Exception Handling(İstisna İşlemler)

Python hatalarını işlemek için try: ve except: kullanılır. Bir program ciddi bir hata durumunu algılar ve sonuç olarak kontrollü bir şekilde çıkar. Program bir terminale hata mesajı yazdırır. Bu uygulamayı daha sağlam hale gelir. İstisnalara örnek olarak yanlış giriş, yanlış dosya adı, dosya bulunamayan hataların zarif bir şekilde ele alınması, uygulamalarımızın çökmesini önler. Hata türlerini engellemek adına try ,except kullanırız.

Örnek:

try:

    print(10 + '5')

except:

    print('Something went wrong')

    # Something went wrong

Örnek:

try:

    name = input('Enter your name:')

    year\_born = input('Year you were born:')

    age = 2019 - year\_born

    print(f'You are {name}. And your age is {age}.')

except:

    print('Something went wrong')

    #Something went wrong

Yukarıda sorunun olduğuna dahil bilgiyi aldık fakat ne tür bir sorun olduğunu bilemiyoruz. Aşağıdaki örnekte, hatayı ele alacak ve ayrıca ortaya çıkan hatanın türünü de bize söyleyecektir.

try:

    name = input('ADIN:')

    year\_born = input('DOĞDUĞUN YIL:')

    age = 2019 - year\_born

    print(f'BENİM ADIM  {name}. {age} YILINDA DOĞDUM.')

except TypeError:

    print('Type error occured')

except ValueError:

    print('Value error occured')

except ZeroDivisionError:

    print('zero division error occured')

Yukarıdaki kodda çıktı TypeError olacaktır. Şimdi ek bir blok ekleyelim:

try:

    name = input('ADIN:')

    year\_born = input('DOĞDUĞUN YIL:')

    age = 2019 - int(year\_born)

    print('ad:',name)

    print('doğduğum yıl:',year\_born)

    print('BENİM ADIM ',(name),(age) ,'YAŞINDAYIM.')

except TypeError:

    print('Type error occur')

except ValueError:

    print('Value error occur')

except ZeroDivisionError:

    print('zero division error occur')

else:

    print('Genellikle yazdığım kodları denerim')

finally:

    print('Ben denizde sık sık yüzerim')

Çıktı:

ad: büşra

doğduğum yıl: 1996

BENİM ADIM  büşra 23 YAŞINDAYIM.

Genellikle yazdığım kodları denerim

Ben denizde sık sık yüzerim

Kısaltma olarak aşağıdaki halini de kullanabiliriz.

try:

    name = input('ADINIZ:')

    year\_born = input('DOĞDUĞUNUZ YIL:')

    age = 2019 - int(year\_born)

    print('SENİN {name} . {age} YAŞINDASIN.')

except Exception as e:

    print(e)

## **Packing and Unpacking Arguments in Python(Python'da Argümanları Paketleme ve Paketi Açma)**

İki operatör kullanırız.

* \*tuples için
* \*\*dictionaries için(dict())

### **Unpacking(Paket Açma)**

def  sum\_of\_five\_nums ( a , b , c , d , e ):

    return  a+  b  +  c  +  d  +  e

lst  = [ 1 , 2 , 3 , 4 , 5 ]

print ( sum\_of\_five\_nums ( lst ))

Bu kodu çalıştırdığımızda bir hata veriyor çünkü bu fonksiyon argüman olarak sayıları (liste değil) alıyor. Listeyi açalım.

def  sum\_of\_five\_nums ( a , b , c , d , e ):

    return  a +  b  +  c  +  d  +  e

lst  = [ 1 , 2 , 3 , 4 , 5 ]

print ( sum\_of\_five\_nums ( \* lst ))

# Liste toplamındaki sayılar 15 olduğunu görebiliriz.

Paket açmayı, bir başlangıç ​​ve bir bitiş bekleyen yerleşik işlevde de kullanabiliriz.

numbers = range(2, 7)

print(list(numbers))

 # [2, 3, 4, 5, 6]

args = [2, 7]

numbers = range(\*args)

print(numbers)      # [2, 3, 4, 5,6]

List veya tuples için aşağıdaki örneği inceleyelim.

countries = ['Finland', 'Sweden', 'Norway', 'Denmark', 'Iceland']

fin, sw, nor, \*rest = countries

print(fin, sw, nor, rest)   # Finland Sweden Norway ['Denmark', 'Iceland']

numbers = [1, 2, 3, 4, 5, 6, 7]

one, \*middle, last = numbers

print(one, middle, last)      #  1 [2, 3, 4, 5, 6] 7

#### Sözlükleri Açma

Örneğimizi inceleyelim.

def unpacking\_person\_info(name, country, city, age):

    return f'{name}  {country} {city} da yaşıyor. Büşra {age} yaşında.'

dct = {'name':'Büşra', 'country':'Türkiye', 'city':'İstanbul', 'age':25}

print(unpacking\_person\_info(\*\*dct))

## Paketleme

Bazen bir python işlevine kaç argüman iletilmesi gerektiğini asla bilemeyiz. Fonksiyonumuzun sınırsız sayıda veya isteğe bağlı sayıda argüman almasına izin vermek için paketleme yöntemini kullanabiliriz.

### Paketleme Listeleri

def sum\_all(\*args):

    s = 0

    for i in args:

        s += i

    return s

print(sum\_all(1, 2, 3))             # 6

print(sum\_all(1, 2, 3, 4, 5, 6, 7)) # 28

### Paketleme Sözlükleri

Örnek:

def packing\_person\_info(\*\*kwargs):

    for key in kwargs:

        print("{key} = {kwargs[key]}")

    return kwargs

print(packing\_person\_info(name="Büşra",

      country="Türkiye", city="İstanbul", age=25))

      #{'name': 'Büşra', 'country': 'Türkiye', 'city': 'İstanbul', 'age': 25}

## Python'da Yayılma

lst\_one = [1, 2, 3]

lst\_two = [4, 5, 6, 7]

lst = [0, \*lst\_one, \*lst\_two]

print(lst)          # [0, 1, 2, 3, 4, 5, 6, 7]

country\_lst\_one = ['Finland', 'Sweden', 'Norway']

country\_lst\_two = ['Denmark', 'Iceland']

nordic\_countries = [\*country\_lst\_one, \*country\_lst\_two]

print(nordic\_countries)  # ['Finland', 'Sweden', 'Norway', 'Denmark', 'Iceland']

## Numaralandırm İşlemi

Bir listenin dizini ile ilgileniyorsak, listedeki her bir öğenin dizinini almak için yerleşik numaralandırma işlevini kullanırız.

for index, item in enumerate([20, 30, 40]):

    print(index, item)

#0 20 1 30  2 40 yazar

for index, i in enumerate(countries):

    print('Merhaba')

    if i == 'Türkiye':

        print('The country {ü} has been found at index {index}')

## Posta Kodu(zip)

Bazen listeler arasında dolaşırken birleştirmek isteriz. Aşağıdaki örneği inceleyelim.

fruits = ['banana', 'orange', 'mango', 'lemon', 'lime']

vegetables = ['Tomato', 'Potato', 'Cabbage','Onion', 'Carrot']

fruits\_and\_veges = []

for f, v in zip(fruits, vegetables):

    fruits\_and\_veges.append({'fruit':f, 'veg':v})

print(fruits\_and\_veges)

#[{'fruit': 'banana', 'veg': 'Tomato'}, {'fruit': 'orange', 'veg': 'Potato'}, {'fruit': 'mango', 'veg': 'Cabbage'}, {'fruit': 'lemon', 'veg': 'Onion'}, {'fruit': 'lime', 'veg': 'Carrot'}]