Iteratorlar va Generatorlar

Python ikkita umumiy maqsadli iterator ob'ektini taqdim etadi. Birinchisi, ketma-ketlik iteratori, [\_\_getitem\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__getitem__) usulni qo'llab-quvvatlovchi ixtiyoriy ketma-ketlik bilan ishlaydi. Ikkinchisi chaqiriladigan ob'ekt va qo'riqchi qiymati bilan ishlaydi, ketma-ketlikdagi har bir element uchun chaqiriladiganni chaqiradi va sentinel qiymati qaytarilganda iteratsiyani tugatadi.

Iter()=>Qaytarilgan iterator ob'ektlari uchun ob'ektni yozing va o'rnatilgan ketma-ketlik turlari uchun o'rnatilgan qaytarilgan funksiyaning bir argumentli shakli

Keeling kod da iteratorlarini kuramiz bizga iteratorlarda asosiy logikamiz next()da yoziladi keeling yozib kuramiz

class MyIterator:  
 def \_\_init\_\_(self, start):  
 self.start = start  
 self.step = 1  
  
 def \_\_iter\_\_(self):  
 return self  
  
 def \_\_next\_\_(self):  
 item = self.start  
 self.start += self.step  
 return item  
  
  
my\_iterator = MyIterator(2)  
for i in my\_iterator:  
 print(i)

print(next(my\_iterator))

2 dan boshlab 1 ga oshiradigan cheksiz likka ketatigan sonlar ketma ketligi

Agar next() bilan yozsak nechta next yossak ushancha sini chiqaradi

Generator:

**Pythonda generatorlar**

Oxirgi yangilangan: 06-sentabr, 2023-yil

Python'dagi Generator - bu Yield kalit so'zidan foydalangan holda iteratorni qaytaradigan funksiya. Ushbu maqolada biz Python-da generator funktsiyasi qanday ishlashini muhokama qilamiz.

**Pythonda generator funktsiyasi**

Python-da generator funksiyasi oddiy funktsiya kabi aniqlanadi, lekin u har doim qiymat yaratishi kerak bo'lganda, u buni qaytarish emas, balki [yield kalit so'zi](https://www.geeksforgeeks.org/python-yield-keyword/) bilan bajaradi. Agar def tanasi hosilni o'z ichiga olsa, funksiya avtomatik ravishda Python generator funksiyasiga aylanadi.

Generator yaratish uchun oddiy def funksiya orqali yaratsak ham buladi yield va return farqi nma desak

Return bitta qiymat qaytaradi yield esa bir nechta qiymatni qaytarishi mumkun ularni asosiy farqi ushanda

Generator metodlari:

Send=>Amalga oshirishni davom ettiradi va generator funktsiyasiga qiymatni "yuboradi". Argument valuejoriy ifodaning natijasiga aylanadi yield. Usul send()generator tomonidan berilgan keyingi qiymatni qaytaradi yoki StopIterationagar generator boshqa qiymat bermasdan chiqsa, oshirad

Misol kuramiz:

def square():  
 while True:  
 value = yield  
 yield value \*\* 2  
  
gen = square()  
next(gen)  
print(gen.send(4))  
  
next(gen)  
print(gen.send(10))  
for i in range(1, 10):  
 next(gen)  
 print(gen.send(i))  
 if i == 5:  
 gen.close()

throw():

bu usha joygacha ishla undan keyin xatolik tashla degan manoda keladi xatolik turini uzi iz beramiz qolda:

misol quyidagicha soda kodda koramiz

def func():  
 for i in range(1, 25):  
 yield i \*\* 2  
  
  
gen = func()  
print(next(gen))  
print(next(gen))  
print(next(gen))  
print(next(gen))  
print(next(gen))  
gen.throw(IndexError)

close():

bir soz bilan aytganda kodimizni usha joyga yetganda toxtatib qoyadi desak ham boladi: keeling kod shaklida koramiz

def func():  
 try:  
 yield 1  
 yield 2  
 yield 3  
 except StopIteration:  
 print('GeneratorExit')  
  
  
gen = func()  
print(next(gen))  
gen.close()  
print(next(gen))

pastga ikkinchi marta print qilganda xatolik tasheldi yani close()

qilabmiz yani kodimizni toxtatabmiz.

Generator va Iterator haqida qisqacha malumot

Etibor uchun raxmat!