# Декораторы

In [1]:
```
def decorator(func):
    return func


@decorator
def decorated():
    print('Hello!')
```

In [2]:
```
decorated = decorator(decorated)
```

In [ ]:
```

```

In [3]:
```
def decorator(func):
    def new_func():
        pass
    return new_func


@decorator
def decorated():
    print('Hello!')


decorated()
```

In [4]:
```
print(decorated.__name__)
```

```
new_func
```

## Написать декоратор, который записывает в лог результат декорируемой функции

In [5]:
```
import functools


def logger(func):
    @functools.wraps(func)
    def wrapped(*args, **kwargs):
        result = func(*args, **kwargs)
        with open('log.txt', 'w') as f:
            f.write(str(result))

        return result
    return wrapped


@logger
def summator(num_list):
    return sum(num_list)


print('Summator: {}'.format(summator([1, 2, 3, 4])))

print(summator.__name__)
```

```
Summator: 10
summator
```

## Написать декоратор с параметром, который записывает лог в указанный файл

In [7]:
```
def logger(filename):
    def decorator(func):
        def wrapped(*args, **kwargs):
            result = func(*args, **kwargs)
            with open(filename, 'w') as f:
                f.write(str(result))
            return result
        return wrapped
    return decorator


@logger('new_log.txt')
def summator(num_list):
    return sum(num_list)


# summator = logger('log.txt')(summator)


summator([1, 2, 3, 4, 5, 6])

with open('new_log.txt', 'r') as f:
    print(f.read())
```

```
21
```

In [8]:
```
def first_decorator(func):
    def wrapped():
        print('Inside first_decorator product')
        return func()
    return wrapped


def second_decorator(func):
    def wrapped():
        print('Inside second_decorator product')
        return func()
    return wrapped
```

In [9]:
```
@first_decorator
@second_decorator
def decorated():
    print('Finally called...')


# decorated = first_decorator(second_decorator(decorated))


decorated()
```

```
Inside first_decorator product
Inside second_decorator product
Finally called...
```

In [ ]:
```
```

In [10]:
```
def bold(func):
    def wrapped():
        return "<b>" + func() + "</b>"
    return wrapped


def italic(func):
    def wrapped():
        return "<i>" + func() + "</i>"
    return wrapped


@bold
@italic
def hello():
    return "hello world"


# hello = bold(italic(hello))


print(hello())
```

```
<b><i>hello world</i></b>
```