# Generators

Monday, August 4, 2025 7:42 PM

It is the process of creating the new collection using yield keyword, where the function is used to perform operations.

Eg:

```
def sam():
   print('Hii')
   return 1
   print('Hello')
   return 2
   print('Bye')
   return 3
print(sam())
```

OP:

hii

1

```
def sam():
   print('Hii')
   yield 1                  # [1]
   print('Hello')
   yield 2                  #  [1,2]
   print('Bye')
   yield 3                  #   [1,2,3]
print(list(sam()))
```

OP:

```
Hii
Hello
Bye
[1, 2, 3]
```

- Instead of return, yield keyword is used.
- While using yield, typecasting is required.
- Whenever the control sees the yield keyword, it considers the function as a generator function.
- If we call the  function without typecasting, it will give us the address.
- Yield: It is used to pause the execution to store the value to the collection.
  After storing, it will again return back for the execution.
- It will resume the execution till the last instruction or until it doesn't see the return keyword.

Difference Between Return and Yield

| | |
|---|---|

| Return | Yield |
|---|---|
| Used to stop the execution by returning The values. | It pause the execution, store the value in collection And continue the execution. |
| Used in normal function | Used in generator function |
| Can be called directly | For calling, typecasting is required. |

Now, we can say that it is a function used to create a collection using yield keyword.

- If yield and return both are present then yield is given the more priority.
- But if control sees the return keyword first then it will stop the execution there only.

Eg:

```
def sam():
   print('Hii')
   yield 1
   print('Hello')
   return 2
   print('Bye')
   yield 3
print(tuple(sam()))'''

#WAP to create sqr of all the numbers from 1 to 10
'''def sqr():
   for i in range(1,11):
      yield i**2
print(list(sqr()))'''

#WAP to extract all the string values from the list if it is palindrome
'''def sam(l):
   for i in l:
      if type(i)==str and i[::-1]==i:
         yield i
print(list(sam(['nayan',12,4.5,'abcd'])))'''

#WAP to create a dictionary using generator of char and its ASCII value
'''def sam():
   for i in range(65,91):
      yield chr(i),i
print(dict(sam()))'''
```

Assignment:

1. WAP to extract all the prime numbers between m to n.
2. WAP to generate 10 fibonacci series.
3. WAP to swap case the character of the input string.