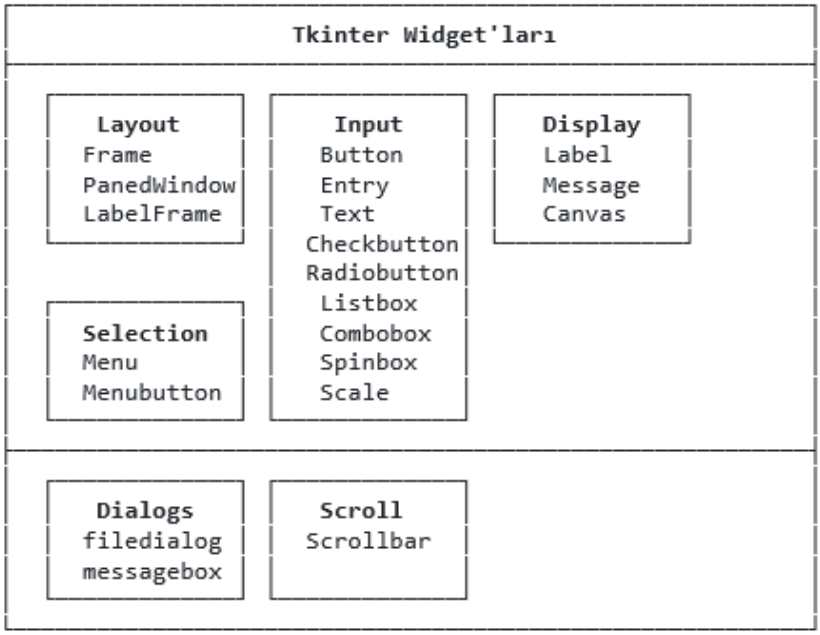
**4. Ana Döngü:**

```
pencere.mainloop()
```

- Pencereyi görünür hale getirir
- Fare ve klavye olaylarını dinler
- Uygulamayı çalışır halde tutar
- Sonsuz döngüdür, ancak pencere kapatılınca biter

### 8.3 Widget'ların Eklenmesi ve Olayların Bağlanması

Widget'lar 8.1.2 de belirttiğimiz Frame, Label, Button, Entry, Text, Canvas, Menu, Scrollbar, Checkbutton, Radiobutton gibi form elemanlarını eklenilmesini sağlar.




Şekil 13. Tkinter Widgetları

### 8.3.1 Etiketlerin Eklenmesi

Ekrana yazı eklemek için kullanılır. Form üzerine etiketlerin eklenebilmesi için Tkinter kütüphanesinden Label Metodu kullanılır.

```
etiket = tk.Label(pencere, text="Merhaba Tkinter!", font=("Arial", 16))
```

- text: Gösterilecek metin
- font: Yazı tipi, boyut ve stil
- bg: Arka plan rengi (background)
- fg: Ön plan rengi (foreground)

### 8.3.2 Butonların Eklenmesi

Tıklanabilir buton oluşturur. Tkinter kütüphanesinden Button Metodu kullanılır.

```
buton = tk.Button(pencere, text="Bana Tıkla", command=selam_ver)
```

- text: Buton üstündeki metin
- command: Tıklama olayında çağrılacak fonksiyon
- state: "normal", "active", "disabled" durumları

### 8.3.3 Yerleşimlerin ayarlanması

pack() nesneleri pencereye yerleştirir. **F**orm üzerinde yatayda ve dikeyde nasıl yer alacaklarının belirleyebilirsiniz.

```
etiket.pack(pady=20)
buton.pack(pady=10)
```

- pady: Dikey boşluk (piksel)
- padx: Yatay boşluk
- side: Hangi tarafta (TOP, BOTTOM, LEFT, RIGHT)
- fill: Boş alanı doldurma (X, Y, BOTH)
- expand: Boş alanda genişleme

**Örnek Uygulama**

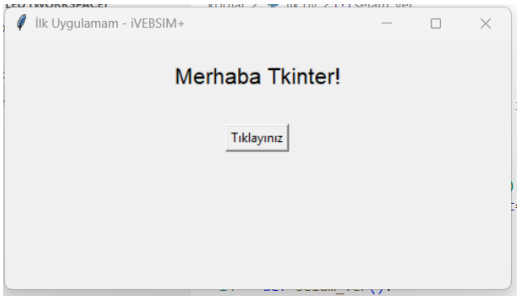
```
import tkinter as tk

# Ana pencere oluştur
pencere = tk.Tk()
pencere.title("İlk Uygulamam - iVEBSIM+ ")
pencere.geometry("800x600")

# Etiket oluştur (metin göster)
etiket = tk.Label(pencere, text="Merhaba Tkinter!", font=("Arial", 16))
etiket.pack(pady=20)

# Tıklama fonksiyonu
def selam_ver():
   etiket.config(text="Butona tıklandı!")
```

```
buton = tk.Button(pencere, text="Tıklayınız", command=selam_ver)
buton.pack(pady=10)

# Ana döngüyü başlat
pencere.mainloop()
```

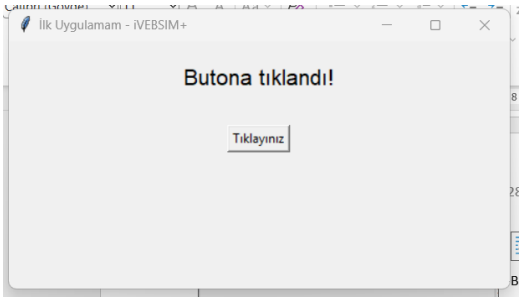
**Bu kod şu işlemleri yapar:**

1. Tkinter kütüphanesini yükler
2. Ana pencereyi oluşturur ve ayarlar
3. "Merhaba Tkinter!" metnini gösteren etiket oluşturur
4. Tıklama fonksiyonunu tanımlar
5. "Tıklayınız" butonunu oluşturur
6. Widget'ları pencereye yerleştirir
7. Pencereyi gösterir ve olayları dinler

**Kodların Çalıştırılması**




Butona tıklanıldığında da formu yandaki gibi görürüz.




### 8.3.4 Entry Eklenmesi

Kullanıcıdan Tek satırlık veri girişi almak için kullanılır .

```
kutu = tk.Entry(pencere)
kutu.pack()
```

**Örnek Uygulama**

```
import tkinter as tk

pencere = tk.Tk()
pencere.title("İlk Uygulamam - iVEBSIM+")
pencere.geometry("800x600")

# Etiket
etiket = tk.Label(pencere, text="Adınız:")
etiket.pack()

# Giriş kutusu
giris = tk.Entry(pencere)
giris.pack()
```

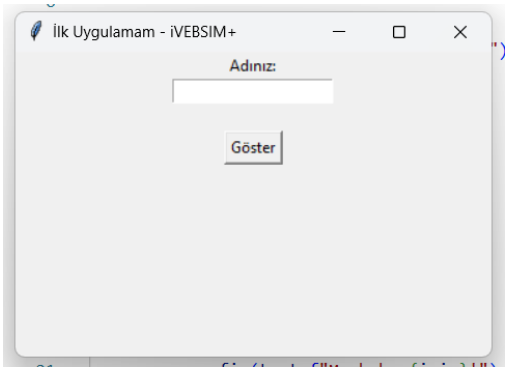
TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Avrupa Birliği tarafından finanse edilmektedir

```
# Sonucu gösterecek etiket
sonuc = tk.Label(pencere, text="")
sonuc.pack()

def goster():
    isim = giris.get()
    sonuc.config(text=f"Merhaba {isim}!")

buton = tk.Button(pencere, text="Göster", command=goster)
buton.pack()

pencere.mainloop()
```

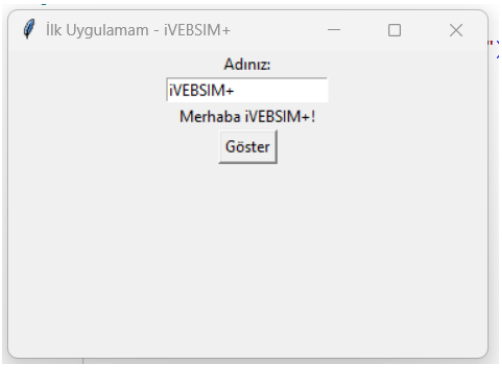
**Bu kod şu işlemleri yapar:**

1. Tkinter kütüphanesini yükler
2. Ana pencereyi oluşturur ve ayarlar
3. "Adınız" metnini gösteren etiket oluşturur
4. Adın girilmesi için Giriş kutusunu oluşturur
5. Sonucun gösterileceği Etiket oluşturulur.
6. Tıklama fonksiyonunu tanımlar
7. "Göster" butonunu oluşturur
8. Pencereyi gösterir ve olayları dinler

**Kodların Çalıştırılması**




Butona tıklanıldığında da formu yandaki gibi görürüz.




### 8.3.5 Radiobutton Eklenmesi

Kullanıcının birden fazla seçenek içerisinden sadece bir tanesini seçmesi için kullanılır.

```
secenek = tk.StringVar()
r1 = tk.Radiobutton(pencere, text="Erkek", variable=secenek, value="Erkek")
r2 = tk.Radiobutton(pencere, text="Kadın", variable=secenek, value="Kadın")
r1.pack()
r2.pack()
```

### 8.3.6 Checkbutton Eklenmesi

Kullanıcının kendisine verilen seçeneklerden bir ya da daha fazlasını seçmesi için kullanılır.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Avrupa Birliği tarafından finanse edilmektedir

```
secim = tk.IntVar()
check = tk.Checkbutton(pencere, text="Python seviyorum", variable=secim)
check.pack()
```

### 8.3.7 Scrollbar Eklenmesi

Form ekranına kaydırma çubuğu eklenmesi için kullanılır.

```
scroll = tk.Scrollbar(pencere)
scroll.pack(side="right", fill="y")
metin = tk.Text(pencere, yscrollcommand=scroll.set)
metin.pack()
scroll.config(command=metin.yview)
```

### 8.3.8 Menu Eklenmesi

Üst menü çubuğu eklemek için kullanılır.

```
menu = tk.Menu(pencere)
pencere.config(menu=menu)
dosya = tk.Menu(menu)
menu.add_cascade(label="Dosya", menu=dosya)
dosya.add_command(label="Çıkış", command=pencere.quit)
```

### 8.3.9 Olayların Bağlanması

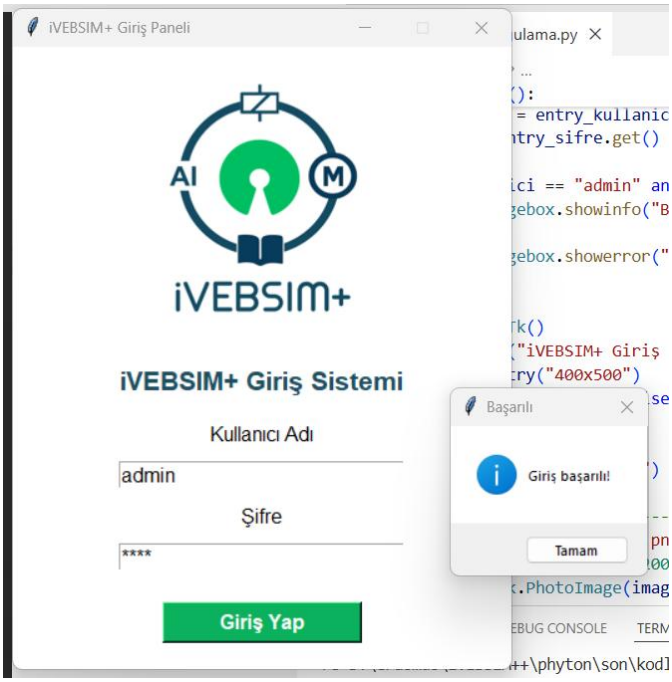
Widget'larda tıklanma durumlarında çalımasını istediğimiz kodları fonksiyon içerisinde tan ımlayabiliriz.

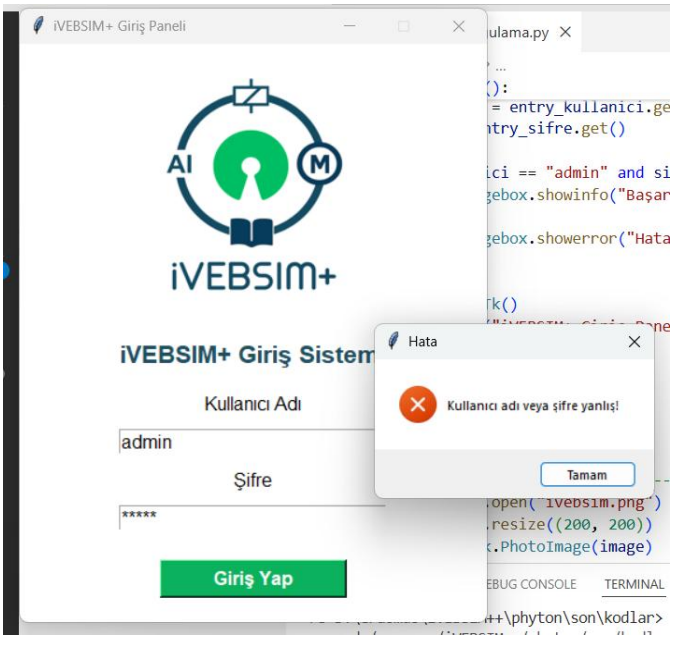
```
 def goster():
    isim = giris.get()
    sonuc.config(text=f"Merhaba {isim}!")
buton = tk.Button(pencere, text="Göster", command=goster)
```

**Örnek Kodlar :**

```
import tkinter as tk
from tkinter import messagebox
from PIL import Image, ImageTk   # Pillow kütüphanesi gerekli

# Giriş fonksiyonu
def giris_yap():
    kullanici = entry_kullanici.get()
    sifre = entry_sifre.get()

    if kullanici == "admin" and sifre == "1234":
        messagebox.showinfo("Başarılı", "Giriş başarılı!")
    else:
        messagebox.showerror("Hata", "Kullanıcı adı veya şifre yanlış!")
```

```
# Ana pencere
pencere = tk.Tk()
pencere.title("iVEBSIM+ Giriş Paneli")
pencere.geometry("400x500")
pencere.resizable(False, False)
# Arka plan rengi
pencere.configure(bg="white")

# ---------- LOGO ----------
image = Image.open("ivebsim.png")
image = image.resize((200, 200))  # Logo boyutu
logo = ImageTk.PhotoImage(image)

logo_label = tk.Label(pencere, image=logo, bg="white")
logo_label.pack(pady=20)

# ---------- BAŞLIK ----------
baslik = tk.Label(pencere, text="iVEBSIM+ Giriş Sistemi",
                  font=("Arial", 16, "bold"),
                  bg="white",
                  fg="#1f4e5f")
baslik.pack(pady=10)

# ---------- KULLANICI ADI ----------
label_kullanici = tk.Label(pencere, text="Kullanıcı Adı",
                           font=("Arial", 12),
                           bg="white")
label_kullanici.pack(pady=5)

entry_kullanici = tk.Entry(pencere, font=("Arial", 12),
width=25)
entry_kullanici.pack(pady=5)

# ---------- ŞİFRE ----------
label_sifre = tk.Label(pencere, text="Şifre",
                       font=("Arial", 12),
                       bg="white")
label_sifre.pack(pady=5)

entry_sifre = tk.Entry(pencere, font=("Arial", 12),
width=25, show="*")
entry_sifre.pack(pady=5)

# ---------- BUTON ----------
giris_buton = tk.Button(pencere,
                        text="Giriş Yap",
                        font=("Arial", 12, "bold"),
                        bg="#0bb15d",
                        fg="white",
                        width=15,
                        command=giris_yap)
```



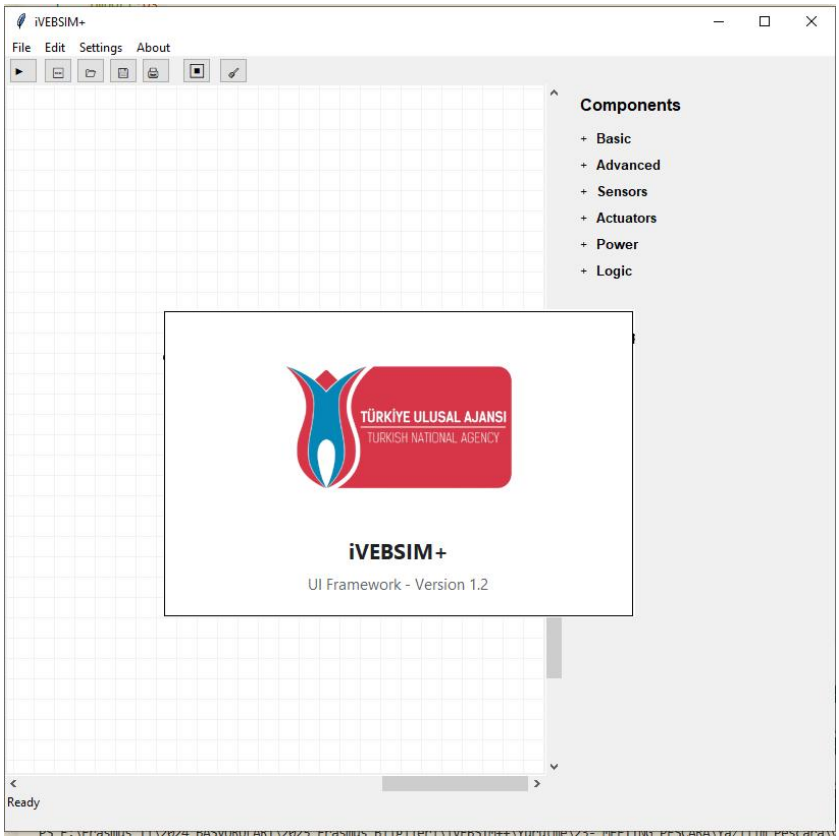

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Avrupa Birliği tarafından finanse edilmektedir

```
giris_buton.pack(pady=20)

pencere.mainloop()
```

## 9. SİMÜLATÖR ARAYÜZÜNÜN OLUŞTURULMASI

Simülatör arayüzünün anlatımına başlamadan önce önemli bir açıklama yapılmalıdır. Simülatör arayüzünün anlatımı genel başlıklar üzerinden gidilerek anlatılmıştır. Her genel başlığa ait örnek program parçacıkları ana yazılımdan alıntılanmıştır. Binlerce satırdan oluşan bir programda programın satır satır anlatılmasına imkan yoktur. Bu sebeple çok fazla kod ayrıntısına girilmemiştir. Yazılımı oluşturan kodların tamamı kitapçığın sonunda basılı ve proje web sitesinde dijital olarak yer almaktadır.

Simülatör arayüzü çalışmaya başladığında önce bir karşılama ekranı gelmektedir. Bu ekranda bu programa destek veren kurumlar yazmaktadır. Bu pencere fare ile kliklendiğinde programın arayüzü ekrana gelmektedir. Bizim yazılımımızda karşılama ekranı main.py ve program ana penceresi ui.py olarak isimlendirilmiştir.




Şekil 14 – Karşılama Ekranı

### 9.1 Karşılama Ekranı Oluşturulması

Uygulama başlatmak ve kullanıcıya bir Karşılama (Splash) Ekranı göstermek için şu adımlar izlenir:

Python Tkinter kütüphanesi eklenir.

```
import os

import sys
import tkinter as tk
from ui import App
```

Root olarak verilen ana TK penceresi üzerinde bir üst pencere yani karşılama ekranı (splash) oluşturulur.

```
def show_splash(root):
    splash = tk.Toplevel(root)
    splash.overrideredirect(True)#açıklma
```

Pencere büyüklüğü sabit olarak görünür.

```
# Initial size
    width, height = 460, 300
    splash.update_idletasks()
    sw = splash.winfo_screenwidth()
    sh = splash.winfo_screenheight()
    x = (sw - width) // 2
    y = (sh - height) // 2
    splash.geometry(f"{width}x{height}+{x}+{y}")

    container = tk.Frame(splash, bg="#ffffff", bd=1, relief="solid")
    container.pack(fill="both", expand=True)
```

Pencere içeriği oluşturulur.

```
# Logo image
    logo_path = resource_path("sembol", "logoarkaplan.png")
    logo_img = None
    if logo_img is not None:
        logo = tk.Label(container, image=logo_img, bg="#ffffff")
        logo.image = logo_img
        logo.pack(pady=(18, 8))
    else:
        logo = tk.Label(container, text="⚡", font=("Segoe UI", 56),
bg="#ffffff", fg="#1a73e8")
        logo.pack(pady=(22, 6))

    title = tk.Label(container, text="iVEBSIM+", font=("Segoe UI", 16,
"bold"), bg="#ffffff", fg="#202124")
    title.pack()
```

```
    subtitle = tk.Label(container, text="UI Framework - Version 1.2",
font=("Segoe UI", 11), bg="#ffffff", fg="#5f6368")
    subtitle.pack(pady=(2, 12))

    about = tk.Label(
```

Karşılama ekranı boyutu tekrar hesaplanır ve ortalanır.

```
# Recenter after layout so size is accurate
    splash.update_idletasks()
    width = splash.winfo_width()
    height = splash.winfo_height()
    sw = splash.winfo_screenwidth()
    sh = splash.winfo_screenheight()
    x = (sw - width) // 2
    y = (sh - height) // 2
    splash.geometry(f"{width}x{height}+{x}+{y}")
```


Şekil 15 – Splash Görünümü

Karşılama ekranı tıklama ve tuş basımına bağlanarak, “proceed” tetiklenir. Proceed karşılama ekranını kapatır ve ana ekranı çağırır.

```
def proceed(event=None):
        try:
            splash.destroy()
        except Exception:
            pass
        root.deiconify()
        root.geometry("1100x780")
        App(root)
        root.focus_force()

    # Close on any click or key
    for widget in (splash, container, logo, title, subtitle, about):
        widget.bind("<Button-1>", proceed)
        widget.bind("<Key>", proceed)
```

Karşılama ekranı çağrılır ve olay döngüsü başlatılır.

```
def run():
    root = tk.Tk()
    root.withdraw()
    show_splash(root)
    root.mainloop()


if __name__ == "__main__":
    run(
```

### 9.2 Arayüz Altyapısının Oluşturulması

``ui.py` dosyası, uygulamanın grafiksel arayüzünü sağlayan `App` sınıfını tanımlar. Uygulama; menüler, araç çubuğu, orta kısımda çizim alanı (`Canvas`) ve sağ/sol yan paneller (bileşen akordeonu) içerir. Uygulama arayüzünü oluşturmak için bir önceki bahsedilen konulara ek olarak şu adımlar izlenir:

`build.ui içerisinde ekran penceresi oluşturulur.

```
def _build_ui(self):
        self.master.title("iVEBSIM+")
        self.grid(row=0, column=0, sticky="nsew")
        self.master.rowconfigure(0, weight=1)
        self.master.columnconfigure(0, weight=1)
```

Menü çubuğu (File / Edit / Settings / About) `tk.Menu` ile oluşturulur. Her menü komutu bir metoda bağlanır (`_new_file`, `_save_file` vb.).

```
# Menu bar
        self.menubar = tk.Menu(self.master)

        # File menu
        self.menu_file = tk.Menu(self.menubar, tearoff=0)
        self.menu_file.add_command(label=self._t("new"),
command=self._new_file)
        self.menu_file.add_command(label=self._t("open"),
command=self._open_file)
```

Araç çubuğu ‘ttk.Button’ ile oluşturulur. Kaydet, Başlat, Durdur, Temizle vb. gibi butonlar eklenir.

```
self.toolbar = ttk.Frame(self)
        self.toolbar.grid(row=0, column=0, sticky="ew")
```

```
        self.btn_start = ttk.Button(self.toolbar, text="▶", width=3,
command=self.start_sim)
        self.btn_start.pack(side="left", padx=4)
        # Icon-only file action buttons next to Start
        self.btn_new = ttk.Button(self.toolbar, text="🆕", width=3,
command=self.clear)
        self.btn_clear.pack(side="left", padx=4)
```

Ana alan ttk.PanedWindow ile ayarlanır ve `PanedWindow` ile iki panel olacak şekilde bölümlendirilir. Sol taraf çizim (`Canvas`) alanını içeren `canvas_frame` , sağ taraf bileşen akordeonu içeren `side_panel` olarak ayarlanır.

Canvas alanı "scrollregion" ile oluşturulmuş ve yatayda ve dikeyde scrollbar'lar eklenmiştir. Canvas alanı yeniden boyutlandırıldığında "configure" ile ızgara güncellenecektir. Aynı zamanda `Canvas` üzerinde koordinat kullanırken `canvasx` ve `canvasy` metotlarını kullanılmaktadır. Böylece ekrandan kanvasa dönüşüm sağlanmaktadır.

```
 self.main = ttk.PanedWindow(self, orient=tk.HORIZONTAL)
        self.main.grid(row=1, column=0, sticky="nsew")
        self.rowconfigure(1, weight=1)
        self.columnconfigure(0, weight=1)

        # Canvas area
        self.canvas_frame = ttk.Frame(self.main)
        self.main.add(self.canvas_frame, weight=3)
        self.canvas_frame.rowconfigure(0, weight=1)
        self.canvas_frame.columnconfigure(0, weight=1)
        self.canvas = tk.Canvas(self.canvas_frame, bg="#ffffff",
scrollregion=(0, 0, 2000, 2000))
        self.canvas.grid(row=0, column=0, sticky="nsew")
        self.h_scroll = ttk.Scrollbar(self.canvas_frame,          # Redraw
grid on resize/scroll when enabled
        self.canvas.bind("<Configure>", lambda e:
(self._grid_enabled.get() and self._draw_grid()))
        self.canvas.bind_all("<MouseWheel>", lambda e:
(self._grid_enabled.get() and self._draw_grid()))
```

Yan panelde akordeon tarzı kategoriler (`basic`, `advanced`, `sensors`, `actuators`, `power`, `logic`) - `_add_accordion_section` ile başlık ve gövde olarak oluşturulur. Ayrıca en alta bir dip logo eklenmiştir. Akordeon işlemleri "def _add_accordion_section" ile yapılmaktadır. "İndicator" ile açık/kapalı kontrolü sağlanmaktadır.

```
        # Side panel with categorized components (accordion)
        self.side_panel = ttk.Frame(self.main, width=260)
```

```
        self.main.add(self.side_panel, weight=1)

              # Make accordion responsive to width changes
        self.side_canvas.bind(
            "<Configure>",

        # Build stacked, collapsible categories
        self._accordion_sections = []
        self._add_accordion_section(self._t("basic"), expanded=True)
        self._add_accordion_section(self._t("advanced"), expanded=False)

        # Footer logo image bottom-right
        footer = ttk.Frame(self.side_panel)
        footer.pack(side="bottom", fill="x", padx=10, pady=8)
        self._footer_logo_img = None
def _add_accordion_section(self, title, expanded=False):
        """Create one collapsible accordion section with empty
placeholder list."""
        # Header
        header_frame = ttk.Frame(self.accordion)
        header_frame.pack(fill="x", expand=False, padx=10, pady=(6, 0))

        # Toggle indicator
        indicator = tk.StringVar(value="-" if expanded else "+")

        # Body
        body_frame = ttk.Frame(self.accordion)
        if expanded:
            body_frame.pack(after=header_frame, fill="x", expand=False,
padx=10, pady=(2, 6))

        # Empty state content
        empty = ttk.Label(body_frame,
text=self._t("components_empty").format(category=title),
foreground="gray")
        empty.pack(fill="x", expand=False, padx=6, pady=8)

        # Toggle logic
        def toggle():
            if body_frame.winfo_manager():
                body_frame.forget()
                indicator.set("+")
            else:
                body_frame.pack(after=header_frame, fill="x",
expand=False, padx=10, pady=(2, 6))

        self._accordion_sections.append((header_frame, body_frame,
indicator))
```

Izgara (Grid) ve Görünüm Fonksiyonları şu şekilde gerçekleştirilir.

```
    def _toggle_grid(self):
        if self._grid_enabled.get():
            self._draw_grid()
        else:
            self._clear_grid()
```

Menü Komutları ve Yardımcı Metodlar ayrı ayrı tanımlanmıştır.

- Dosya işlemleri (yer tutucu): `_new_file`, `_open_file`, `_save_file`, `_save_as_file`

- Düzen işlemleri (yer tutucu): `_undo`, `_redo`, `_cut`, `_copy`, `_paste`.

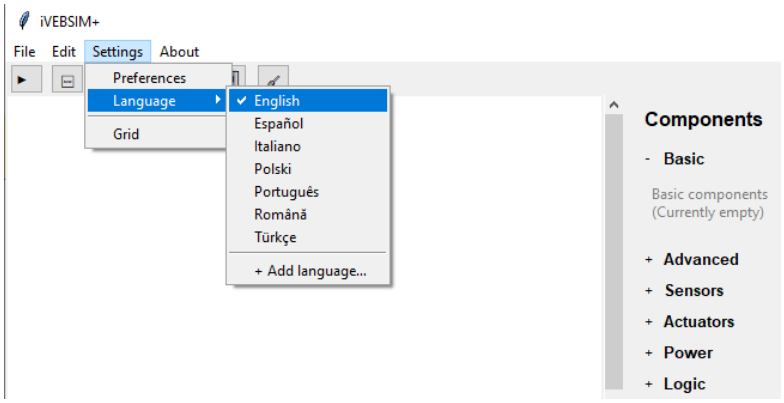
- Ayarlar: `_show_preferences`, `_change_language` (dil değiştirme fonksiyonu yer tutucu), Grid toggle checkbutton.

- Hakkımızda: `_show_about` → uygulama hakkında bilgi penceresi gösterir.

```
def _new_file(self):
        self._status(self._t("new_file_clicked"))
        messagebox.showinfo(self._t("info"), self._t("new_file_info"))

    def _open_file(self):
        self._status(self._t("open_file_clicked"))
        messagebox.showinfo(self._t("info"), self._t("open_file_info"))

    def _save_file(self):
        self._status(self._t("save_file_clicked"))
        messagebox.showinfo(self._t("info"), self._t("save_file_info"))
    def _undo(self):
        self._status(self._t("undo_clicked"))
        messagebox.showinfo(self._t("info"), self._t("undo_info"))

    def _redo(self):
        self._status(self._t("redo_clicked"))
        messagebox.showinfo(self._t("info"), self._t("redo_info"))

    def _show_preferences(self):
        self._status(self._t("preferences_clicked"))
        messagebox.showinfo(self._t("preferences"),
self._t("preferences_info"))
```

**9.3 Çok Dillilik**

Yazılım çok dillilik üzerine kuruludur.




Şekil 16 – Çok Dillilik Ekran Görüntüsü

Programda çok dilliliği sağlamak için Python i18n kütüphanesi kullanılmaktadır. Uygulama başlatılırken (__init__ metodunda), varsayılan dil kodu "en" (İngilizce) olarak ayarlanır.

```
# i18n setup
        self.lang_code = "en"
        self.translations = self._load_translations(self.lang_code)
```

Dil dosyaları languages klasöründe JSON formatında saklanır (örneğin: en.json, tr.json, es.json vb.). Her JSON dosyası "code" ve "name" zorunlu alanlar (dil kodu ve görünür adı) gibi bir yapıya sahiptir. config.json dosyası seçili dili saklar: {"selected_language": "en"}. Bu, uygulama yeniden açıldığında son seçilen dili hatırlaması için.


Şekil 17 – json Dosyaları

Çeviri Yükleme Fonksiyonu (_load_translations) Parametre olarak dil kodu alır (örneğin "tr") ve languages/{code}.json dosyasını açar ve JSON'u yükler.

```
def _load_translations(self, code: str) -> dict:
        path = os.path.join(self._languages_dir(), f"{code}.json")
        try:
            with open(path, "r", encoding="utf-8") as f:
```

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Avrupa Birliği tarafından finanse edilmektedir

```
                data = json.load(f)
                if isinstance(data, dict):
                    return data
        except Exception:
            pass
```

Settings menüsünde "Language" alt menüsü vardır. _populate_language_menu() fonksiyonu languages klasöründeki tüm JSON dosyalarını tarar. Her dosyadan "code" ve "name"'i alır. Kullanıcı bir dil seçtiğinde _on_language_selected() çağrılır , lang_code güncellenir.

```
def _populate_language_menu(self):
        try:
            self.menu_language.delete(0, "end")
        except Exception:
            return
def _on_language_selected(self):
        selected = self._language_var.get()
        if selected == getattr(self, "lang_code", None):
            return
        self.lang_code = selected
        self.translations = self._load_translations(self.lang_code)
        self._save_selected_language(self.lang_code)
        self._rebuild_ui()
```

Dil seçildikten sonra UI tamamen yeniden inşa edilir. Dil değiştiğinde tüm widget'ları yok etmek ve yeniden oluşturmak gerekmektedir. Tkinter'da metinler dinamik değişmez. Bu sebeple Menü bar kaldırılır ve tüm UI'yi sıfırdan kurar.

```
def _rebuild_ui(self):
        try:
            self.master.config(menu=None)
        except Exception:
            pass
        for child in self.winfo_children():
            try:
                child.destroy()
            except Exception:
                pass
        self._grid_lines = []
        self._accordion_sections = []
        self._build_ui()
```

## 10. ELEMANLARIN EKLENMESİ VE SİMÜLASYONUN OLUŞTURULMASI

NEXT ROMANIA……

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Avrupa Birliği tarafından finanse edilmektedir

## KODLAR

**MAIN.PY**

```
import os
import sys
import tkinter as tk
from PIL import Image, ImageTk

from ui import App


def resource_path(*parts):
    """Get absolute path to resource, works for dev and for PyInstaller
onefile."""
    base_path = getattr(sys, "_MEIPASS",
os.path.dirname(os.path.abspath(__file__)))
    return os.path.join(base_path, *parts)


def show_splash(root):
    splash = tk.Toplevel(root)
    splash.overrideredirect(True)

    # Initial size
    width, height = 460, 300
    splash.update_idletasks()
    sw = splash.winfo_screenwidth()
    sh = splash.winfo_screenheight()
    x = (sw - width) // 2
    y = (sh - height) // 2
    splash.geometry(f"{width}x{height}+{x}+{y}")

    container = tk.Frame(splash, bg="#ffffff", bd=1, relief="solid")
    container.pack(fill="both", expand=True)

    # Logo image
    logo_path = resource_path("sembol", "logoarkaplan.png")
    logo_img = None
    try:
        if os.path.exists(logo_path):
            im = Image.open(logo_path)
            # Fit into splash width with some margin
            target_w = 280
            ratio = target_w / max(1, im.width)
            target_h = int(im.height * ratio)
            im = im.resize((target_w, target_h))
            logo_img = ImageTk.PhotoImage(im)
    except Exception:
```

```
        logo_img = None
    if logo_img is not None:
        logo = tk.Label(container, image=logo_img, bg="#ffffff")
        logo.image = logo_img
        logo.pack(pady=(18, 8))
    else:
        logo = tk.Label(container, text="⚡", font=("Segoe UI", 56),
bg="#ffffff", fg="#1a73e8")
        logo.pack(pady=(22, 6))

    title = tk.Label(container, text="iVEBSIM+", font=("Segoe UI", 16,
"bold"), bg="#ffffff", fg="#202124")
    title.pack()

    subtitle = tk.Label(container, text="UI Framework - Version 1.2",
font=("Segoe UI", 11), bg="#ffffff", fg="#5f6368")
    subtitle.pack(pady=(2, 12))

    about = tk.Label(
        container,
        text=(
            "Basit arayüz önizlemesi\n"
            "Bileşen kategorileri (şimdilik boş)\n"
            "Devam etmek için herhangi bir yere tıklayın"
        ),
        justify="center",
        font=("Segoe UI", 10),
        bg="#ffffff",
        fg="#5f6368",
    )
    about.pack(pady=(0, 16))

    # Recenter after layout so size is accurate
    splash.update_idletasks()
    width = splash.winfo_width()
    height = splash.winfo_height()
    sw = splash.winfo_screenwidth()
    sh = splash.winfo_screenheight()
    x = (sw - width) // 2
    y = (sh - height) // 2
    splash.geometry(f"{width}x{height}+{x}+{y}")

    def proceed(event=None):
        try:
            splash.destroy()
        except Exception:
            pass
        root.deiconify()
```

```
        root.geometry("1100x780")
        App(root)
        root.focus_force()

    # Close on any click or key
    for widget in (splash, container, logo, title, subtitle, about):
        widget.bind("<Button-1>", proceed)
        widget.bind("<Key>", proceed)

    # Make sure splash is above
    splash.attributes("-topmost", True)
    splash.after(50, lambda: splash.attributes("-topmost", False))


def run():
    root = tk.Tk()
    root.withdraw()
    show_splash(root)
    root.mainloop()


if __name__ == "__main__":
    run()
```

**UI.PY**

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
from PIL import Image, ImageTk
import os
import sys
import json
import shutil

from core import SimulationCore, DEFAULT_WIDTH, DEFAULT_HEIGHT




class App(tk.Frame):
    def __init__(self, master, symbols_dir=None):
        super().__init__(master)
        self.master = master
        self.symbols_dir = symbols_dir  # Not used in this version
        self.core = SimulationCore()
        self.dragging_component_key = None
        self.ghost_item = None
        self.comp_id_to_canvas = {}
        self.comp_id_to_text = {}
        self.comp_id_to_ports = {}
        self.cable_id_to_lines = {}
        self._anim_tick = 0
        self._cable_wave = 0
        self._cable_wave_step = 2
        self._tick_interval_ms = 120

        # Cable editing
        self._dragging_cable = None
        self._dragging_segment = None
        self._drag_start_x = 0
        self._drag_start_y = 0

        # i18n setup
        self.lang_code = "en"
        self.translations = self._load_translations(self.lang_code)

        self._build_ui()
        self._status(self._t("ready"))

    # Resource helper
    def _resource_path(self, *parts) -> str:
```

```
        base_path = getattr(sys, "_MEIPASS",
os.path.dirname(os.path.abspath(__file__)))
        return os.path.join(base_path, *parts)

    # UI building
    def _build_ui(self):
        self.master.title("iVEBSIM+")
        self.grid(row=0, column=0, sticky="nsew")
        self.master.rowconfigure(0, weight=1)
        self.master.columnconfigure(0, weight=1)

        # Menu bar
        self.menubar = tk.Menu(self.master)

        # File menu
        self.menu_file = tk.Menu(self.menubar, tearoff=0)
        self.menu_file.add_command(label=self._t("new"),
command=self._new_file)
        self.menu_file.add_command(label=self._t("open"),
command=self._open_file)
        self.menu_file.add_command(label=self._t("save"),
command=self._save_file)
        self.menu_file.add_command(label=self._t("save_as"),
command=self._save_as_file)
        self.menu_file.add_separator()
        self.menu_file.add_command(label=self._t("exit"),
command=self._exit_app)
        self.menubar.add_cascade(label=self._t("file"), menu=self.menu_file)

        # Edit menu
        self.menu_edit = tk.Menu(self.menubar, tearoff=0)
        self.menu_edit.add_command(label=self._t("undo"), command=self._undo)
        self.menu_edit.add_command(label=self._t("redo"), command=self._redo)
        self.menu_edit.add_separator()
        self.menu_edit.add_command(label=self._t("cut"), command=self._cut)
        self.menu_edit.add_command(label=self._t("copy"), command=self._copy)
        self.menu_edit.add_command(label=self._t("paste"),
command=self._paste)
        self.menubar.add_cascade(label=self._t("edit"), menu=self.menu_edit)

        # Settings menu
        self.menu_settings = tk.Menu(self.menubar, tearoff=0)
        self.menu_settings.add_command(label=self._t("preferences"),
command=self._show_preferences)
        # Language submenu
        self.menu_language = tk.Menu(self.menu_settings, tearoff=0)
        self._language_var = tk.StringVar(value=self.lang_code)
        self._populate_language_menu()
```

```
        self.menu_settings.add_cascade(label=self._t("language"),
menu=self.menu_language)
        self.menu_settings.add_separator()
        self._grid_enabled = tk.BooleanVar(value=False)
        self.menu_settings.add_checkbutton(label=self._t("grid"),
variable=self._grid_enabled, command=self._toggle_grid)
        self.menubar.add_cascade(label=self._t("settings"),
menu=self.menu_settings)

        # About menu
        self.menu_about = tk.Menu(self.menubar, tearoff=0)
        self.menu_about.add_command(label=self._t("about"),
command=self._show_about)
        self.menubar.add_cascade(label=self._t("about"), menu=self.menu_about)

        self.master.config(menu=self.menubar)

        self.toolbar = ttk.Frame(self)
        self.toolbar.grid(row=0, column=0, sticky="ew")

        self.btn_start = ttk.Button(self.toolbar, text="▶", width=3,
command=self.start_sim)
        self.btn_start.pack(side="left", padx=4)
        # Icon-only file action buttons next to Start
        self.btn_new = ttk.Button(self.toolbar, text="🆕", width=3,
command=self._new_file)
        self.btn_new.pack(side="left", padx=2)
        self.btn_open = ttk.Button(self.toolbar, text="📂", width=3,
command=self._open_file)
        self.btn_open.pack(side="left", padx=2)
        self.btn_save = ttk.Button(self.toolbar, text="💾", width=3,
command=self._save_file)
        self.btn_save.pack(side="left", padx=2)
        self.btn_print = ttk.Button(self.toolbar, text="🖨", width=3,
command=self._print_canvas)
        self.btn_print.pack(side="left", padx=(2, 8))
        self.btn_stop = ttk.Button(self.toolbar, text="⏹", width=3,
command=self.stop_sim)
        self.btn_stop.pack(side="left", padx=4)
        self.btn_clear = ttk.Button(self.toolbar, text="🧹", width=3,
command=self.clear)
        self.btn_clear.pack(side="left", padx=4)

        self.main = ttk.PanedWindow(self, orient=tk.HORIZONTAL)
        self.main.grid(row=1, column=0, sticky="nsew")
        self.rowconfigure(1, weight=1)
        self.columnconfigure(0, weight=1)
```

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Avrupa Birliği tarafından finanse edilmektedir

```
        # Canvas area
        self.canvas_frame = ttk.Frame(self.main)
        self.main.add(self.canvas_frame, weight=3)
        self.canvas_frame.rowconfigure(0, weight=1)
        self.canvas_frame.columnconfigure(0, weight=1)
        self.canvas = tk.Canvas(self.canvas_frame, bg="#ffffff",
scrollregion=(0, 0, 2000, 2000))
        self.canvas.grid(row=0, column=0, sticky="nsew")
        self.h_scroll = ttk.Scrollbar(self.canvas_frame, orient="horizontal",
command=self.canvas.xview)
        self.h_scroll.grid(row=1, column=0, sticky="ew")
        self.v_scroll = ttk.Scrollbar(self.canvas_frame, orient="vertical",
command=self.canvas.yview)
        self.v_scroll.grid(row=0, column=1, sticky="ns")
        self.canvas.configure(xscrollcommand=self.h_scroll.set,
yscrollcommand=self.v_scroll.set)
        self._grid_lines = []
        # Redraw grid on resize/scroll when enabled
        self.canvas.bind("<Configure>", lambda e: (self._grid_enabled.get()
and self._draw_grid()))
        self.canvas.bind_all("<MouseWheel>", lambda e:
(self._grid_enabled.get() and self._draw_grid()))

        # Side panel with categorized components (accordion)
        self.side_panel = ttk.Frame(self.main, width=260)
        self.main.add(self.side_panel, weight=1)

        # Components title
        components_label = ttk.Label(self.side_panel,
text=self._t("components"), font=("Arial", 12, "bold"))
        components_label.pack(pady=(10, 5), padx=10, anchor="w")

        # Scrollable container for accordion
        self.side_canvas = tk.Canvas(self.side_panel, highlightthickness=0)
        self.side_scrollbar = ttk.Scrollbar(self.side_panel,
orient="vertical", command=self.side_canvas.yview)
        self.side_canvas.configure(yscrollcommand=self.side_scrollbar.set)
        self.side_canvas.pack(side="left", fill="both", expand=True)
        self.side_scrollbar.pack(side="right", fill="y")

        self.accordion = ttk.Frame(self.side_canvas)
        self._accordion_window = self.side_canvas.create_window((0, 0),
window=self.accordion, anchor="nw")

        # Make accordion responsive to width changes
        self.side_canvas.bind(
            "<Configure>",
```

```
            lambda e: self.side_canvas.itemconfigure(self._accordion_window,
width=e.width)
        )
        self.accordion.bind(
            "<Configure>",
            lambda e:
self.side_canvas.configure(scrollregion=self.side_canvas.bbox("all"))
        )

        # Build stacked, collapsible categories
        self._accordion_sections = []
        self._add_accordion_section(self._t("basic"), expanded=True)
        self._add_accordion_section(self._t("advanced"), expanded=False)
        self._add_accordion_section(self._t("sensors"), expanded=False)
        self._add_accordion_section(self._t("actuators"), expanded=False)
        self._add_accordion_section(self._t("power"), expanded=False)
        self._add_accordion_section(self._t("logic"), expanded=False)

        # Footer logo image bottom-right
        footer = ttk.Frame(self.side_panel)
        footer.pack(side="bottom", fill="x", padx=10, pady=8)
        self._footer_logo_img = None
        try:
            logo_path = self._resource_path("sembol", "logoarkaplan.png")
            if os.path.exists(logo_path):
                im = Image.open(logo_path)
                # scale to side panel width approx
                target_w = 120
                ratio = target_w / max(1, im.width)
                target_h = int(im.height * ratio)
                im = im.resize((target_w, target_h))
                self._footer_logo_img = ImageTk.PhotoImage(im)
        except Exception:
            self._footer_logo_img = None
        if self._footer_logo_img is not None:
            footer_logo = ttk.Label(footer, image=self._footer_logo_img,
anchor="se")
        else:
            footer_logo = ttk.Label(footer, text="⚡", anchor="se",
font=("Segoe UI", 16))
        footer_logo.pack(anchor="se")

        # Bindings
        self.canvas.tag_bind("port", "<ButtonPress-1>", self._on_port_press)
        self.canvas.tag_bind("port", "<ButtonRelease-1>",
self._on_port_release)
        self.canvas.tag_bind("component", "<ButtonPress-1>",
self._on_component_press)
```

```
        self.canvas.tag_bind("component", "<B1-Motion>",
self._on_component_motion)
        self.canvas.tag_bind("component", "<ButtonRelease-1>",
self._on_component_release)
        self.canvas.tag_bind("component", "<Button-3>",
self._on_component_right_click)
        self.canvas.tag_bind("cable", "<ButtonPress-1>", self._on_cable_press)
        self.canvas.tag_bind("cable", "<B1-Motion>", self._on_cable_motion)
        self.canvas.tag_bind("cable", "<ButtonRelease-1>",
self._on_cable_release)
        self.canvas.tag_bind("cable", "<Button-3>",
self._on_cable_right_click)

        self.status = tk.StringVar(value="")
        ttk.Label(self, textvariable=self.status).grid(row=2, column=0,
sticky="ew")
        self.tip = ttk.Label(self, text="", foreground="#666")
        self.tip.grid(row=3, column=0, sticky="ew")


    def _create_component_section(self, parent_frame, category_name):
        """Create an empty component section for the given category"""
        # Create a scrollable frame for components
        canvas = tk.Canvas(parent_frame, height=300)
        scrollbar = ttk.Scrollbar(parent_frame, orient="vertical",
command=canvas.yview)
        scrollable_frame = ttk.Frame(canvas)

        scrollable_frame.bind(
            "<Configure>",
            lambda e: canvas.configure(scrollregion=canvas.bbox("all"))
        )

        canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
        canvas.configure(yscrollcommand=scrollbar.set)

        # Empty placeholder for now
        empty_label = ttk.Label(
            scrollable_frame,
            text=self._t("components_empty").format(category=category_name),
            font=("Arial", 10),
            foreground="gray",
        )
        empty_label.pack(pady=20, padx=10)

        canvas.pack(side="left", fill="both", expand=True)
        scrollbar.pack(side="right", fill="y")
```

```
    def _add_accordion_section(self, title, expanded=False):
        """Create one collapsible accordion section with empty placeholder
list."""
        # Header
        header_frame = ttk.Frame(self.accordion)
        header_frame.pack(fill="x", expand=False, padx=10, pady=(6, 0))

        # Toggle indicator
        indicator = tk.StringVar(value="-" if expanded else "+")
        indicator_label = ttk.Label(header_frame, textvariable=indicator,
width=2)
        indicator_label.pack(side="left")

        title_label = ttk.Label(header_frame, text=title, font=("Arial", 10,
"bold"))
        title_label.pack(side="left", anchor="w")

        # Body
        body_frame = ttk.Frame(self.accordion)
        if expanded:
            body_frame.pack(after=header_frame, fill="x", expand=False,
padx=10, pady=(2, 6))

        # Empty state content
        empty = ttk.Label(body_frame,
text=self._t("components_empty").format(category=title), foreground="gray")
        empty.pack(fill="x", expand=False, padx=6, pady=8)

        # Toggle logic
        def toggle():
            if body_frame.winfo_manager():
                body_frame.forget()
                indicator.set("+")
            else:
                body_frame.pack(after=header_frame, fill="x", expand=False,
padx=10, pady=(2, 6))
                indicator.set("-")

        header_frame.bind("<Button-1>", lambda e: toggle())
        title_label.bind("<Button-1>", lambda e: toggle())
        indicator_label.bind("<Button-1>", lambda e: toggle())

        self._accordion_sections.append((header_frame, body_frame, indicator))

    def _show_about(self):
        message = self._t("about_message")
        messagebox.showinfo(self._t("about"), message)
```

```
    def _status(self, txt):
        self.status.set(txt)
        self.update_idletasks()

    # Simulation controls
    def start_sim(self):
        self.core.start()
        self._status(self._t("simulation_started"))

    def stop_sim(self):
        self.core.stop()
        self._status(self._t("simulation_stopped"))

    def clear(self):
        self.core.components.clear()
        self.core.connections.clear()
        self.canvas.delete("all")
        self._grid_lines.clear()
        if self._grid_enabled.get():
            self._draw_grid()
        self._status(self._t("canvas_cleared"))

    # Component interactions (placeholder)
    def _on_component_press(self, event):
        self._status(self._t("component_pressed"))

    def _on_component_motion(self, event):
        pass

    def _on_component_release(self, event):
        pass

    def _on_component_right_click(self, event):
        self._status("Component right-clicked")

    def _on_port_press(self, event):
        self._status(self._t("port_pressed"))

    def _on_port_release(self, event):
        self._status(self._t("port_released"))

    def _on_cable_press(self, event):
        self._status(self._t("cable_pressed"))

    def _on_cable_motion(self, event):
        pass

    def _on_cable_release(self, event):
```

```
        pass

    def _on_cable_right_click(self, event):
        self._status(self._t("cable_right_clicked"))

    # Menu command methods
    def _toggle_grid(self):
        if self._grid_enabled.get():
            self._draw_grid()
        else:
            self._clear_grid()

    def _clear_grid(self):
        for line_id in self._grid_lines:
            self.canvas.delete(line_id)
        self._grid_lines.clear()

    def _draw_grid(self, spacing: int = 20, color: str = "#eef3f8"):
        self._clear_grid()
        # Get current visible region
        x0 = int(self.canvas.canvasx(0))
        y0 = int(self.canvas.canvasy(0))
        x1 = int(self.canvas.canvasx(self.canvas.winfo_width()))
        y1 = int(self.canvas.canvasy(self.canvas.winfo_height()))
        # Snap start to spacing
        start_x = (x0 // spacing) * spacing
        start_y = (y0 // spacing) * spacing
        for x in range(start_x, x1 + 1, spacing):
            line = self.canvas.create_line(x, y0, x, y1, fill=color)
            self._grid_lines.append(line)
        for y in range(start_y, y1 + 1, spacing):
            line = self.canvas.create_line(x0, y, x1, y, fill=color)
            self._grid_lines.append(line)
        # Send grid to back
        for line_id in self._grid_lines:
            self.canvas.tag_lower(line_id)
    def _new_file(self):
        self._status(self._t("new_file_clicked"))
        messagebox.showinfo(self._t("info"), self._t("new_file_info"))

    def _open_file(self):
        self._status(self._t("open_file_clicked"))
        messagebox.showinfo(self._t("info"), self._t("open_file_info"))

    def _save_file(self):
        self._status(self._t("save_file_clicked"))
        messagebox.showinfo(self._t("info"), self._t("save_file_info"))
```

```
    def _save_as_file(self):
        self._status(self._t("save_as_clicked"))
        messagebox.showinfo(self._t("info"), self._t("save_as_info"))

    def _exit_app(self):
        self._status(self._t("exit_application"))
        self.master.quit()

    def _undo(self):
        self._status(self._t("undo_clicked"))
        messagebox.showinfo(self._t("info"), self._t("undo_info"))

    def _redo(self):
        self._status(self._t("redo_clicked"))
        messagebox.showinfo(self._t("info"), self._t("redo_info"))

    def _cut(self):
        self._status(self._t("cut_clicked"))
        messagebox.showinfo(self._t("info"), self._t("cut_info"))

    def _copy(self):
        self._status(self._t("copy_clicked"))
        messagebox.showinfo(self._t("info"), self._t("copy_info"))

    def _paste(self):
        self._status(self._t("paste_clicked"))
        messagebox.showinfo(self._t("info"), self._t("paste_info"))

    def _show_preferences(self):
        self._status(self._t("preferences_clicked"))
        messagebox.showinfo(self._t("preferences"),
self._t("preferences_info"))

    # i18n helpers
    def _languages_dir(self) -> str:
        return self._resource_path("languages")

    def _config_path(self) -> str:
        return os.path.join(self._languages_dir(), "config.json")

    def _ensure_default_languages(self) -> None:
        os.makedirs(self._languages_dir(), exist_ok=True)
        defaults = {
            # Only ensure files exist by copying from languages directory; do
not embed strings here
            # This method will just make sure base files exist by copying our
bundled JSONs if needed.
        }
```

```
        # We already added JSON files into the languages directory via the
project; nothing to generate here.
        # Ensure config exists
        if not os.path.exists(self._config_path()):
            try:
                with open(self._config_path(), "w", encoding="utf-8") as f:
                    json.dump({"selected_language": "tr"}, f,
ensure_ascii=False, indent=2)
            except Exception:
                pass

    def _load_saved_language(self) -> str:
        # Ensure defaults present on first run
        self._ensure_default_languages()
        try:
            with open(self._config_path(), "r", encoding="utf-8") as f:
                data = json.load(f)
                code = data.get("selected_language")
                if isinstance(code, str) and code.strip():
                    return code
        except Exception:
            pass
        return "tr"

    def _save_selected_language(self, code: str) -> None:
        try:
            os.makedirs(self._languages_dir(), exist_ok=True)
            with open(self._config_path(), "w", encoding="utf-8") as f:
                json.dump({"selected_language": code}, f, ensure_ascii=False,
indent=2)
        except Exception:
            pass

    def _load_translations(self, code: str) -> dict:
        path = os.path.join(self._languages_dir(), f"{code}.json")
        try:
            with open(path, "r", encoding="utf-8") as f:
                data = json.load(f)
                if isinstance(data, dict):
                    return data
        except Exception:
            pass
        # Fallback to Turkish file
        try:
            with open(os.path.join(self._languages_dir(), "tr.json"), "r",
encoding="utf-8") as f:
                data = json.load(f)
                if isinstance(data, dict):
```

```
                    return data
        except Exception:
            pass
        # Last resort: minimal inline fallback
        return {"code": "tr", "name": "Türkçe", "title": "Elektrik Devre
Simülasyonu"}

    def _t(self, key: str) -> str:
        return str(self.translations.get(key, key))

    def _list_available_languages(self):
        langs = []  # list of (code, name)
        try:
            os.makedirs(self._languages_dir(), exist_ok=True)
            for fname in os.listdir(self._languages_dir()):
                if not fname.lower().endswith(".json"):
                    continue
                fpath = os.path.join(self._languages_dir(), fname)
                try:
                    with open(fpath, "r", encoding="utf-8") as f:
                        data = json.load(f)
                    code = data.get("code") or os.path.splitext(fname)[0]
                    name = data.get("name") or code
                    if code != "config":
                        langs.append((code, name))
                except Exception:
                    continue
        except Exception:
            pass
        # Ensure unique by code
        seen = set()
        unique = []
        for code, name in langs:
            if code in seen:
                continue
            seen.add(code)
            unique.append((code, name))
        unique.sort(key=lambda x: x[1].lower())
        return unique

    def _populate_language_menu(self):
        try:
            self.menu_language.delete(0, "end")
        except Exception:
            return
        for code, name in self._list_available_languages():
            self.menu_language.add_radiobutton(
                label=name,
```

```
                variable=self._language_var,
                value=code,
                command=self._on_language_selected,
            )
        self.menu_language.add_separator()
        self.menu_language.add_command(label="+ Add language…",
command=self._import_language)

    def _on_language_selected(self):
        selected = self._language_var.get()
        if selected == getattr(self, "lang_code", None):
            return
        self.lang_code = selected
        self.translations = self._load_translations(self.lang_code)
        self._save_selected_language(self.lang_code)
        self._rebuild_ui()

    def _import_language(self):
        self._status(self._t("import_language_clicked"))
        src = filedialog.askopenfilename(
            title=self._t("select_language_json"),
            filetypes=[("JSON files", "*.json")],
        )
        if not src:
            return
        try:
            with open(src, "r", encoding="utf-8") as f:
                data = json.load(f)
            code = data.get("code")
            name = data.get("name")
            if not code or not isinstance(code, str):
                messagebox.showerror(self._t("error"),
self._t("invalid_language_file"))
                return
            os.makedirs(self._languages_dir(), exist_ok=True)
            dst = os.path.join(self._languages_dir(), f"{code}.json")
            shutil.copyfile(src, dst)
            self._populate_language_menu()
            messagebox.showinfo(self._t("info"),
self._t("language_imported").format(name=name or code))
        except Exception as e:
            messagebox.showerror(self._t("error"), str(e))

    def _rebuild_ui(self):
        try:
            self.master.config(menu=None)
        except Exception:
            pass
```

```
        for child in self.winfo_children():
            try:
                child.destroy()
            except Exception:
                pass
        self._grid_lines = []
        self._accordion_sections = []
        self._build_ui()

    def _change_language(self):
        self._status("Change language clicked")
        messagebox.showinfo("Dil", "Dil değiştirme işlemi")

    def _print_canvas(self):
        fname = filedialog.asksaveasfilename(defaultextension=".ps",
filetypes=[("PostScript", "*.ps")])
        if not fname:
            return
        try:
            self.canvas.postscript(file=fname, colormode='color')
            self._status(self._t("canvas_exported"))
        except Exception as e:
            messagebox.showerror(self._t("error"), str(e))
```

**NOTLAR:**

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Avrupa Birliği tarafından finanse edilmektedir

**NOTLAR:**

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY

Avrupa Birliği tarafından finanse edilmektedir

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Avrupa Birliği tarafından finanse edilmektedir