**Assignment: 9**

**Experiments**

1. **Write your own iterator class and iterate through it (without generator function).**

**Coding:**

class MyIterator:  
 def \_\_init\_\_(self, list\_of\_numbers):  
 self.list\_of\_numbers = list\_of\_numbers  
 self.index = 0  
  
 def \_\_iter\_\_(self):  
 return self  
  
 def \_\_next\_\_(self):  
 if self.index == len(self.list\_of\_numbers):  
 raise StopIteration  
 else:  
 index = self.index  
 self.index += 1  
 return self.list\_of\_numbers[index]  
  
#Taking input from user  
list\_of\_numbers = [int(x) for x in input('Enter the numbers : ').split()]  
# Creating an instance of MyIterator  
my\_iterator = MyIterator(list\_of\_numbers)  
#Iterating through the iterator  
for num in my\_iterator:  
 print(num)

**Output:**



1. **Write your iterator by using generator function.**

**Coding:**

def my\_generator(n):  
 for i in range(n):  
 yield i  
  
n = int(input("Enter a number: "))  
  
for item in my\_generator(n):  
 print(item)

**Output:**



1. **Write a function that calculates simple interest. Now decorate the output of this function by returning the modified amount.**

**Coding:**

def calculate\_simple\_interest(principal, rate, time):  
 si = (principal \* rate \* time) / 100  
 return si  
  
def decorate\_amount(amount):  
 return 'Amount : ' + str(amount)  
  
principal = float(input('Enter principal amount : '))  
rate = float(input('Enter rate of interest : '))  
time = float(input('Enter time in years : '))  
  
simple\_interest = calculate\_simple\_interest(principal, rate, time)  
print(decorate\_amount(simple\_interest))

**Output:**

