3.2 Метод __call__. Функторы и классы-декораторы

9 из 11 шагов пройдено 18 из 25 баллов получено

Видео-разбор подвига (решение смотреть только после своей попытки): https://youtu.be/Ac0s64-XEdE (https://youtu.be/Ac0s64-XEdE)

Подвиг 7. Необходимо объявить класс-декоратор с именем HandlerGET, который будет имитировать обработку GET-запросов на стороне сервера. Для этого сам класс HandlerGET нужно оформить так, чтобы его можно было применять к любой функции как декоратор. Например:

```
@HandlerGET
def contact(request):
    return "Сергей Балакирев"
```

Здесь request - это произвольный словарь с данными текущего запроса, например, такой: {"method": "GET", "url": "contact.html"}. А функция должна возвращать строку.

Затем, при вызове декорированной функции:

```
res = contact({"method": "GET", "url": "contact.html"})
```

должна возвращаться строка в формате:

"GET: <данные из функции>"

В нашем примере - это будет:

"GET: Сергей Балакирев"

Если ключ method в словаре request отсутствует, то по умолчанию подразумевается GET-запрос. Если же ключ method принимает другое значение, например, "POST", то декорированная функция contact должна возвращать значение None.

Для реализации имитации GET-запроса в классе HandlerGET следует объявить вспомогательный метод со следующей сигнатурой:

```
def get(self, func, request, *args, **kwargs): ...
```

Здесь func - ссылка на декорируемую функцию; request - словарь с переданными данными при вызове декорированной функции. Именно в этом методе следует формировать возвращаемую строку в указанном формате:

"GET: Сергей Балакирев"

P.S. В программе достаточно объявить только класс. Ничего на экран выводить не нужно.