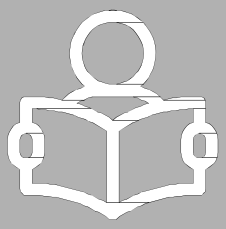
PYTHON

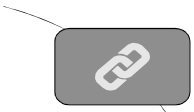
Hata Yakalama ve İstisnalar

Kapsam



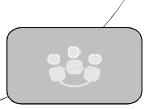
- Hata Kavramı
- Hata Yakalama
- Hata Türleri
- Try-Except Blokları

Hata Kavramı



Kodları yazarken Python, birtakım hata mesajları verir. Bu hata mesajları programda yapılan söz dizimi (syntax) hatalarının görülmesini sağlar.

Programcılar her durumu düşünüp hataları önceden belirlemelidir.



Hata mesajlarını kullanıcıların görmesi istenmediğinde Python, olası hata oluşumlarına karşı önlemler alınmasını sağlayan ifadeler sunar.

Hata Yakalama

Beklenmedik durumlarda programın bir hata mesajı vermesi ve çalışmayı durdurması yerine, hataya kullanıcının istediği şekilde cevap vermesini sağlamanın bir yolu olarak adlandırılmaktadır.
Hata yakalama Python programlama dilinin önemli bir parçasıdır ve kaynak kodunu çok karışık hale getirmeden programınızın güvenilir bir şekilde çalışmasını sağlar.

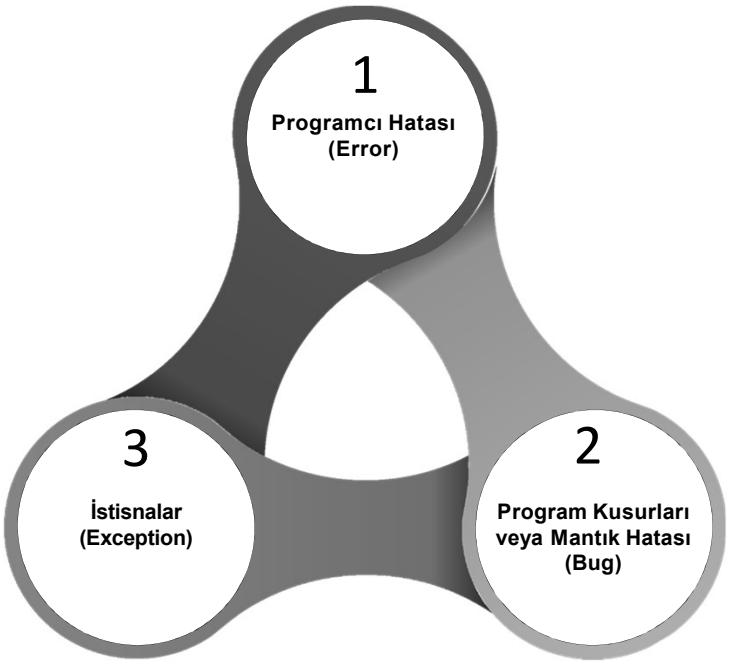
Kullanıcıdan sayılar alan ve aldığı sayıların 2 katını ekrana yazan ve boş satır okuduğunda program sonlandırılmasını sağlayan bir uygulama yapalım.

Kod

```
while True:
    x = input("Bir sayı girin: ")
    if not x:
        break
    print(float(x)*2)
```

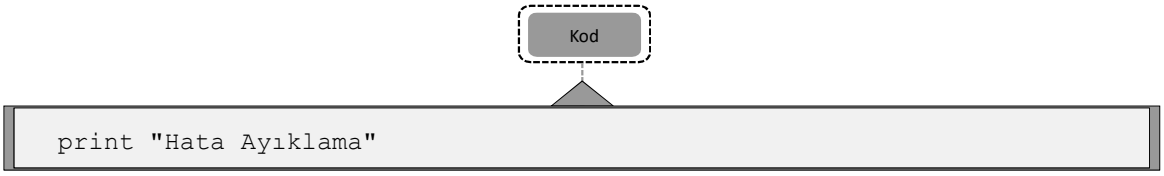
Örnek'te girilen bir string ifade sayıya dönüştürülemediği için float() fonksiyonu bir ValueError hatası (Python terimiyle "exception") verir.
Böyle hatalar programın çalışmasını durdurur.
Oysa, bir hata yakalama (exception handling) yapısı kullanılır ise bu tür sorunları programı durdurmadan halletmek mümkün olmaktadır.

Hata Türleri



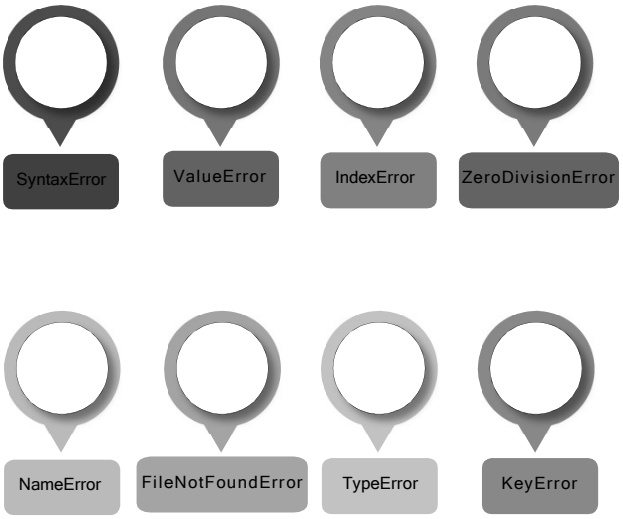
Programcı Hatası

Geliştiricilerden kaynaklanan bir hatadır. Programcının syntax yazım yanlışı vb. yaptığı hatalardır. Bu hata giderilmeden program hiçbir şekilde çalışmaz. Bu tür hataların hangi satırda ve hangi kodlardan dolayı yapıldığı bellidir. Bu yüzden fark edilmesi ve çözülmesi kolaydır.



Örnek'te print parantez içine alınmadığı için kullanıcıdan kaynaklı SyntaxError hatası verir.
Eksik parantez var, diye mesaj verir.

Python'da sıklıkla karşılaşılan hatalar



Python'da olabilecek bütün hata mesajlarına https://docs.python.org/2/library/exceptions.html adresinden bakabilirsiniz.

Python'da sıklıkla karşılaşılan hatalar

SyntaxError

- Syntax hataları, tüm programlama dillerinde olan ve yazım hatası anlamına gelen hata kodlarıdır.
- Bilgisayar programları her ne kadar çok hızlı çalışan ve zeki yapılar olsalar da maalesef ne yapacaklarını anlayabilmeleri için bizim onlara doğru komutları (kodları) yazmamızı beklerler, bu yazım içerisinde o dilin yazım kuralları tarafından tanımlanmamış bir kelime, harf ya da noktalama işareti ile karşılaştıklarında da syntax error (yazım hatası) üretirler.

Örnek

```
print 1/2
  File "<ipython-input-5-274676b61d18>", line 1
    print 1/2
          ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(1/2)?
```

Python'da sıklıkla karşılaşılan hatalar

ValueError

- Genelde kullanıcıdan input istediğimiz kodlarda karşılaştığımız bir hata çeşididir. Siz kullanıcıdan integer bir input isteyerek kodunuzu yazarsınız, kullanıcı integer dışında bir şeyi input olarak girer ve bu hatayla karşılaşır.

Örnek

```
Lütfen Yarıçap Giriniz. Ankara
Traceback (most recent call last):

  File "<ipython-input-6-3d5076312e7d>", line 1, in <module>
    yaricap = int(input("Lütfen Yarıçap Giriniz."))

ValueError: invalid literal for int() with base 10: 'Ankara'
```

Python'da sıklıkla karşılaşılan hatalar

IndexError

- Python'da bir dizi değişkenin (liste, demet ya da metin (string)) uzunluğunun ötesinde bir elemana ulaşmaya çalışılınca bu hata oluşur.
- Bilgisayar programları her ne kadar çok hızlı çalışan ve zeki yapılar olsalar da maalesef ne yapacaklarını anlayabilmeleri için bizim onlara doğru komutları (kodları) yazmamızı beklerler, bu yazım içerisinde o dilin yazım kuralları tarafından tanımlanmamış bir kelime, harf ya da noktalama işareti ile karşılaştıklarında da syntax error (yazım hatası) üretirler.

Örnek

```
veri = [i for i in range(1,10)]
veri[9]
Traceback (most recent call last):

  File "<ipython-input-9-d694c97c3f64>", line 1, in <module>
    veri[9]

IndexError: list index out of range
```

Python'da sıklıkla karşılaşılan hatalar

ZeroDivisionError

- 0 ile bölme bu hatayı tetikler.

Örnek

```
Traceback (most recent call last):

  File "<ipython-input-10-f6cc6d14333b>", line 1, in <module>
    3.0/0

ZeroDivisionError: division by zero
```

Python'da sıklıkla karşılaşılan hatalar

NameError

- Tanımlanmamış bir değişken ismine ulaşılmaya çalışıldığında meydana gelir.

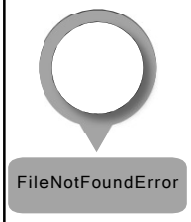
Örnek

```
Traceback (most recent call last):

  File "<ipython-input-11-395264872def>", line 1, in <module>
    print(degisken_adi)

NameError: name 'degisken_adi' is not defined
```

Python'da sıklıkla karşılaşılan hatalar



FileNotFoundError

- Var olmayan dosya açma hatasıdır.

Örnek

```
Traceback (most recent call last):

  File "<ipython-input-19-647c52a3240a>", line 1, in <module>
    dosya=open("dosyaAc.txt","r")

FileNotFoundError: [Errno 2] No such file or directory: 'dosyaAc.txt'
```

Python'da sıklıkla karşılaşılan hatalar

TypeError

- TypeError, yanlış/desteklenmeyen bir nesne türünde bir işlem gerçekleştirildiğinde ortaya çıkar.
  - İki tür arasında desteklenmeyen işlem: string ile nümerik değerin toplanmaya çalışılması gibi.
  - Çağrılamayan bir tanımlayıcıyı çağırmak: bir değişkeni fonksiyon gibi çağırmak.
  - Yanlış türde liste dizini: liste indeksini string veya float kullanmak gibi.
  - İteratif olmayan bir tanımlayıcı ile iterasyon: 3.45 ile iterasyon yapmak gibi.

Örnek

```
Traceback (most recent call last):

  File "C:\Users\_Bilgisayar_\.spyder-py3\temp.py", line 1, in <module>
    for i in range(1,3.45):

TypeError: 'float' object cannot be interpreted as an integer
```

Python'da sıklıkla karşılaşılan hatalar

KeyError

- Bir sözlük veri yapısında, olmayan bir anahtara (key) ulaşmak istediğinizde gerçekleşecektir.

Örnek

```
sözlük={"adı":"Ali", "soyadı": "Can"}
Traceback (most recent call last):

  File "<ipython-input-26-dff6878cc2fb>", line 1, in <module>
    sözlük["mesleği"]

KeyError: 'mesleği'
```

try-except Blokları

Python programlama dili beklenilen veya beklenmeyen hataları ayıklamak için iki ana blok sunmaktadır.

try

try bloğu içerisinde hata olması durumunda, Python except bloğuna atlar ve oradaki kodların çalıştırılmasını sağlar. Hata ile karşılaşılacak yerler blok içine alınan kısım olarak da bilinir.

01

except

try bloğunda herhangi bir hatanın olması durumunda burada yazılmış kodlar çalışır. try bloğunda herhangi bir hata olmaması durumunda buradaki kodlar çalıştırılmayarak atlanır.

02

Örnek 1

Örneğe bakıldığında s1 değerine 5, s2 değerine ise 0 girilirse, bir sayının sıfıra bölümü tanımsız olduğu için except bloğu çalışıp, ekrana "Bir sayıyı sıfıra bölemezsiniz sıfır dışında bir sayı girin" diye mesaj yazar.

Kod

```
s1 = int(input("Birinci Sayı : "))
s2 = int(input("İkinci Sayı : "))
try:
    sonuc = s1/s2
    print("Sonuc :",sonuc)
except ZeroDivisionError:
    print("Bir sayıyı sıfıra bölemezsiniz lütfen sıfır dışında bir sayı girin")
```

Örnek 2

Örnek'te görüldüğü üzere sayıları sıfırdan farklı ve sadece sayı verilirse herhangi bir hata vermeyecek ve sadece try bloğu çalışacaktır. Ama sayılardan herhangi birine sıfır veya farklı veri türünde değer girilmesi halinde except bloğu çalışıp bilgilendirme yapacaktır. Bu örnekte de ikinci sayıya "a" girildiği için kullanıcıya "lütfen sayısal bir değer girin" diye mesaj yazacaktır. Except bloğuna hatanın türü yazılmadığında yorumlayıcı tarafından bütün hataları kapsayacaktır. Bunu da sadece except bloğu ile genel olarak "beklenmeyen bir hata oluştu" diye mesaj ile belirtilebilir.

Kod

```
try:
    s1 = int(input("Birinci Sayı : "))
    s2 = int(input("İkinci Sayı : "))
    sonuc = s1/s2
    print("Sonuc : ",sonuc)
except ZeroDivisionError:
    print("Lütfen ikinci sayıya sıfırdan farklı bir değer girin")
except ValueError:
    print("Lütfen sayısal bir değer girin")
except:
    print("Beklenmeyen bir hata oluştu")
```

try except as

Kullanıcıya Python'a ait hata mesajları gösterilmek istendiğinde "as" ifadesi kullanılmaktadır.

Kod

```
try:
    s1 = int(input("Birinci Sayı : "))
    sonuc = s1**2
    print("Sonuc : ",sonuc)
except ValueError as hata:
    print("Lütfen sayı giriniz")
    print(hata)
```

Örnek'te sayı yerine harf girilmiştir ve hata mesajı da ekrana yazdırılmıştır.

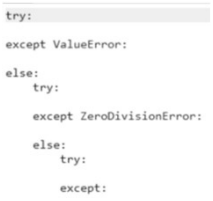
try except else

Oluşabilecek hataları adım adım ayıklanmak isteniyorsa “else” ifadesi kullanılmaktadır.

Kod

```
try:
    s1 = int(input("Birinci Sayı : "))
except ValueError:
    print("sayı girmediniz")
else:
    try:
        print(10/s1)
    except ZeroDivisionError:
        print("sayı sıfıra bölünemez")
```

Örnek programda herhangi bir sayı girilmemiştir veya karakter girilmiştir, except ValueError bloğu devreye girerek, ekrana “sayı girmediniz“ diye hata mesajı vermektedir.



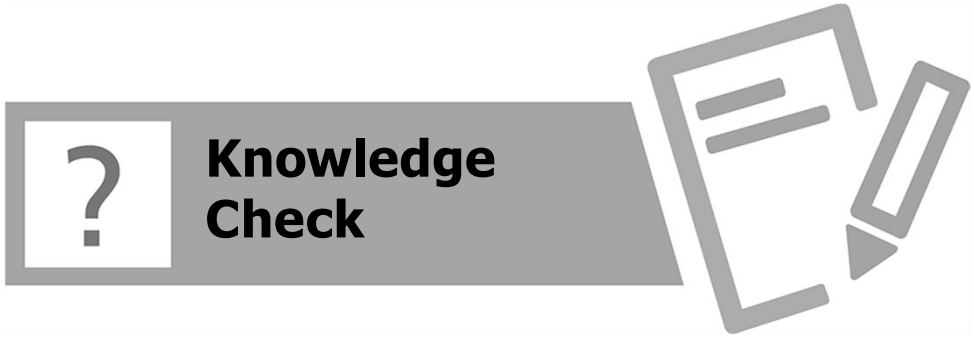
continue

Yanlış giden bir şey olduğunda devam edilmesini sağlar.

Kod

```
while True:
    x = input("Bir sayı girin: ")
    if not x:
        break
    try:
        y = float(x)
    except ValueError:
        print("Geçersiz sayı")
        continue
    print(y**2)
```

Kullanıcı hata aldığında programın devam etmesi için Örnek’teki gibi bir uygulama yapılabilir. continue ifadesi ile sayı yanlış girilirse tekrar girilmesi sağlanmaktadır.
Örnek’te ilk önce a harfi sonra sıfır değeri girince hata mesajları verir ve tekrar sayı girilmesi istenir. En sonunda 30 sayısı girildiğinde sonuç ekrana yazdırılır. Hiçbir şey girilmezse program sonlanır.



1	**Kullanıcı tarafından 3 tane sayı girilecek. Bu üç sayı toplanıp ekrana yazdırılacaktır. Hata ayıklama ile programı yapınız.**

1

Kullanıcı tarafından 3 tane sayı girilecek. Bu üç sayı toplanıp ekrana yazdırılacaktır. Hata ayıklama ile programı yapınız.

```
try:
    s1 = int(input("Birinci Sayi : "))
    s2 = int(input("ikinci Sayi : "))
    s3 = int(input("üçüncü Sayi : "))
    sonuc = s1+s2+s3
    print("Sonuc : ", sonuc)
except ValueError:
    print("lütfen sayısal bir değer giriniz")
```

Doğru Cevap

2

Kullanıcı tarafından 2 sayı girilip ortalaması alınacaktır. Hata ayıklama ile programı yapınız ve hatayı ekrana yazdırınız.

2	**Kullanıcı tarafından 2 sayı girilip ortalaması alınacaktır. Hata ayıklama ile programı yapınız ve hatayı ekrana yazdırınız.**

```
s1 = input("ilk sayi: ")
s2 = input("ikinci sayi: ")
try:
    sayi1=int(s1)
    sayi2=int(s2)
    sonuc=(sayi1+sayi2)/2
    print(sonuc)
except ValueError as hata:
    print(hata)
```

Doğru Cevap

Teşekkürler