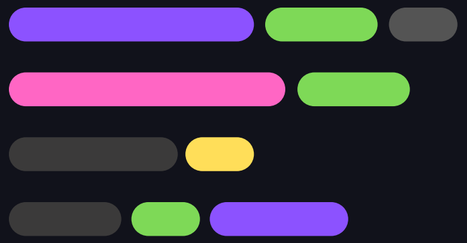
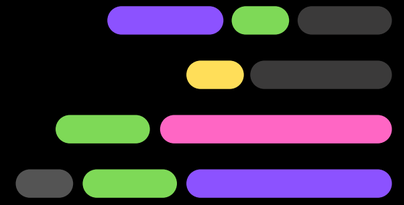
{TEMEL PROGRAMLAMA II}

Samsun Üniversitesi
Teknik Bilimler Meslek Yüksekokulu
Arka-Yüz Yazılım Geliştirme Pr.



Öğr. Gör. Furkan DURMUŞ

Bu Haftanın Ders Kazanımları



01 Debug Kavramı

02 Hata Yakalama

03 Hata Türleri

Debug Kavramı

01

Debug

Debug Nedir ?

Debug, programdaki hatayı bulma, nedenini anlama ve düzeltme sürecidir.

01

Debug

Debug sadece hata mesajı çıkan durumlar için değildir.
Program çalışıp yanlış sonuç veriyorsa da debug yapılır.

01

Debug

Debug sayesinde :

- Kodun satır satır nasıl çalıştığını görülür.

01

Debug

Debug sayesinde :

- Kodun satır satır nasıl çalıştığını görülür.
- Değişkenlerin değerlerini takip edilebilir.

01

Debug

Debug sayesinde :

- Kodun satır satır nasıl çalıştığını görülür.
- Değişkenlerin değerlerini takip edilebilir.
- Hatanın hangi satırda başladığını bulunur.

01

Debug

Debug sayesinde :

- Kodun satır satır nasıl çalıştığını görülür.
- Değişkenlerin değerlerini takip edilebilir.
- Hatanın hangi satırda başladığını bulunur.
- Tahmin etmek yerine gözlem yapılır.

01

Debug

Debug sayesinde :

- Kodun satır satır nasıl çalıştığını görülür.
- Değişkenlerin değerlerini takip edilebilir.
- Hatanın hangi satırda başladığını bulunur.
- Tahmin etmek yerine gözlem yapılır.
- Mantıksal hataları daha kolay yakalanır.

01

Debug

VS Code, Python için breakpoint koyma, değişkenleri inceleme ve kodu adım adım çalıştırma gibi debug özellikleri sunar. Python tarafında bunun için Python Debugger extension kullanılır; Python extension ile birlikte gelir.

01

Debug

Yandaki kodu bilgisayarınızda çalıştırınız ve bug'ı mantıken bulunuz.

```
mail = input("E-posta giriniz: ")

if "@" and "." in mail:
    print("Geçerli e-posta")
else:
    print("Geçersiz e-posta")
```

01

Debug

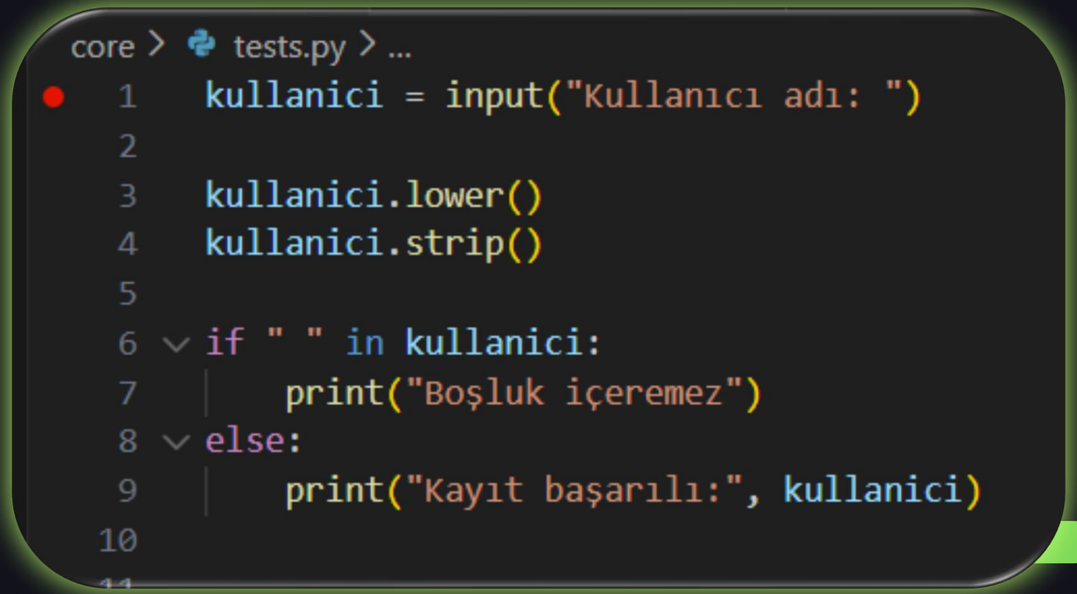
Yandaki kodu çalıştırınız.

```python
kullanici = input("Kullanıcı adı: ")

kullanici.lower()
kullanici.strip()

if " " in kullanici:
    print("Boşluk içeremez")
else:
    print("Kayıt başarılı:", kullanici)
```

01

Debug

Debug'ın nereden başlayacağını belirlemek için satır sayısının yanındaki boşluğa tıklanmalı ve kırmızı nokta görülmelidir. Bu kırmızı noktaya **BreakPoint** denir.



01

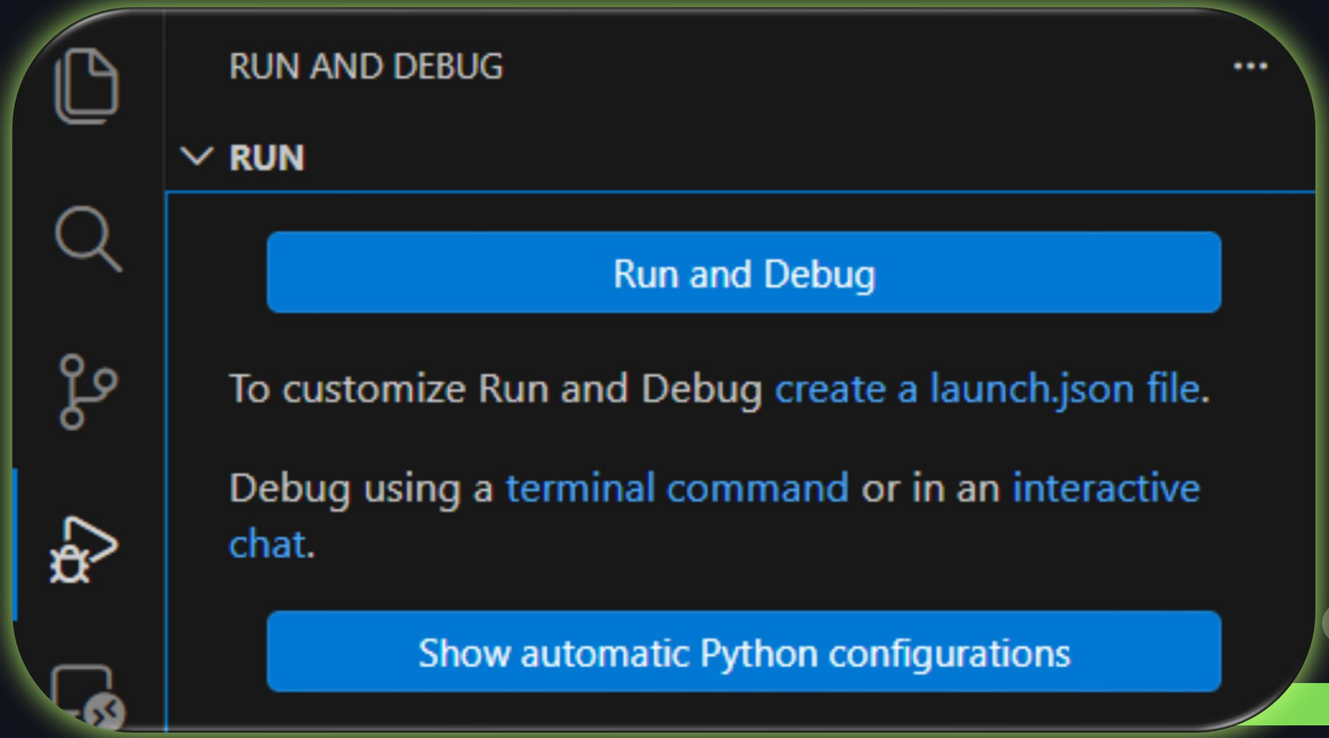
Debug

Debug modu açmak için önce sol panelde yanda gözüken ikona tıklayınız.

01

Debug

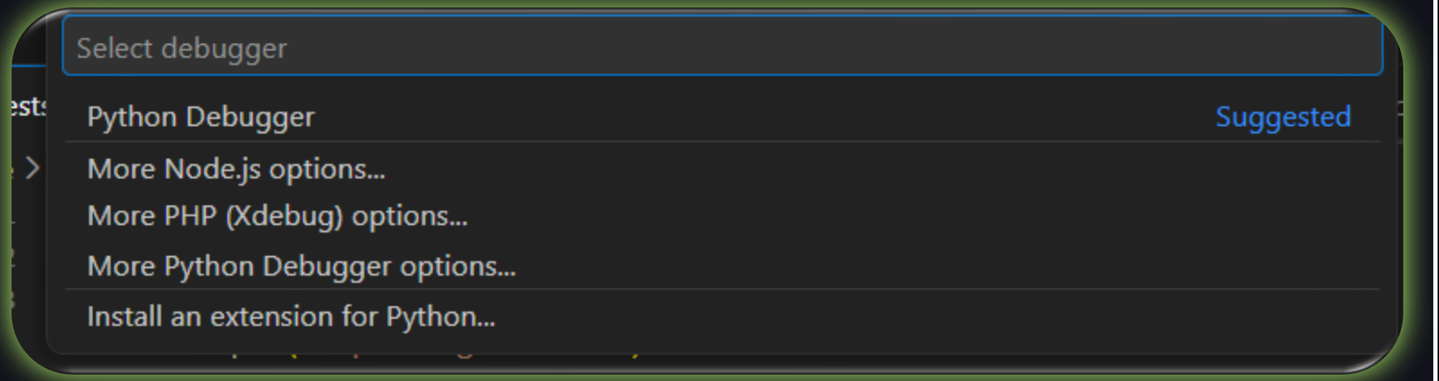
Açılan pencerede **run and debug** butonuna tıklayınız.



01

Debug

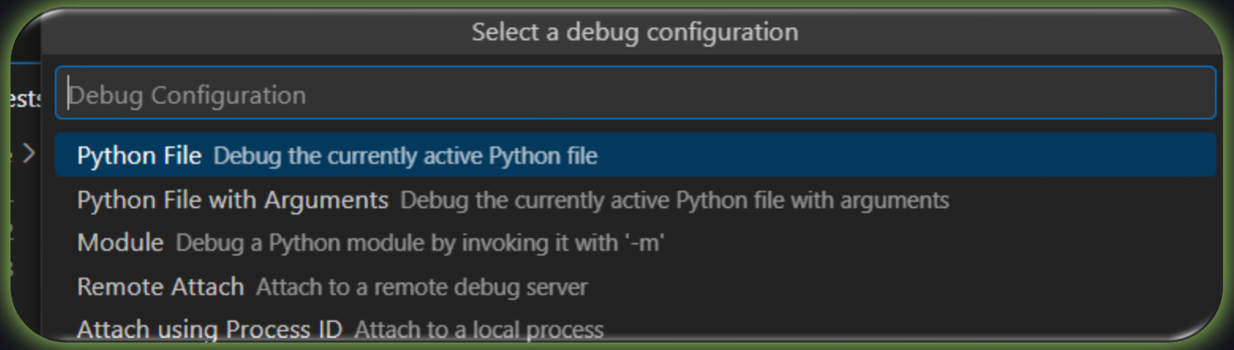
Ardından üst tarafta açılan menüden **Python Debugger** seçiniz.



01

Debug

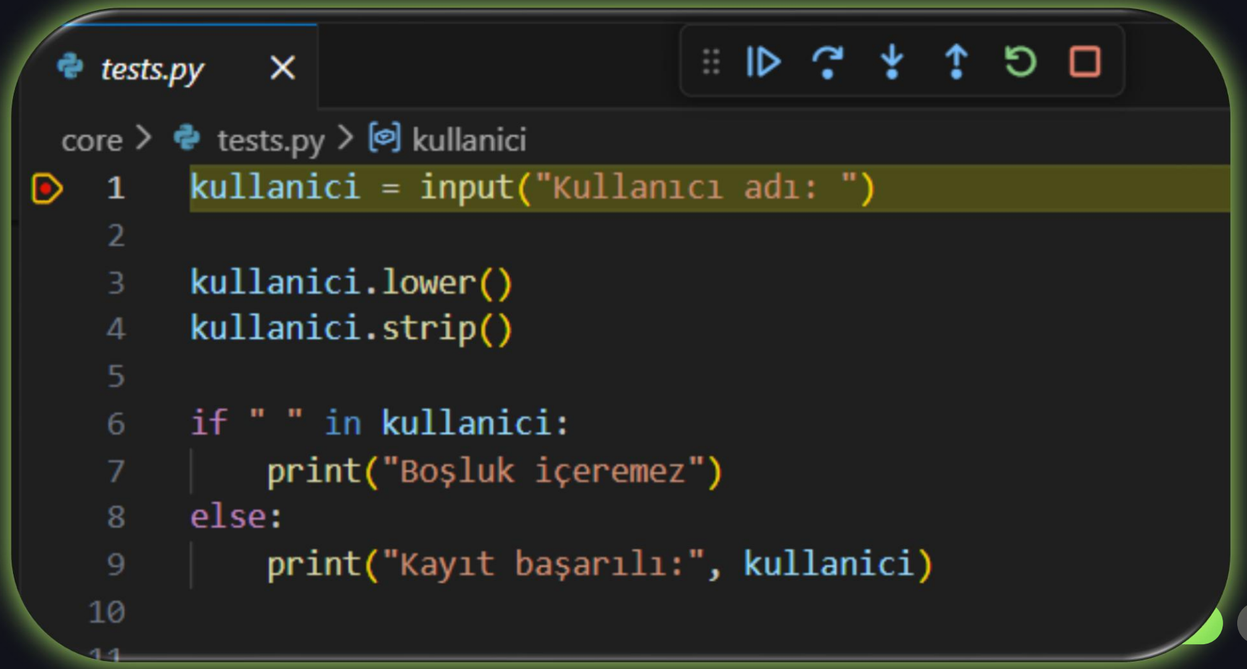
Açılan yeni menüden **Python File** seçiniz.



01

Debug

Yandaki görüntüyü elde ediniz.



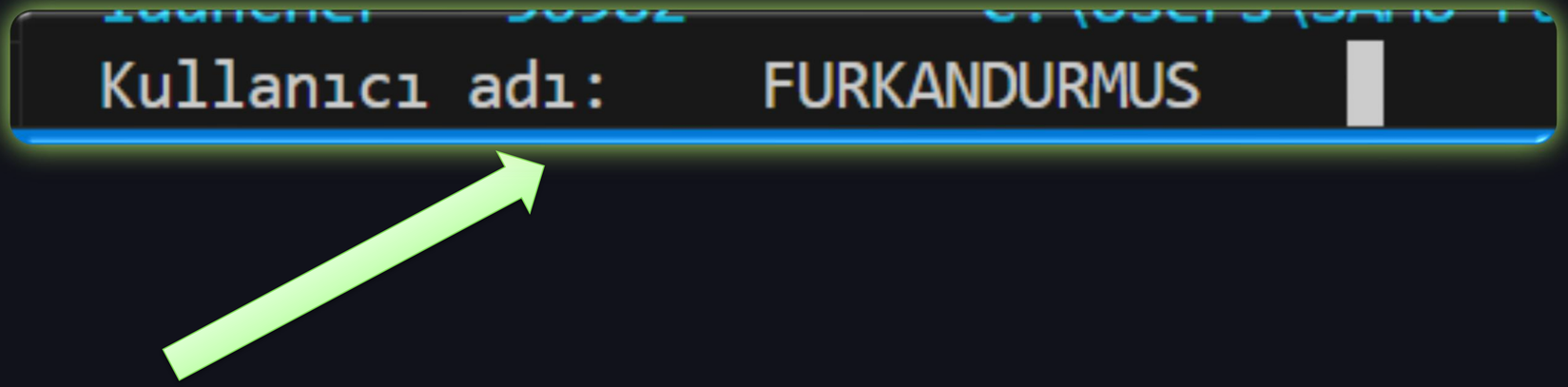
01

Debug

Üstte açılan menüdeki Step over yada F10 ile kodu satır satır ilerletebilirsiniz.

01

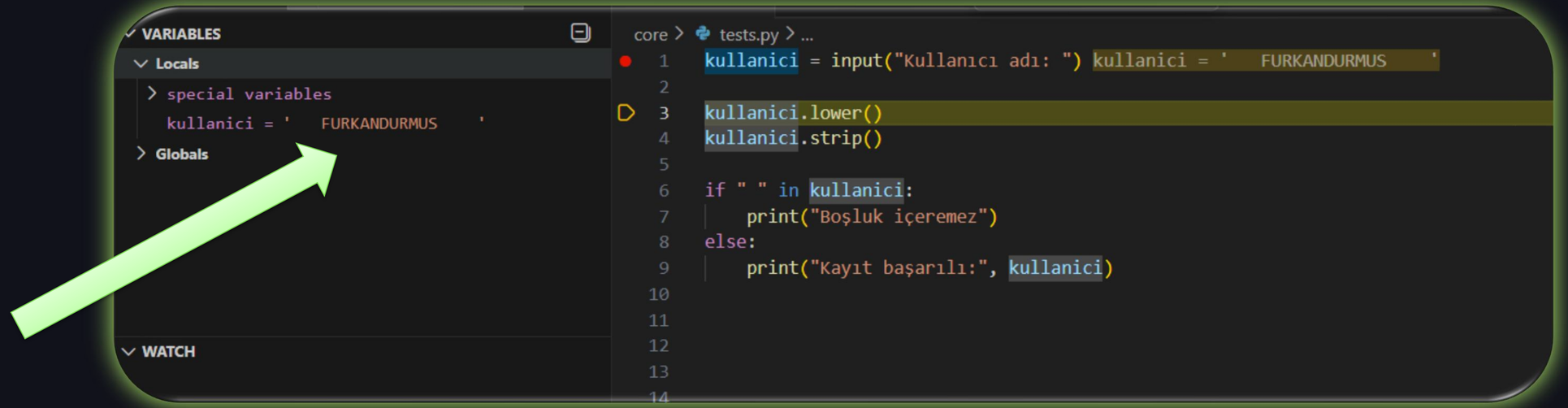
Debug



Adınızı başında ve sonunda boşluklar bırakarak büyük harfle input olarak giriniz.

01

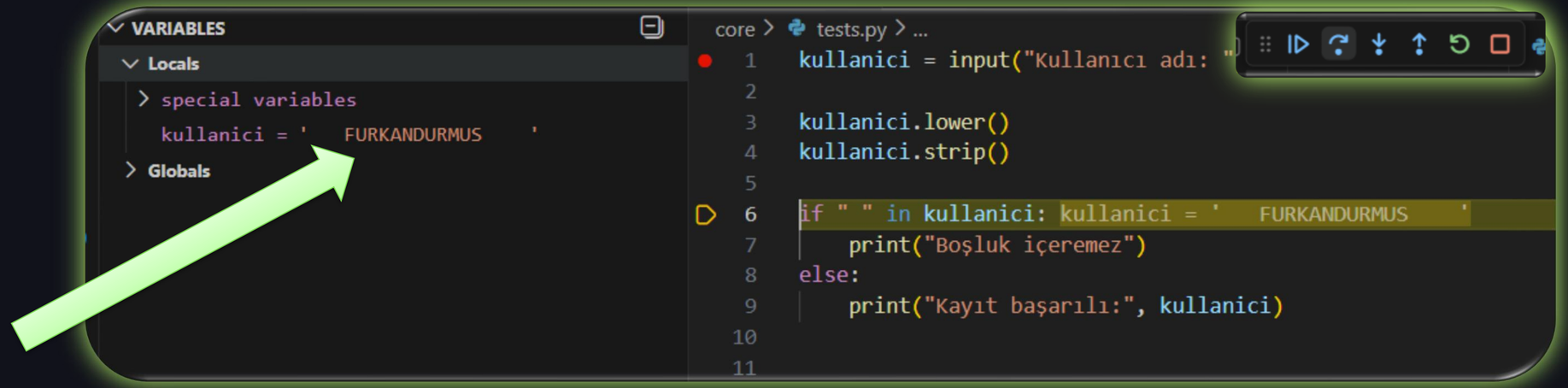
Debug



Kullanici değişkeni sol tarafta local variables'in içine eklendi. Step over yani ilerle butonuna tıkladıkça kullanici değişkenindeki değişim gözlemlenebilir.

01

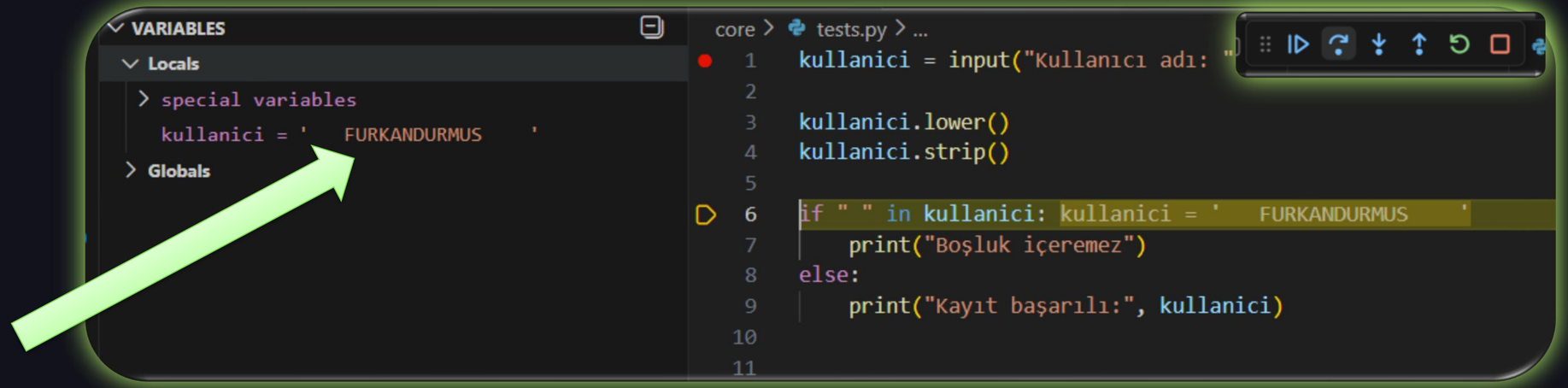
Debug



Kod 6. satıra gelmesine rağmen kullanici değişkeni lower ve stripten etkilenmedi. Sebebi ne olabilir ?

01

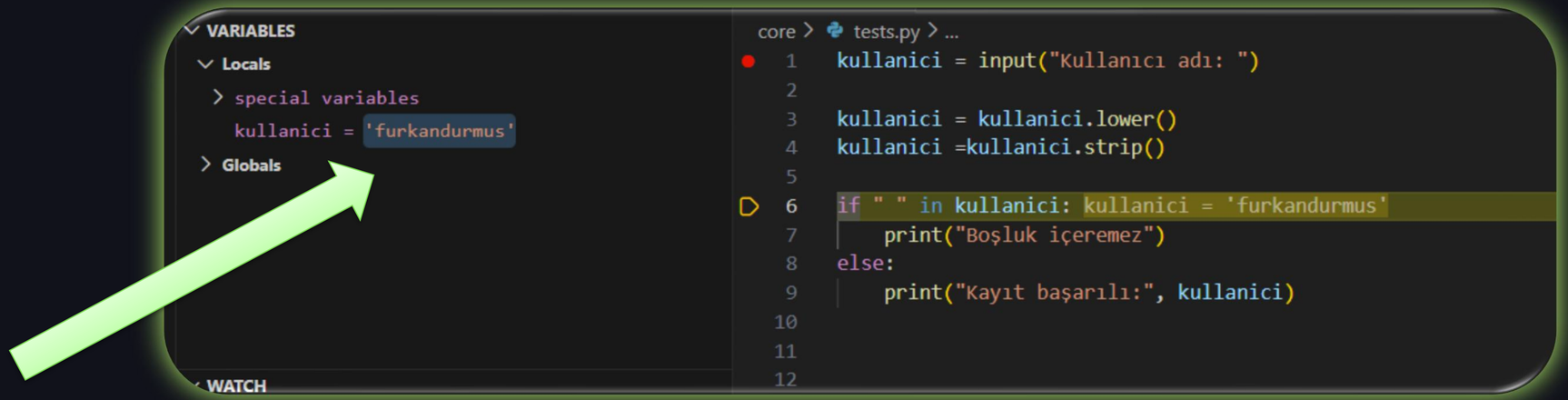
Debug



Değişkenin güncellenip tekrar kendine atanması yapılmadığı için kullanici güncellenmedi. Debug ile hatalı noktayı bulmuş olduk.

01

Debug



Sizde kodu üstteki gibi düzeltip debug yaparak değişkenin güncellendiğini gözlemleyiniz.

01

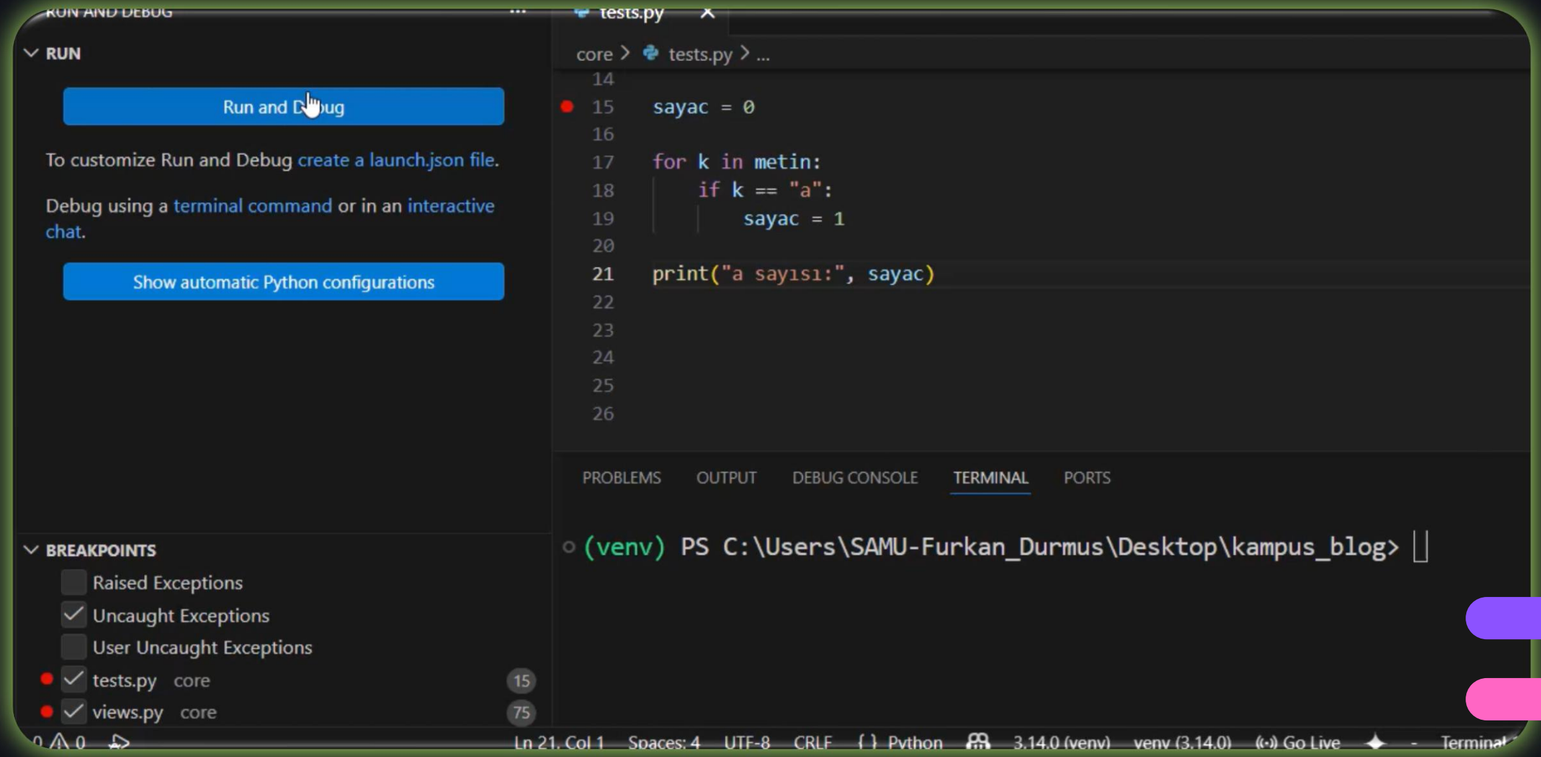
Debug

Yakdaki kodu debug ile inceleyiniz.

```python
metin = input("Metin: ")

sayac = 0

for k in metin:
    if k == "a":
        sayac = 1

print("a sayısı:", sayac)
```

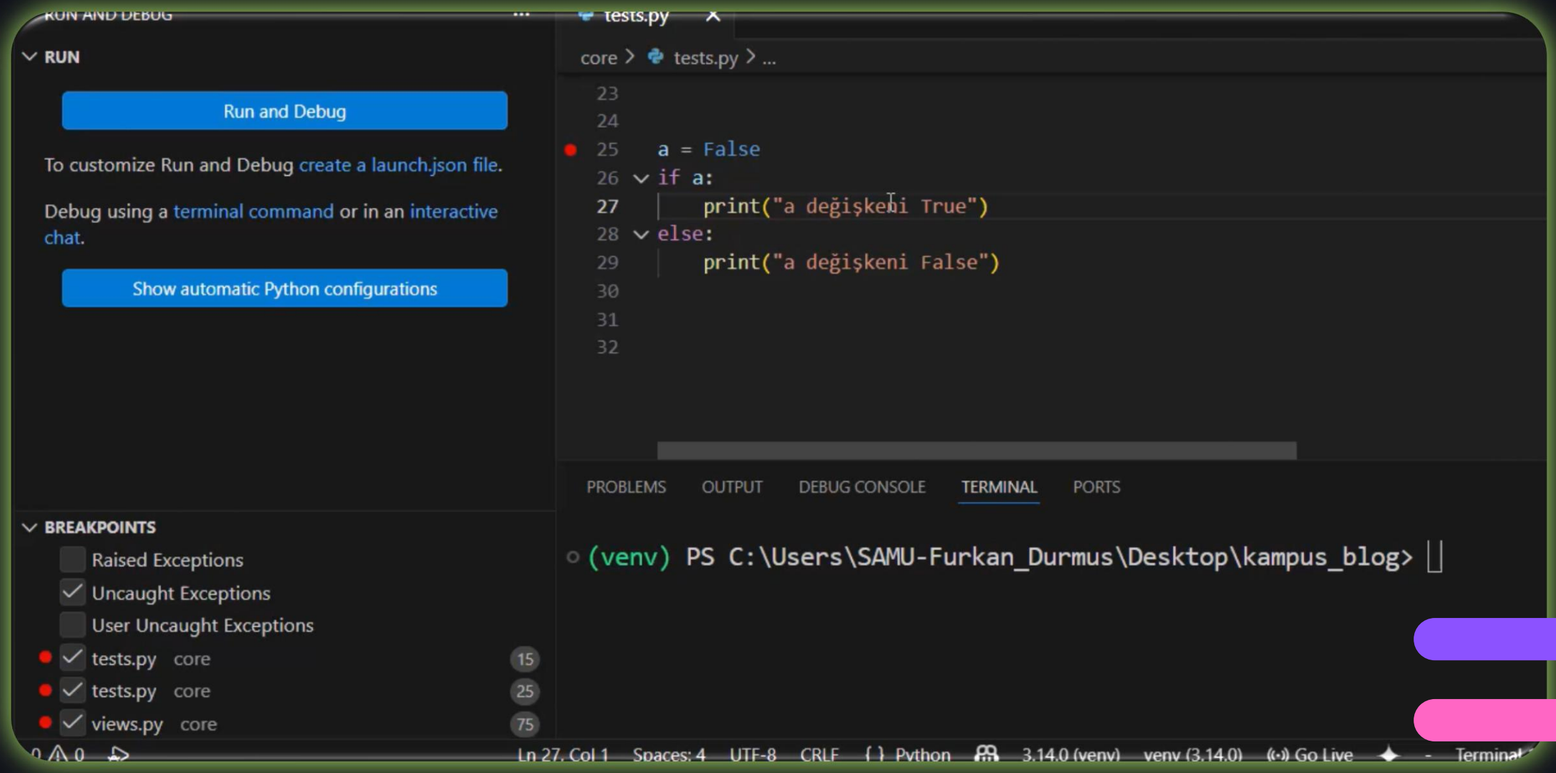
01

Debug


RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Debug using a terminal command or in an interactive chat.
Show automatic Python configurations
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
tests.py core 15
views.py core 75
tests.py
core > tests.py > ...
14
15 sayac = 0
16
17 for k in metin:
18 if k == "a":
19 sayac = 1
20
21 print("a sayısı:", sayac)
22
23
24
25
26
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(venv) PS C:\Users\SAMU-Furkan_Durmus\Desktop\kampus_blog>
Ln 21, Col 1 Spaces: 4 UTF-8 CRLF { } Python 3.14.0 (venv) venv (3.14.0) Go Live Terminal

01

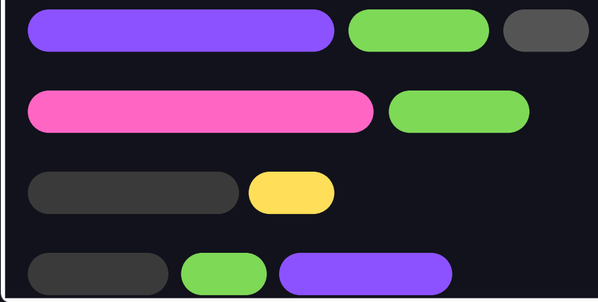
Debug


RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Debug using a terminal command or in an interactive chat.
Show automatic Python configurations
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
tests.py core 15
tests.py core 25
views.py core 75
tests.py
core > tests.py > ...
23
24
25 a = False
26 if a:
27 print("a değişkeni True")
28 else:
29 print("a değişkeni False")
30
31
32
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
(venv) PS C:\Users\SAMU-Furkan_Durmus\Desktop\kampus_blog>
Ln 27, Col 1
Spaces: 4
UTF-8
CRLF
{} Python
3.14.0 (venv)
venv (3.14.0)
Go Live
Terminal

01

Debug

Yakdaki kodu debug ile inceleyiniz.



```python
en_buyuk = 0

for i in [-5, -2, -10]:
    if i > en_buyuk:
        en_buyuk = i

print(en_buyuk)
```

Hata Yakalama

02

Hata Yakalama

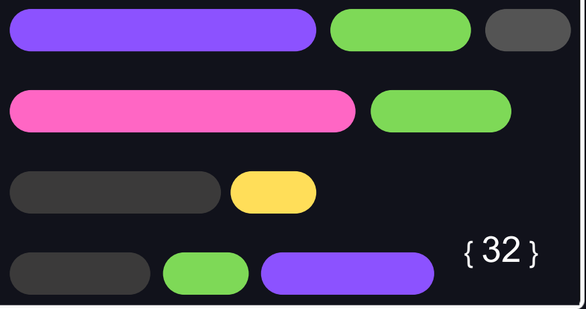
Python programlama dilinde hata yakalama **try-except** blokları aracılığıyla yapılmaktadır.

02

Hata Yakalama

En temel try-except bloğu şu şekildedir:

```
try:
    # hata oluşması muhtemel kod bloğu

except:

    # hata durumunda yapılacak işlemler
```



02

Hata Yakalama

```
try:
    # hata oluşması muhtemel kod bloğu
except:
        # hata durumunda yapılacak işlemler
```

try bloğunda, hata oluşma ihtimali bulunan kodlar yazılır ve eğer bir hata oluşursa **except** bloğu devreye girer. Hata oluştuğunda except bloğunun devreye girmesi “Hatayı yakalama” olarak adlandırılır. Hata olmazsa program çalışmaya except bloğundan sonraki satırdan devam edecektir.

02

Hata Yakalama

Yandaki kodu çalıştırıp hata almaya çalışınız.

```
try:
    sayi = int(input("Sayı: "))
    print("Dönüştürme başarılı.")
    sonuc = 100 / sayi
    print("Sonuç:", sonuc)
except:
    print("Bir hata oluştu.")
```

02

Hata Yakalama

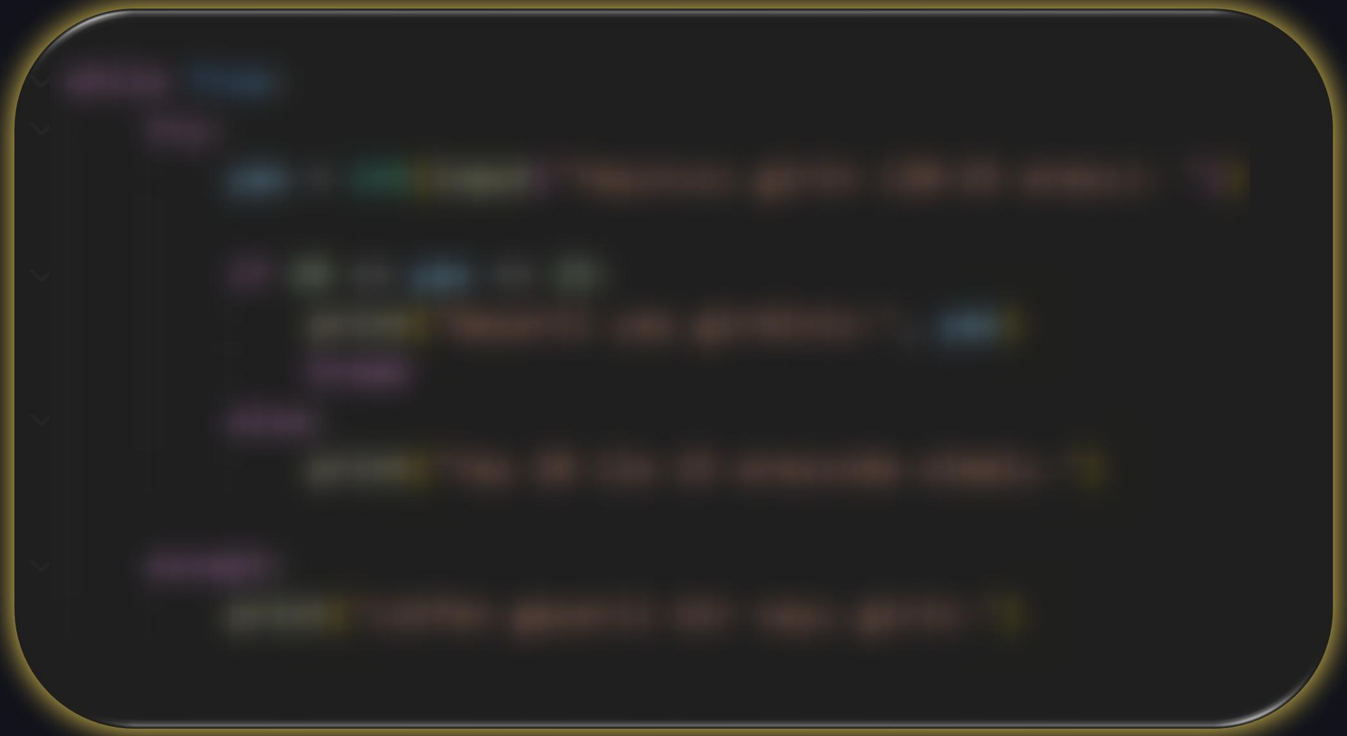
Görüldüğü üzere, program hata mesajı vererek sonlanmadı. Program, programcının kontrolü altında çalışmaya devam etti. Programcı burada, istisnai bir durum karşısında programının yarıda kesilmesini engellemiştir.

```
try:
    sayi = int(input("Sayı: "))
    print("Dönüştürme başarılı.")
    sonuc = 100 / sayi
    print("Sonuç:", sonuc)
except:
    print("Bir hata oluştu.")
```

02

Hata Yakalama

Kullanıcıdan 20 ila 25 arasında sayı girene kadar hata vermeden input alan kodu yazınız.



02

Hata Yakalama

Kullanıcıdan 20 ila 25 arasında sayı girene kadar hata vermeden input alan kodu yazınız.

```python
while True:
    try:
        yas = int(input("Yaşınızı girin (20-25 arası): "))

        if 20 <= yas <= 25:
            print("Geçerli yaş girdiniz:", yas)
            break
        else:
            print("Yaş 20 ile 25 arasında olmalı.")

    except:
        print("Lütfen geçerli bir sayı girin.")
```

02

Hata Yakalama

İçerisinde 5 isim bulunan bir liste oluşturunuz. Kullanıcıdan index'i alın ve o index'te bulunan ismi ekrana yazıdırın.

```
isimler = ["Ali", "Ayşe", "Mehmet", "Fatma", "Ahmet"]
```

02

Hata Yakalama

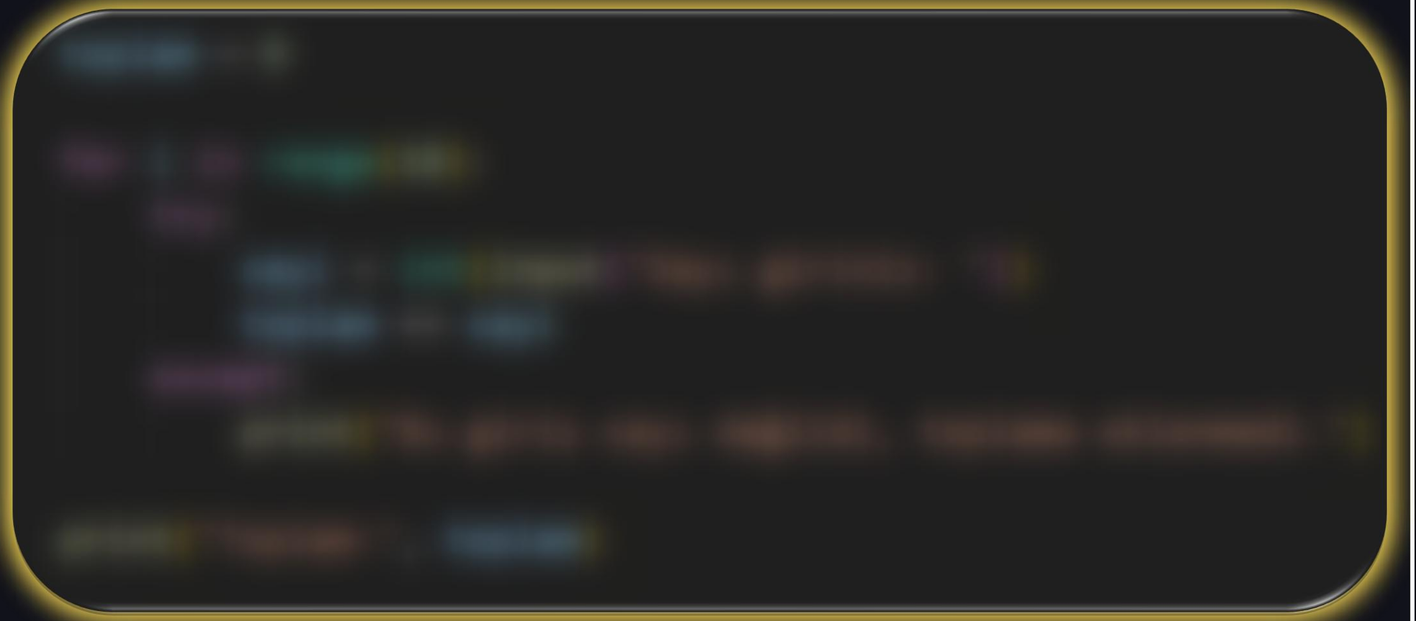
İçerisinde 5 isim bulunan bir liste oluşturunuz. Kullanıcıdan index'i alın ve o index'te bulunan ismi ekrana yazıdırın.

```python
isimler = ["Ali", "Ayşe", "Mehmet", "Fatma", "Ahmet"]

try:
    index = int(input("Kaçıncı kişiyi görmek istiyorsunuz?: "))
    print(isimler[index])
except:
    print("Geçersiz işlem yaptınız.")
```

02

Hata Yakalama

Kullanıcıdan 10 adet veri alın. Girilen değerlerden sayısal olanları toplayın; sayı olmayan girişlerde hata mesajı vererek kullanıcıdan tekrar sayı girmesini isteyin.



02

Hata Yakalama

Kullanıcıdan 10 adet veri alın. Girilen değerlerden sayısal olanları toplayın; sayı olmayan girişlerde hata mesajı vererek kullanıcıdan tekrar sayı girmesini isteyin.

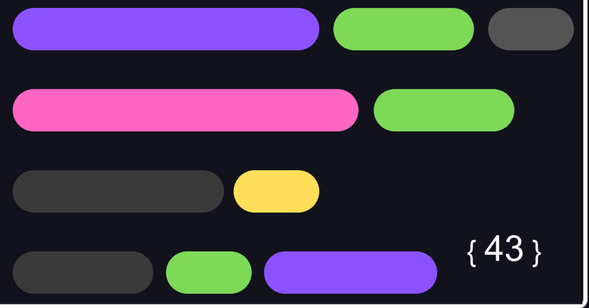
```python
toplam = 0

for i in range(10):
    try:
        sayi = int(input("Sayı giriniz: "))
        toplam += sayi
    except:
        print("Bu giriş sayı değildi, toplama eklenmedi.")

print("Toplam:", toplam)
```

Hata Türleri

03

Hata Türleri

Python’da oluşan her hata bir türe sahiptir.



03

Hata Türleri

1. ValueError

Değer uygun değilse oluşur.

```
sayi = int("abc")
```

03

Hata Türleri

1. ValueError

Değer uygun değilse oluşur.

```
try:
    sayi = int("abc")
except ValueError:
    print("Geçersiz sayı girdiniz.")
```

03

Hata Türleri

2. TypeError

Yanlış veri tipiyle işlem yapılırsa oluşur.

```
liste = [1, 2, 3]
print(liste + 5)
```

03

Hata Türleri

2. TypeError

Yandaki kodu çalıştırınız ve test ediniz.

```
try:
    sayi = int(input("Sayı: "))
    sayi = sayi + [1, 2, 3]

except ValueError:
    print("Geçersiz sayı girdiniz.")
except TypeError:
    print("Sayı ile liste toplanamaz.")
```

03

Hata Türleri

Not : Bir try bloğunun birden fazla except bloğu **olabilir**. Böylece try içerisindeki hataya özel mesaj ekrana yazdırabiliriz.

```
try:
    sayi = int(input("Sayı: "))
    sayi = sayi + [1, 2, 3]

except ValueError:
    print("Geçersiz sayı girdiniz.")
except TypeError:
    print("Sayı ile liste toplanamaz.")
```

03

Hata Türleri

3.ZeroDivisionError

Sıfıra bölme hatası.

```
sayi = 10 / sayi
```

03

Hata Türleri

3.ZeroDivisionError

Yandaki kodu çalıştırınız.

```
try:
    sayi = int(input("Sayı: "))
    sayi = 10 / sayi

except ValueError:
    print("Geçersiz sayı girdiniz.")
except ZeroDivisionError:
    print("Sıfıra bölme hatası.")
except:
    print("Beklenmeyen bir hata oluştu.")
```

03

Hata Türleri

4. IndexError

Olmayan indexe erişme hatası.

```
liste = [1, 2, 3]

print(liste[5])
```

03

Hata Türleri

5. KeyError

Dictionary'de olmayan anahtara erişme hatası.

```
sozluk = {"ad": "Ahmet"}

print(sozluk["yas"])
```

Son...