# DEKORATORY I METAPROGRAMOWANIE

Zadania

## Zadanie 1.

Napisz dekorator `@log_call`, który wypisuje komunikat za każdym razem, gdy dekorowana funkcja zostanie wywołana: `Wywołano funkcję <nazwa_funkcji>`

## Zadanie 2.

Napisz dekorator `@limit_calls(n)`, który pozwala wywołać dekorowaną funkcję maksymalnie `n` razy. Po przekroczeniu limitu każde kolejne wywołanie powinno wypisać komunikat: `Limit wywołań funkcji został przekroczony.`

## Zadanie 3.

Napisz dekorator `@delay(seconds)`, który wstrzymuje wykonanie dekorowanej funkcji o podaną liczbę sekund. Funkcja `time.sleep()` powinna być użyta w dekoratorze.

## Zadanie 4.

Napisz dekorator `@restrict_types(allowed_types)`, który sprawdza, czy wszystkie pozycyjne argumenty przekazane do dekorowanej funkcji są zgodne z podanymi typami. Jeśli któryś argument ma niewłaściwy typ, dekorator powinien zgłosić wyjątek `TypeError` z komunikatem: `"Argument <nazwa_argumentu> ma nieprawidłowy typ.`
`Oczekiwano: <oczekiwany_typ>."`

## Zadanie 5.

Napisz dekorator `@dynamic_cache`, który działa jak klasyczny mechanizm memoizacji — zapamiętuje wyniki funkcji dla danych argumentów — ale dodatkowo przy pierwszym wywołaniu wypisuje typy przekazanych argumentów oraz typ zwróconej wartości. Dla kolejnych wywołań z tymi samymi argumentami dekorator powinien zwracać wynik z pamięci podręcznej bez ponownego logowania.