SISTEMAS DISTRIBUÍDOS
@DECORATORS

Márcio Martins Ferreira Júnior

O QUE SÃO DECORATORS?

Um decorator é uma função “embrulha” outra função sem modificar seu comportamento, mas podendo executar outras coisas antes ou depois dela

Isso torna mais eficiente a aplicação de lógicas comuns, como controle de acesso, logging e medição de desempenho

```
@nome_do_decorator
def minha_funcao():
    print("Sarve, mundo!")
```

POR QUE USAR?


```python
def ping(func):

  def wrapper():
    print('ping')
    func()

  return wrapper

@ping
def pong():
  print("pong")

pong()
```

Em python, funções são objetos de primeira classe, o que possibilita que sejam passadas e retornadas por outras funções, então o decorator usa desse recurso para receber uma função, rodar um código extra e retornar uma nova função

- Reutilização de código
- Separação de responsabilidades
- Facilita a manutenção do código
- Sintaxe clara e expressiva

Aplicando decoradores

```python
def decorator1(func):
    def wrapper(*args, **kwargs):
        print("Decorator 1: Antes da função")
        result = func(*args, **kwargs)
        print("Decorator 1: Depois da função")
        return result
    return wrapper

def decorator2(func):
    def wrapper(*args, **kwargs):
        print("Decorator 2: Antes da função")
        result = func(*args, **kwargs)
        print("Decorator 2: Depois da função")
        return result
    return wrapper

@decorator1
@decorator2
def minha_funcao():
    print("Executando a função principal!")

minha_funcao()
```

Podemos empilhar vários decorators, o que possibilita modificar não apenas o comportamento de funções e métodos, mas também dos próprios decoradores.

```
Decorator 1: Antes da função
Decorator 2: Antes da função
Executando a função principal!
Decorator 2: Depois da função
Decorator 1: Depois da função
```

Parâmetros em decorators

É possível adicionar parâmetros aos decorators, que podem ser usados na sua lógica interna.

```
Tempo da Função 1 executada em 1.0034 segundos
Tempo da Função 2 executada em 2.0016 segundos
Tempo da Função 3 executada em 3.0031 segundos
batatinha executada em 5.0051 segundos
```



```python
import time

def calcular_tempo(nome_funcao):
    def decorator(func):
        def wrapper(*args, **kwargs):
            start_time = time.time()
            result = func(*args, **kwargs)
            end_time = time.time()
            tempo_execucao = end_time - start_time
            print(f"{nome_funcao} executada em {tempo_execucao:.4f} segundos")
            return result
        return wrapper
    return decorator

@calcular_tempo("Tempo da Função 1")
def funcao1():
    time.sleep(1)

@calcular_tempo("Tempo da Função 2")
def funcao2():
    time.sleep(2)

@calcular_tempo("Tempo da Função 3")
def funcao3():
    time.sleep(3)

@calcular_tempo("batatinha")
def funcao4():
    time.sleep(5)


funcao1()
funcao2()
funcao3()
funcao4()
```

Casos de uso

```python
def logger(func):
  def wrapper(*args, **kwargs):
    print(f"Função: {func.__name__}")
    print(f"Argumentos: {args}")
    res = func(*args, **kwargs)
    print(f"Resultado: {res}")
    return res
  return wrapper

@logger
def diz_oi(n1, n2):
  print(f"{n1} cumprimentou {n2}")
  return

diz_oi("João", "Maria")

print("-----------------------------------")

@logger
def multiplica(a, b):
  return a * b

multiplica(10, 20)
```

```
Função: diz_oi
Argumentos: ('João', 'Maria')
João cumprimentou Maria
Resultado: None
```

```
Função: multiplica
Argumentos: (10, 20)
Resultado: 200
200
```

```python
def checar_permissao(user):
  def decorator(func):
    def wrapper(*args, **kwargs):
      if user == "admin":
        return func(*args, **kwargs)
      else:
        print("Acesso Negado!.")
    return wrapper
  return decorator

# ---------------------------------

@checar_permissao("admin")
def acessar_sistema():
  print("Bem-vindo ao sistema!")

acessar_sistema()

# ---------------------------------

@checar_permissao("usuario")
def acessar_sistema():
  print("Bem-vindo ao sistema!")

acessar_sistema()
```

```
Bem-vindo ao sistema!
Acesso Negado!.
```

link do colab

DÁ LIKE
COMPARTILHA
INSCREVE NO CANAL
ATIVA O SININHO