



Decorators in Python-Complete Tutorial

Python / March 28, 2020



This **Decorators in Python-Complete Tutorial** is made for you. At the end of this tutorial, you will have full knowledge of the **Decorators in Python**.

Hello, & Welcome!

Part 18- Decorators in Python-Complete Tutorial

In this article of Python Tutorial, you will learn following-

1. What is a Decorator in Python?
2. How to Create a Decorator in Python?
3. Use of Decorator in Python.

What is a Decorator in Python?

A Decorator extends the behavior of the function. A decorator allows implementing new features to an existing function without modifying the function itself. You can call the decorator before the function in which you wanna decorate.

For example-

```
@class_method  
def commons(cls):
```

Here, `@class_method` is a decorator. It extends the behavior of the function and it describes the function behavior.

How to Create a Decorator in Python?

A Decorator is itself a function. You can create your own decorator, so for creating own decorator, let's see in the example below-

Before defining any decorator, use the "@" operator.

Syntax of Decorator-

```
@decorator_name
def function_name
    #function body
```

Call decorator before any function, in which you wanna use a decorator.

Example-

```
def cough_dec(func):
    def func_wrapper():
        print("cough")          #code before function
        func()
        print("cough")          #code after function
    return func_wrapper

@cough_dec
def question():
    print("can you give discount?")

@cough_dec
def answer():
    print("it's only 50")

question()
answer()
```

OUTPUT-

```
*cough*
can you give discount?
*cough*
*cough*
it's only 50
*cough*
```

Here, “**def func_wrapper()**” is a wrapper function, which is defining the decorator. “**return func_wrapper**” is returning wrapper function. “**@cough_dec**” is a decorator.

To create your own decorator, you can create it as a function like in the example “**def cough_dec(func)**“. And inside that function, create a wrapper function like “**def func_wrapper()**“. Where you can define the behavior of a function.

After that, outside from wrapper function “**def func_wrapper()**“, return it as “**return func_wrapper**“. So, by doing this you can create your own decorator and then you can use this decorator many time, whenever you want.

Use of Decorator in Python–

Decorators are used in –

- Logging,
- Run time check,
- Synchronization,
- Type checking,
- Debugging,
- In web framework and many more.

That’s all for the **Decorator in Python**. I hope now you have a better understanding of the **Decorator in Python**.

Congratulations! You successfully learned the **Decorator in Python**.



In the next tutorial, we will start learning **Reading & Writing files in Python**.

Till then, Enjoy Learning Python!

All the Best!

Are you ML Beginner and confused, from where to start ML, then read my BLOG – [How do I learn Machine Learning?](#)

If you are looking for Machine Learning Algorithms, then read my Blog – [Top 5 Machine Learning Algorithm](#).

If you are wondering about Machine Learning, read this Blog- [What is Machine Learning?](#)

Thank YOU!

Though of the Day...

“Live as if you were to die tomorrow. Learn as if you were to live forever.”

– Mahatma Gandhi

[← Previous Post](#)

[Next Post →](#)

Leave a Comment

Your email address will not be published. Required fields are marked *

Type here..

☐ Save my name, email, and website in this browser for the next time I comment.

Follow Us on Twitter-



Follow us on Pinterest-



Our Facebook Group-



