**27/04/2019 Lecture30**

**Custom Iterator**

**#write class with name as autogenerate with init method in it.**

class AutoGenerate:

def \_\_init\_\_(self,start,end,step=1):

self.start=start

self.end = end

self.step = step

def next(self):

self.start+=self.step

if self.start >=self.end:

raise StopIteration

return self.start

def \_\_next\_\_(self):

return self.next()

def \_\_iter\_\_(self):

return self

def main():

x= AutoGenerate(0,100,5)

for y in x:

print y

if \_\_name\_\_=="\_\_main\_\_":

main()

**GENERATOR**

#write class with name as autogenerate with init method in it.

class AutoGenerate:

def \_\_init\_\_(self,start,end,step=1):

self.start=start

self.end = end

self.step = step

def next(self):

self.start+=self.step

if self.start >=self.end:

raise StopIteration

return self.start

def \_\_next\_\_(self):

yield self.next()

def \_\_iter\_\_(self):

return self

def main():

x= AutoGenerate(0,100,5)

for y in x:

print y

if \_\_name\_\_=="\_\_main\_\_":

main()

**Reveresed iterator**

>>> x=range(4)

>>> for y in reversed(x):

print(y)

3

2

1

0

>>>Reversed\_iterator.py

#!/usr/bin/python

class ReferenceCount:

def \_\_init\_\_(self, start):

self.start = start

#Forward Iterator

def \_\_iter\_\_(self):

start = self.start

while start > 0:

yield start

start -= 1

def \_\_reversed\_\_(self):

start = 1

while start <= self.start:

yield start

start += 1

def main():

x = ReferenceCount(10)

print("Display from 10---1")

for y in x:

print (y)

print("Display from 1---10")

for w in reversed(x):

print(w)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**acceptYield.py**

#!/usr/bin/python

def increament():

x= yield

yield x+10

data = increament()

next(data)

#data.send(None)

print(data.send(10))

**#AcceptYieldWithWrapperSimplified**

#!/usr/bin/python

def initWrapper(func):

def Init():

data=func()

next(data)

return data

return Init

@initWrapper

def Increement():

x=yield

yield x+10

data=Increement()

print(data.send(10))

**#write a function which accepts string via yield and returns upper case**

#!/usr/bin/python

def initWrapper(func):

def Init():

data=func()

next(data)

return data

return Init

@initWrapper

def toUpper():

x=yield

yield x.upper()

data=toUpper()

print(data.send('swapnil'))

**#Write decorator chaining example from git**