Декоратор как класс

Реализация декораторов

```python
class CountCalls:
    def __init__(self, func: Callable) -> None:
        functools.update_wrapper(self, func)
        self.func = func
        self.num_calls = 0

    def __call__(self, *args, **kwargs) -> Any:
        self.num_calls += 1
        print("Вызов номер {num} функции {func}".format(
            num=self.num_calls, func=self.func.__name__)
        )
        return self.func(*args, **kwargs)
```

Класс

```python
def count_calls(func: Callable) -> Callable:

    @functools.wraps(func)
    def wrapper(*args, **kwargs):
        wrapper.num_calls += 1
        result = func(*args, **kwargs)
        print("Вызов номер {num} функции {func}".format(
            num=wrapper.num_calls, func=func.__name__)
        )
        return result
    wrapper.num_calls = 0
    return wrapper
```

Функция

Декоратор-класс используется **крайне редко**. Почти всегда используется реализация функцией

Итоги модуля

- **@contextmanager**
- **def** timer_with_precision(precision):

  def timer_decorator(...)
- **@createtime**

  class Functions:

  . . . . .

  my_funcs = Functions(1000)
- **@for_all_methods(timer)**

  dir, getattr, setattr
- **@CountCalls**

Спасибо за внимание