

17/9/25

Task 8.1 implement Python generator and decorators

Aim:- write a Python program to implement Python generator and decorators.

Algorithm:-

- 1) Start Function
- 2) Initialize counter set value to 0
- 3) Generate values.
 - * yield the current value
 - * Increment value by 1
- 4) Create generator object:
- 5) iterate and print value
 - * for each value produced by the generator object:
 - * print value.

8.1 Program:-

```
def my_generator(n):
    # Initialize counter
    value = 0
    # loop until counter is less than n
    while value < n:
        # produce the current value of counter
        yield value
        # increment the counter
```

Q79 Acceptable

the name of a bank branch for troubleshooting

the beneficiary and destination

Q80

reject for how

Q81 Input - to provide feedback to - & Output -

0

1

2

reject of advice

* reject of advice

* illegal advice

* Q82 Illegal advice

* illegal advice

* for each advice provided by this subscriber

copy:

* write advice

Q83 Answer:

QSF N1 - Generation (or)

* police (or)

advice to

* 1000 number in case of emergency

* advise > 0;

* illegal advice

* illegal advice

* illegal advice

```
value += 1  
# iterate over the generator object provided  
# by my_generator  
for value in my_generator(3):  
    # print each value produced by generator  
    print(value)
```



Result:- Thus, the ~~program~~ Python generator
has been executed and implemented successfully.

TASK 8.2 :- Imagine you are working on a messaging application that needs to format user's preferences.

Aim:- Write a Python program to implement Python generator and decorators.

Algorithm:-

1) Create decorators:

2) Define Functions:

* Define shout function to return the input text. Apply

3) Define greet function:

* Accept a function [func] as input

4) Execute the program:

* call greet [shout] to print the greeting in uppercase

* call greet [whisper] to print the greeting in lowercase.

Program:-

```
def uppercase_decorator(func):
    def wrapper(text):
        return func(text).upper()
    return wrapper
```

class Application extends Application
{
 public void onCreate() {
 super.onCreate();
 // Set the content view
 setContentView(R.layout.activity_main);

 // Get the text view
 TextView textView = (TextView) findViewById(R.id.textView);

 // Set the text
 textView.setText("Hello World");
 }
}

Output:-

Hi, I AM created By a Function passed as
An argument.

Hi, i am created by a function passed as
argument.

```
def lowercase_decorator(func):  
    def wrapper(text):  
        return func(text).lower()  
    return wrapper
```

@uppercase_decorator

```
def shout(text):  
    return text
```

greet [shout]

greet [whisper]

VEL TECH	
EX No.	8
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCÉ (5)	5
AFCORDO (5)	
TOTAL (20)	15
WITH DATE	

Result:- Thus, the program to implement python generator and decorators was successfuly executed and the output verified