# Cheatography

## **ITERATORS GENERATORS DECORATORS Cheat Sheet** by sahusourabh via cheatography.com/156407/cs/33300/

## **ITERATORS**

#### Looping

Important functions to be implemented

\_\_iter\_\_()

\_\_next\_\_()

\_\_iter\_\_() : takes iteratable object like list,

tuples

\_\_next\_\_(): is used to return the next value in iteration

## Use of iterators

```
1
 2 lst = [1,"Sourabh",5,3.0]
 3 itr = iter(lst)
 4 # iterate through it using next()
 5 print(next(itr))
 6 print(next(itr))
 7 print(itr.__next__())
 8 print(itr.__next__())
1
Sourabh
5
3.0
```

#### Iterators with class

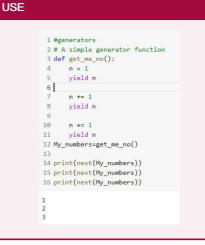


## GENERATORS

1. Generator generate one element at a time from a sequence.

2. Yield is used to get the value

3 It saves the state not like function where once function is called state will be returned to new call



## List Comprehension vs Generation

Comprehension: all in one go

Generation : one by one ...fast

#### **EXAMPLE**

1

```
2 lst= [1, 4, 6, 8]
  2 lst= [1, 4, 6, 8]
3 # square each term using list comprehension
4 square_list = [x**2 for x in lst]
5 # same thing can be done using a generator expression
6 # generator expressions are surrounded by parenthesis ()
7 generator = (x**2 for x in lst)
   8 print(square_list)
   9 print(generator)
[1, 16, 36, 64]
<generator object <genexpr> at 0x7f66c39c39d0>
  1 print(next(generator))
  1 print(next(generator))
16
```

### DECORATORS

1. A decorator is a special function which adds some extra functionality to an existing function

2. A decorator is a function that accepts a function as a parameter and returns a function.

3. Decorators are useful to perform some additional processing required by a function.

Want to add addition functionality to function

- 1 def

ef decor(func): def inner\_function(m,f,y): if "TesIa" in m: print("WOW!! Its TesIa Electric Vehicle") return func(m,f,y)

- return inner\_function
- 8 def sh

def showvehicle(model,fueltype,year):
 print("Broom...its ",model,fueltype,"manufactured in",year) 11 showvehicle = decor(showvehicle)

12 print(showvehicle("BMW 2 Series ","Petrol","2020")) 13 print(showvehicle("Tesla Model 3","EV","2022"))

Broom...its BMW 2 Series Petrol manufactured in 2020

WOW!! Its Tesla Electric Vehicle Broom...its Tesla Model 3 EV manufactured in 2022

#### @decor

showvehicle = decor(showvehicle) instead

of this line, you can use @decor

#### @decor implementation

1 def decor(func): ef decor(tunc): def inner\_function(m,f,y): if "Tesla" in m: print("WOW!! Its Tesla Electric Vehicle") return func(m,f,y) return inner\_functio 8 def showvehicle(model,fueltype,year): print("Broom...its ",model,fueltype,"manufactured in",year) 11 print(showvehicle("BMW 2 Series ","Petrol","2020"))
12 print(showvehicle("Tesla Model 3","EV","2022")) Broom...its BMW 2 Series Petrol manufactured in 2020 None WOW!! Its Tesla Electric Vehicle Broom...its Tesla Model 3 EV manufactured in 2022

By sahusourabh

cheatography.com/sahusourabh/

Published 22nd July, 2022. Last updated 22nd July, 2022. Page 1 of 1.

Sponsored by Readable.com Measure your website readability! https://readable.com