

A photograph of a person's hands, palms up, holding a small, white, fluffy bird chick. The person is wearing a light blue t-shirt. The background is a soft-focus green field of grass. The text is overlaid on the center of the image.

Python Decorators

Gift or Poison?

Anastasiia Tymoshchuk
Cyren GmbH

A stack of several old, leather-bound books is visible on the left side of the image. The spines of the books are visible, showing various colors like brown, green, and black, and some have gold lettering or designs. The books are stacked vertically, with the top one being a dark green/black color.

Presentation Slides

<https://atymo.me/presentations/GiftOrPoison/>

Code snippets

https://github.com/atymoshchuk/python_tutorials



CYREN

All about security

25B

Security Transactions Daily

1.3B

Users Protected

300M

Threats Blocked Daily

What's in the Talk

1. Functions nature in Python
2. Magic of a Decorator
3. Basics
4. When to use Decorators
5. Examples

A desk setup featuring a silver adjustable desk lamp, three small potted succulents in white pots, a pair of glasses, and a silver laptop. The background is a window with a view of a building.

FUNCTIONS

NATURE

IN PYTHON

FUNCTIONS IN PYTHON ARE OBJECTS

```
def say_hello(name):  
    print("Hello %s!" % name)  
say_hello("EuroPython 2018")  
  
my_func = say_hello  
  
my_func("Awesome EuroPython 2018")
```

run

The background of the image is a beautiful beach scene during the 'golden hour' of sunset or sunrise. The sun is a bright, glowing orb positioned just above the horizon line, casting a warm, golden light across the entire scene. The sky is a mix of soft pinks, oranges, and blues, with wispy white clouds scattered throughout. The ocean is a deep blue, with white-capped waves breaking gently onto the shore. The sand in the foreground is a warm, golden-brown color, and the overall atmosphere is peaceful and magical.

Magic of Decorators




There are two types of decorators

1. Function Decorators

Added in Python 2.4

2. Class Decorators

Added in Python 2.6



Function Decorators

Basic Function Decorator

```
def mydecorator(decorated_func):  
    def wrapped(*args, **kwargs):  
        print("Something happened in decorator!")  
        return decorated_func(*args, **kwargs)  
    return wrapped  
  
@mydecorator  
  
def myfunc(myarg):  
    print("my function", myarg)  
  
def mysecond_func(myarg):  
    print("my second function", myarg)  
  
myfunc('for the Talk')  
mysecond_func = mydecorator(mysecond_func)
```

run

Stacked Function Decorator

```
def first_dec(func):  
    def wrapped(*args, **kwargs):  
        print("Something happened in First decorator!")  
        return func(*args, **kwargs)  
    return wrapped  
  
def second_dec(func):  
    def wrapped(*args, **kwargs):  
        print("Something happened in Second decorator!")  
        return func(*args, **kwargs)  
    return wrapped  
  
@first_dec  
@second_dec  
def myfunc(myarg):  
    print("my function", myarg)
```

run



Class Decorators

Basic Class Decorator

```
class my_decorator:  
    ....  
    # probably a lot of code here  
  
@my_decorator  
class MyClass:  
    def do_something(self):  
        ...
```


Class as a decorator

```
class entry_exit(object):  
  
    def __init__(self, f):  
        self.f = f  
  
    def __call__(self):  
        print("Entering", self.f.__name__)  
  
        self.f()  
        print("Exited", self.f.__name__)  
  
@entry_exit  
def func1():  
    print("inside func1()")  
  
@entry_exit
```

run

A sea turtle is swimming horizontally across the frame from left to right. The turtle's head is on the left, and its tail is on the right. Its front flippers are extended forward, and its back flipper is visible. The turtle's shell is a light brown color with darker brown patterns. The background is a deep blue-green ocean with a bright sun or light source at the top center, creating a strong lens flare and illuminating the scene. The text "Dive into Basics" is overlaid in the center of the image.

Dive into Basics

How decorator works?

```
def mydecorator(decorated_func):  
    def wrapped(*args, **kwargs):  
        print("Before decorated function")  
        result = decorated_func(*args, **kwargs)  
        print("After decorated function")  
        return result  
    return wrapped  
  
@mydecorator  
def myfunc(myarg):  
    """prints some text combined with a string from argument"""  
    print("my function", myarg)  
    return "return value"  
  
r = myfunc('for the Talk')
```

run

Why to use @wraps?

```
from functools import wraps

def mydecorator(f):
    @wraps(f)
    def wrapped(*args, **kwargs):
        print("Before decorated function")
        r = f(*args, **kwargs)

        print("After decorated function")
        return r
    return wrapped

@mydecorator
def myfunc(myarg):
    """prints some text combined with a string from argument"""
    print("my function", myarg)
```

run

A photograph of a modern, curved staircase with light-colored stone steps and walls. The staircase is curved and features a white handrail and a glass railing. The text "When to use Decorators?" is overlaid on the image in a large, black, sans-serif font.

When to use Decorators?

Timing with Function Decorators

```
import time

def timeit(method):

    def timed(*args, **kw):
        ts = time.time()
        result = method(*args, **kw)

        te = time.time()

        print('%r (%r, %r) %2.2f sec' % (method.__name__, args, kw, te-ts))
        return result


    return timed

class Foo(object):
```


run

A photograph of a curved stone staircase with a white railing, set against a light-colored stone wall. The text "Timing with Class Decorators" is overlaid on the image.

Timing with Class Decorators



```
class ImportantStuff(object):  
    @time_this  
    def do_stuff_1(self):  
        ...  
    @time_this  
    def do_stuff_2(self):  
        ...  
    @time_this  
    def do_stuff_3(self):  
        ...
```

```
@time_all_class_methods
class ImportantStuff:
    def do_stuff_1(self):
        ...

    def do_stuff_2(self):
        ...

    def do_stuff_3(self):
        ...
```



```
def time_this(original_function):
    print("decorating")
    def new_function(*args,**kwargs):
        print("starting timer")
        import datetime
        before = datetime.datetime.now()
        x = original_function(*args,**kwargs)
        after = datetime.datetime.now()
        print("Elapsed Time = {0}".format(after-before))
        return x
    return new_function

def time_all_class_methods(Cls):
    class NewCls(object):
        def __init__(self,*args,**kwargs):
            self.oInstance = Cls(*args,**kwargs)
```

run



Examples

How to use decorators in unit testing


```
def add_tests(generator):
    def class_decorator(cls):
        """Add tests to `cls` generated by `generator()`. """
        for test_func, func, input, output in generator:
            test = lambda self, i=input, o=output, f=test_func, f:
                test.__name__ = "test_%s(%r, %r)" % (func.__name__, i
                setattr(cls, test.__name__, test)

            # print("added:", cls, test.__name__, test)
        return cls

    return class_decorator

def test_cases(parameters):
    def t(self, func_to_test, data, result):
```

run




A bit more Magic in Python



What will be the output of the code below?

```
def multipliers():  
    return [lambda x : i * x for i in range(4)]  
  
print [m(2) for m in multipliers()]
```

run



```
def multipliers():  
    return [lambda x : i * x for i in range(4)]  
  
print [m(2) for m in multipliers()]
```

Expectation

[0, 2, 4, 6]

Reality

[6, 6, 6, 6]

WHY?



How to fix?

```
def multipliers():  
    for i in range(4): yield lambda x : i * x
```

or

```
def multipliers():  
    return [lambda x, i=i : i * x for i in range(4)]
```


run



Are Decorators in Python



Gift or Poison?

The background of the image shows a close-up of a desk. A grey pencil lies horizontally across the middle. To its left are several orange-brown pencil shavings. In the bottom right corner, the spine and pages of a thick, light-colored book are visible, with a yellow sticky note tucked between the pages. The overall lighting is bright and soft, creating a clean, professional aesthetic.

Keep in touch
atymoshchuk@icloud.com

P.S. We are hiring!

www.cyren.com/about-cyren/careers