Open Source Coding: Python

Bölüm 5: İstisnalar, Modül Yönetimi

İstisna ve Test Deyimleri

try / except	İstisna yakalama ve istisnadan kurtulma
try / finally	İstisna olsa da, olmasa da kod çalıştırma
raise	İstisna fırlatma
assert	Bir duruma bağlı istisna fırlatma
with / as	Context manager kullanma

İstisnaların Kullanım Alanları

1. Hatadan kurtulma / hatayı idare etme
2. İşlem sonucu döndürme
3. Özel durumları idare etme
4. Sonuç işlemleri
5. Kodun akışını değiştirme

Öntanımlı Python İstisnaları

```
$ python3
>>> L = (1, 3, 5, 7, 9)
>>> L[7]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: tuple index out of range
>>>
```

- Öntanımlı Python fonksiyonlarının öntanımlı istisnaları vardır.
- İstisnayı yakalayan bir kod yazılmadıysa, **default exception handler** çalıştırılır ve hata mesajı konsola yazdırılır.

İstisna Yakalamak

```
$ python3
>>> L = (1, 3, 5, 7, 9)
>>> try:
...     L[7]
... except IndexError:
...     print('bad index')
...
bad index
```

- Try bloğunda fırlatılan istisna, except bloğunda, ancak doğru istisna türü karşılanabilirse yakalanabilir.

İstisna İle Karşılaşmak

- Bir kere istisna ile karşılaşmak, try bloğundan çıkmaya yeter. Sonraki satırlardaki kod çalıştırılmaz.

```
$ python3
>>> L = (1, 3, 5, 7, 9)
>>> try:
...     print(L[1]*2))
...     print(L[99])
...     print(L[3]-L[0])
... except IndexError:
...     print('index error')
...
6
index error
>>>
```

Doğru İstisnayı Yakalamak

- Except bloğunda fırlatılan istisna yakalanmazsa, bu istisna yine **default exception handler**'a gider.

```
$ python3
>>> L = (1, 3, 5, 7, 9)
>>> try:
...     print(L[1]*2)
...     print(L[3]-L[0])
...     print(L[99])
... except IOError:
...     print('io exception')
...
6
6
Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
IndexError: tuple index out of range
>>>
```

Tüm İstisnaları Yakalamak

- Except deyiminde bir istisna belirtilmezse, ya da Exception sınıfı belirtilirse tüm istisnalar yakalanabilir.

- Temel istisna sınıfı **BaseException**'dır.

```
$ python3
>>> L = (1, 3, 5, 7, 9)
>>> try:
...     L[7]
>>> except:     # veya except Exception:
...     print('bad index')
...
bad index
```

Hatadan Kurtulmak, İşleme Devam Etmek

- try/except bloğunda hata yakalanabildiyse, sonrasında gelen kodların çalıştırılmasına devam edilir.

```
$ python3
>>> def printObj(x, y):
...     try:
...             print(x[y])
...     except:
...             print(‘bir istisna')
...     print('devam et')
...
>>> printObj((1,2,3,4),100)
bir istisna
devam et
>>>
```

İstisna Fırlatmak

- **raise** deyimiyle istisna fırlatılabilir.

- Raise deyiminden sonra belirtilen istisna sınıfıyla belli bir çeşit istisna fırlatılabilir. Bir sınıf belirtilmezse RuntimeError sınıfı kullanılır.

```
$ python3
>>> try:
...     raise IndexError
... except:
...     print('index error')
...
index error
>>>
```

İstisna Nesnesini Kullanmak (I)

```
def kullan(x):
  print(x[1] * x[0])

try:
  kullan(None)
except TypeError as e:
  print('bad type: ' + e.args[0])
```

```
bad type: 'NoneType' object is not
subscriptable
```

İstisna Nesnesini Kullanmak (II)

```
def faktoriyel(n):
   if(n < 0):
      raise Exception("negatif sayı")
   if(n == 0 or n == 1):
      return 1
   return n * faktoriyel(n-1)

try:
   print(faktoriyel(5))
   print(faktoriyel(-1))

except Exception as e:
   print(e.args[0])
```

```
120
negatif sayı
```

Bazı Öntanımlı İstisnalar

İstisna Sınıfı	ne zaman fırlatılır?
IndexError	bir alt aralık, kapsam dışında olduğunda
KeyError	dictionary'de bir anahtara karşılık gelen bir değer bulunamadığında
NameError	bir local veya global değişken ismi bulunamadığında
TypeError	bir işlem veya fonksiyona uyumsuz bir türde değişken gönderildiğinde
RuntimeError	sınıfı belli olmayan bir istisna (sadece raise) fırlatıldığında
AssertionError	assert deyimi başarısız olduğunda
EOFError	input() kaynağı bittiğinde (dosya sonu)
ZeroDivisionError	sıfıra bölme işlemi yapıldığında

Bir İstisna Sınıfı Tanımlamak

- En basit özel isimli istisna tanımlaması:

```
class MyException(Exception):
    pass
```

```
try:
    if(n<0):
        raise MyException
except MyException:
    print('MyException yakala')
```

- Tanımlanan sınıfa metodlar eklenerek try - except blokları arasında daha fazla veri aktarılabilir, farklı işler yapılabilir.

Birden Fazla except

- try bloğundan sonra alt alta yazılan birden fazla except bloğu ile farklı istisna durumları ayrı ayrı idare edilebilir.

- İstisnalar yazılma sırasıyla kontrol edilir, uygun olan ilk except bloğuna girilir.

```
try:
   a
except IOError:
   print('io error')
except NameError:
   print('name error')
except Exception:
   print('geri kalan her sey’)
```

```
name error
```

finally

- try bloğu içinde istisna olsa da, olmasa **en son mutlaka çalıştırılması** gereken şeyler için bir kod bloğu

- Yaygın kullanım alanı: dosyayı kapatmak, veritabanını kapatmak…

```
try:
    print(faktoriyel(7))
    print(faktoriyel(-1))
finally:
    print('en son’)
```

```
5040
en son
Traceback (most recent call last):
  File "f.py", line 10, in <module>
    print(faktoriyel(-1))
  File "f.py", line 3, in faktoriyel
    raise Exception("negatif sayı")
Exception: negatif sayı
```

```
print(faktoriyel(7))
print(faktoriyel(-1))
print('en son’)
```

```
5040
Traceback (most recent call last):
  File "f.py", line 10, in <module>
    print(faktoriyel(-1))
  File "f.py", line 3, in faktoriyel
    raise Exception("negatif sayı")
Exception: negatif sayı
```

try / except / else

```
try:
   statements # önce ana ifadeyi çalıştır
except name1:
   statements # name1 istisnası fırlatıldıysa çalıştır
except (name2, name3):
   statements # name2 veya name3 fırlatıldıysa
              # çalıştır
except name4 as var:
   statements # name4 istisnası fırlatıldıysa çalıştır
except:
   statements # başka bir istisna fırlatıldıysa
              # çalıştır
else:
   statements # hiç istisna fırlatılmadıysa çalıştır
```

İç içe try

```
D = {'key':'value'}
try:
    try:
        print(D['baska'])
    except KeyError:
        print('key error')

    print('devam et')

    print(faktoriyel(-1))
except Exception as e:
    print(e.args)
```

```
key error
devam et
('negatif sayı',)
```

assert

- Test ve debug işlemlerini kolaylaştırmak için bir deyim
- Kullanım:

```
assert test, data
```

- Eşdeğeri:

```
if __debug__:
   if not test:
      raise AssertionError(data)
```

assert Örneği

```
def f(x):
   assert x < 0, 'x must be negative'
return x ** 2
print(f(-2))
print(f(1))
```

```
4
Traceback (most recent call last):
  File "assert.py", line 5, in <module>
    print(f(1))
  File "assert.py", line 2, in f
    assert x < 0, 'x must be negative'
AssertionError: x must be negative
```

with / as Context Manager

- Basit kullanım:

```
with ifade [as degisken]:
   with-bloğu
```

```
with open(r’data.txt’) as myfile:
   for line in myfile:
      print(line)

myfile = open(r’data.txt')
try:
   for line in myfile:
      print(line)
finally: myfile.close()
```

Python'da Modüller

Modül

- Python kodu (tanımlamaları, ifadeleri) içeren dosyalara denir.
- .py uzantısı
- Bir python modülüne **import** ile erişilir.
  - modul.py modülüne erişmek için:
    - `import modul`
- import deyimi, varsayılan olarak yüklenen modüldeki her şeyi modülün ismini taşıyan bir değişkene atar.
- **from** deyimi ile bir modülden belirli bir fonksiyon alınabilir.

Modül Örneği (I)

fibo.py:

```
def fib(n):
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()

def fib2(n):
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result
```

```
import fibo

fibo.fib(5)
print(fibo.fib2(5))
```

```
1 1 2 3
[1, 1, 2, 3]
```

```
from fibo import fib

fib(5)
```

```
1 1 2 3
```

Modül Örneği (II)

fibo.py:

```
def fib(n):
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()

def fib2(n):
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result
```

```
from fibo import fib, fib2

fib(5)
print(fib2(5))
```

```
1 1 2 3
[1, 1, 2, 3]
```

```
from fibo import *

fib(5)
print(fib2(5))
```

```
1 1 2 3
[1, 1, 2, 3]
```

Modüldeki Globaller

mymodule.py

```
def pow(a,b):
   return a**b

X = 42

print('selam')
```

```
import mymodule

print(mymodule.pow(4,3))

print(mymodule.X)
```

```
selam
64
42
```

Teşekkürler!

Bir sonraki oturumda görüşmek üzere.