Python 3 – iterator (yineleyici) ve iterable (yinelenebilen)

Iterator

- Iterator, sayılabilir sayıda değer içeren bir nesnedir.
- Iterator, yinelenebilen bir nesnedir, yani tüm değerler arasında geçiş yapabilirsiniz.
- Teknik olarak Python'da iterator, iter () ve next () fonksiyonlarından oluşan yineleyici protokolünü uygulayan bir nesnedir.

Iterable

 list, tuple, dict ve set veri yapılarının tümü yinelenebilir nesnelerdir. Bir iterator alabileceğiniz iterable yapılardır. Her birinin iterator elde etmek için kullanılan bir iter() fonksiyonu vardır.

29.06.2022 Python Eğitimi - Dr. Fatma GÜMÜŞ 301

301

Python 3 – iterator (yineleyici) ve iterable (yinelenebilen)

• list, tuple, dict ve set veri • String'ler de bir dizi karakter yapılarının tümü yinelenebilir nesnelerdir.

içeren iterable nesnelerdir:

```
mytuple = ("apple", "banana", "cherry")
myit = iter(mytuple)
                                                     banana
print(next(myit))
print(next(myit))
print(next(myit))
```

 Iterable bir for nesnevi döngüsünde kullanabiliriz:

```
mystr = "apple"
myit = iter(mystr)
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
```

```
mytuple = ("apple", "banana", "cherry")
for x in mytuple:
 print(x)
```

- Dr. Fatma GÜMÜŞ

302

1

302

banana

cherry

Python 3 — iterator (yineleyici) ve iterable (yinelenebilen)

Iterator sınıfı oluşturmak

- Iterator olarak bir nesne/sınıf oluşturmak için __iter__() ve __next__() fonksiyonları kodlamanız gerekir.
- Python'da tüm sınıfların __init__() adında bir fonksiyonu vardır ve bu, nesne oluşturulurken çağırılır.
- <u>iter</u> () fonksiyonu buna benzer davranır, işlemler yapabilirsiniz (ilk değer atama vb.), ancak her zaman iterator nesnenin kendisini döndürmeniz gerekir.
- next__() fonksiyonunda da işlem yapabilirsiniz ve sıradaki bir sonraki öğeyi döndürmesi gerekir.

 29.06.2022
 Python Eğitimi - Dr. Fatma GÜMÜŞ
 303

303

Python 3 – iterator (yineleyici) ve iterable

(yinelenebilen)

Stoplteration

- Yeterli next() ifadeniz varsa veya bir for döngüsünde kullanılmışsa, soldaki örnek sonsuza kadar devam eder.
- Yinelemenin sonsuza kadar devam etmesini önlemek için Stoplteration ifadesini kullanabiliriz.
- __next__() yönteminde, yineleme belirli sayıda yapılırsa bir hata oluşturmak için bir sonlandırma koşulu ekleyebiliriz (sağda).

```
class MyNumbers:
   def __iter__(self):
     self.a = 1
     return self
   def __next__(self):
     x = self.a
     self.a += 1
     return x
 myclass = MyNumbers()
 myiter = iter(myclass)
 print(next(myiter))
 print(next(myiter))
 print(next(myiter))
 print(next(myiter))
 print(next(myiter))
Python Eğitimi - Dr. Fatma GÜMÜS
```

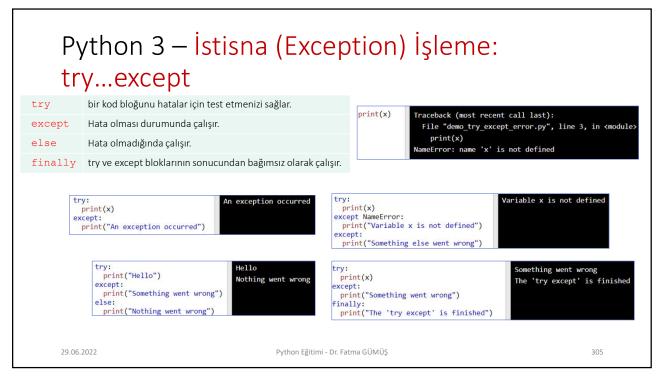
2

```
class MyNumbers:
 def __iter__(self):
    self.a = 1
    return self
        _next__(self):
    if self.a <= 20:
      x = self.a
      self.a += 1
                                 8
      return x
      raise StopIteration
                                 10
myclass = MyNumbers()
                                 11
myiter = iter(myclass)
                                 12
for x in myiter:
  print(x)
                                 16
                                 17
                                 18
                                 19
                                 20
```

2

29.06.2022

304



305

```
Python 3 – İstisna (Exception) İşleme:
      try...except
try
           bir kod bloğunu hatalar için test etmenizi sağlar.
           Hata olması durumunda çalışır.
           Hata olmadığında çalışır.
else
finally try ve except bloklarının sonucundan bağımsız olarak çalışır.
        try:
    f = open("demofile.txt")
         try:
f.write("Lorum Ipsum")
           print("Something went wrong when writing to the file")
           f.close()
         print("Something went wrong when opening the file")
         C:\Users\My Name>python demo_try_except5.py
      29.06.2022
                                                   Python Eğitimi - Dr. Fatma GÜMÜS
                                                                                                                      306
```

306

Python 3 – İstisna (Exception) İşleme:Raise an exception

- Bir koşul oluştuğunda, bir istisna (exception) oluşturmayı seçebilirsiniz.
- Bir istisna atmak (veya yükseltmek) için «raise» anahtar sözcüğünü kullanılır.

```
x = -1

if x < 0:
    raise Exception("Sorry, no numbers below zero")

Exception: Sorry, no numbers below zero")

Traceback (most recent call last):
    File "demo_ref_keyword_raise.py", line 4, in <module>
    raise Exception("Sorry, no numbers below zero")

Exception: Sorry, no numbers below zero
```

• Ne tür bir hata oluşturulacağını ve kullanıcıya yazdırılacak metni tanımlayabilirsiniz (Hata türleri için bkz.

https://docs.python.org/3/library/exceptions.html)

```
x = "hello"

if not type(x) is int:
    raise TypeError("Only integers are allowed")

29.06.2022

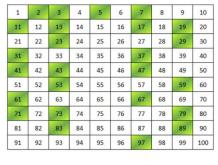
Traceback (most recent call last):
File "demo_ref_keyword_raise2.py", line 4, in <module>
    raise TypeError("Only integers are allowed")
TypeError: Only integers are allowed
29.06.2022
```

307

Python 3 – İstisna (Exception) İşleme:Raise an exception

Alıştırma Ödevi

Daha önce (Fonksiyonlar) bölümünde yazdığınız prime_checker fonksiyonunuzu aşağıdaki girdiler için denetleyin. Gerekiyorsa, beklenen çıktıya göre düzenleme YANLIŞ CEVAP çıktılar:



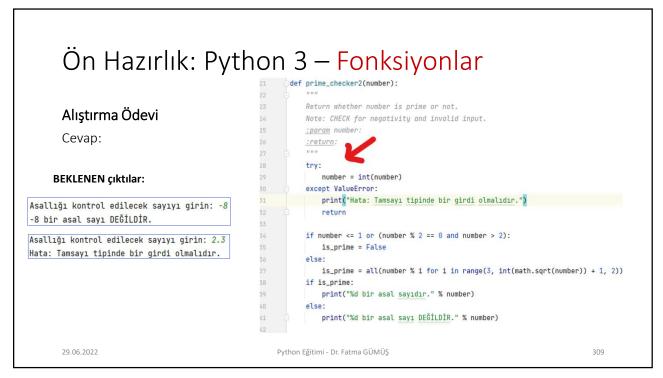
-8 bir asal sayı DEĞİLDİR.

Asallığı kontrol edilecek sayıyı girin: 2.3
Hata: Tamsayı tipinde bir girdi olmalıdır.

29.06.2022 Python Eğitimi - Dr. Fatma GÜMÜŞ

308

308



309