

Introduction to Decorators

Power Up Your Python Code

Geir Arne Hjelle – geirarne@gmail.com

Pycon US May 12, 2021

Exercise 1

Write a decorator that prints BEFORE before calling the decorated function and AFTER afterwards.

```
>>> @before_and_after
... def greet(name):
...     print(f"Hi {name}!")
...
```

```
>>> greet("PyCon")
```

BEFORE

Hi PyCon!

AFTER

Post your code to <https://forms.gle/bW87NFstp24G6gSaA>

Exercise 2

Write a decorator that runs the decorated function twice and returns a 2-tuple with both return values.

```
>>> import random
>>> @do_twice
... def roll_dice():
...     return random.randint(1, 6)
...
>>> roll_dice()
(3, 1)
```

Exercise 3

Write a decorator that stores references to decorated functions in a dictionary.

```
>>> FUNCTIONS = {}
>>> @register
... def roll_dice():
...     return random.randint(1, 6)
...
>>> FUNCTIONS
{'roll_dice': <function __main__.roll_dice()>}
>>> FUNCTIONS["roll_dice"]()
2
```

Exercise 4

Write a decorator that repeatedly calls the decorated function as long as it raises an exception.

```
>>> @retry
... def only_roll_sixes():
...     number = random.randint(1, 6)
...     if number != 6:
...         raise ValueError(number)
...     return number
...
>>> only_roll_sixes()
6
```

Exercise 5 (hard)

Rewrite `@retry` so that it only retries on specific, user-defined exceptions.

```
>>> @retry(ValueError)
... def calculation():
...     number = random.randint(-5, 5)
...     if abs(1 / number) > 0.2:
...         raise ValueError(number)
...     return number
...
>>> calculation()
# -5, 5, or ZeroDivisionError
```

Exercise 6

Adapt `@retry` so that it only tries a maximum of `max_retries` times.

```
>>> @retry(max_retries=3)
... def only_roll_sixes():
...     number = random.randint(1, 6)
...     if number != 6:
...         raise ValueError(number)
...     return number
...
>>> only_roll_sixes()
# 6 or ValueError
```

Exercise 6

Adapt `@retry` so that it only tries a maximum of `max_retries` times.

```
>>> @retry(max_retries=3)
... def only_roll_sixes():
...     number = random.randint(1, 6)
...     if number != 6:
...         raise ValueError(number)
...     return number
...
>>> only_roll_sixes()
# 6 or ValueError
```

Challenge: Can you make `@retry` count all retries across several function calls?

Exercise 7

Write a class-based @Retry decorator that keeps track of the number of retries across all function calls.

```
>>> @Retry
... def only_roll_sixes():
...     # Same as before
...
>>> only_roll_sixes()
Retry attempt 1
Retry attempt 2
6
>>> only_roll_sixes()
Retry attempt 3
6
```

Other Resources

- Tutorial page:
 - github.com/gahjelle/decorators_tutorial
 - Material and video from 2020 course
 - Code from 2021 will be added after the course
- Real Python article:
 - realpython.com/primer-on-python-decorators
- Real Python video course (paywall):
 - realpython.com/courses/python-decorators-101
- PEP 318: Decorators for Functions and Methods
 - www.python.org/dev/peps/pep-0318
- Awesome Python Decorator
 - github.com/lord63/awesome-python-decorator