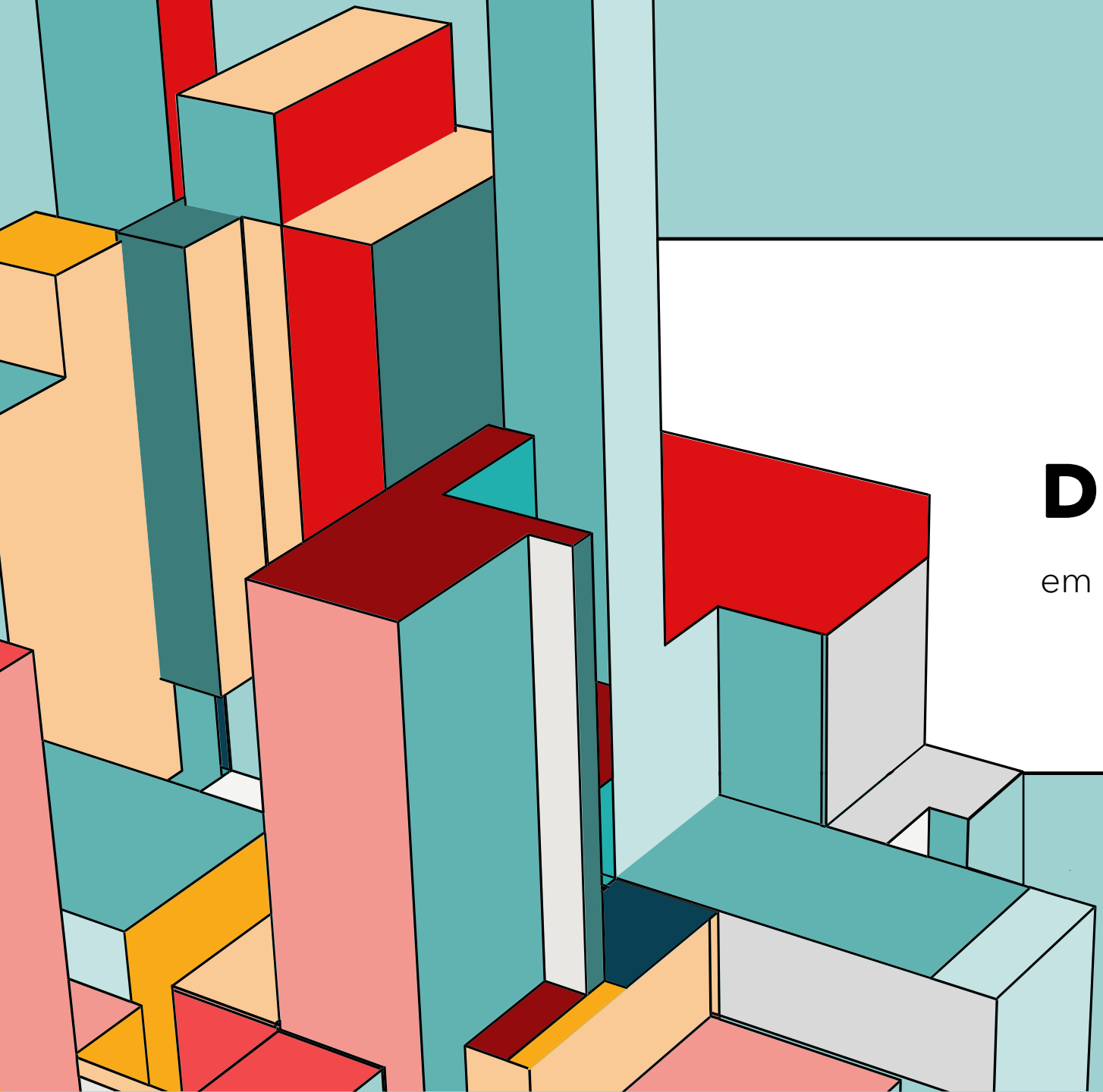
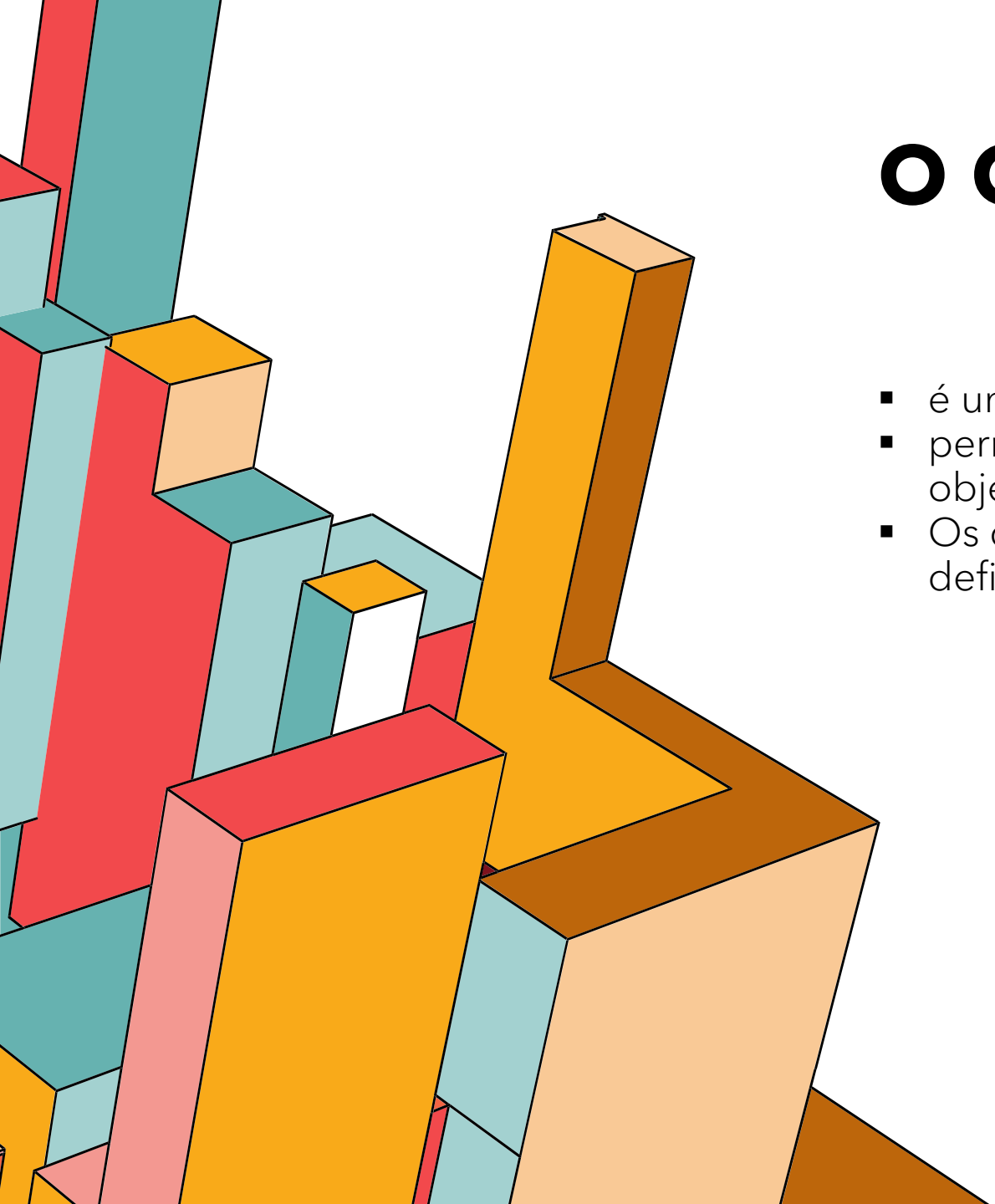


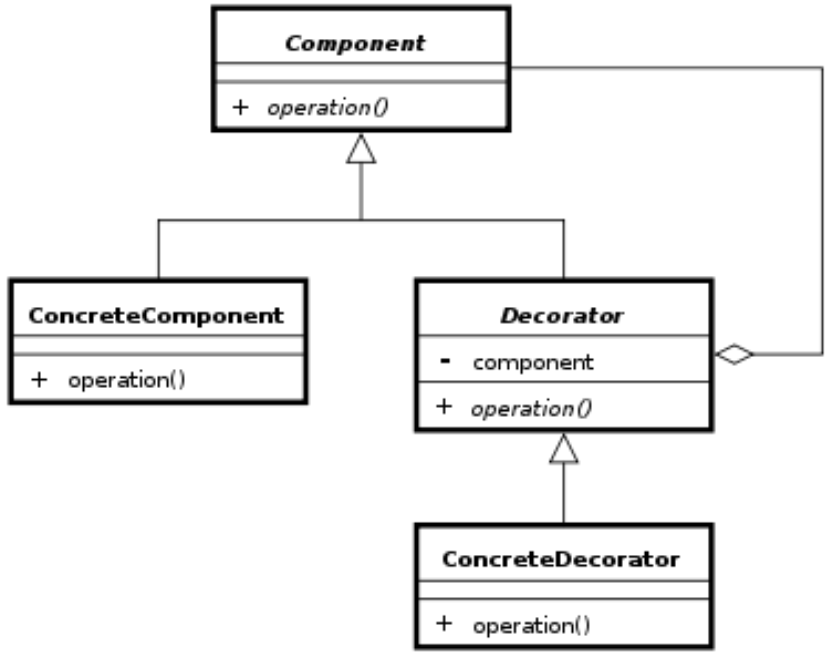
DECORATORS

em Python



O QUE É UM DECORATOR

- é um padrão de design em Python
- permite ao usuário adicionar novas funcionalidades a um objeto existente sem modificar sua estrutura.
- Os decoradores geralmente são chamados antes da definição de uma função que você deseja decorar.



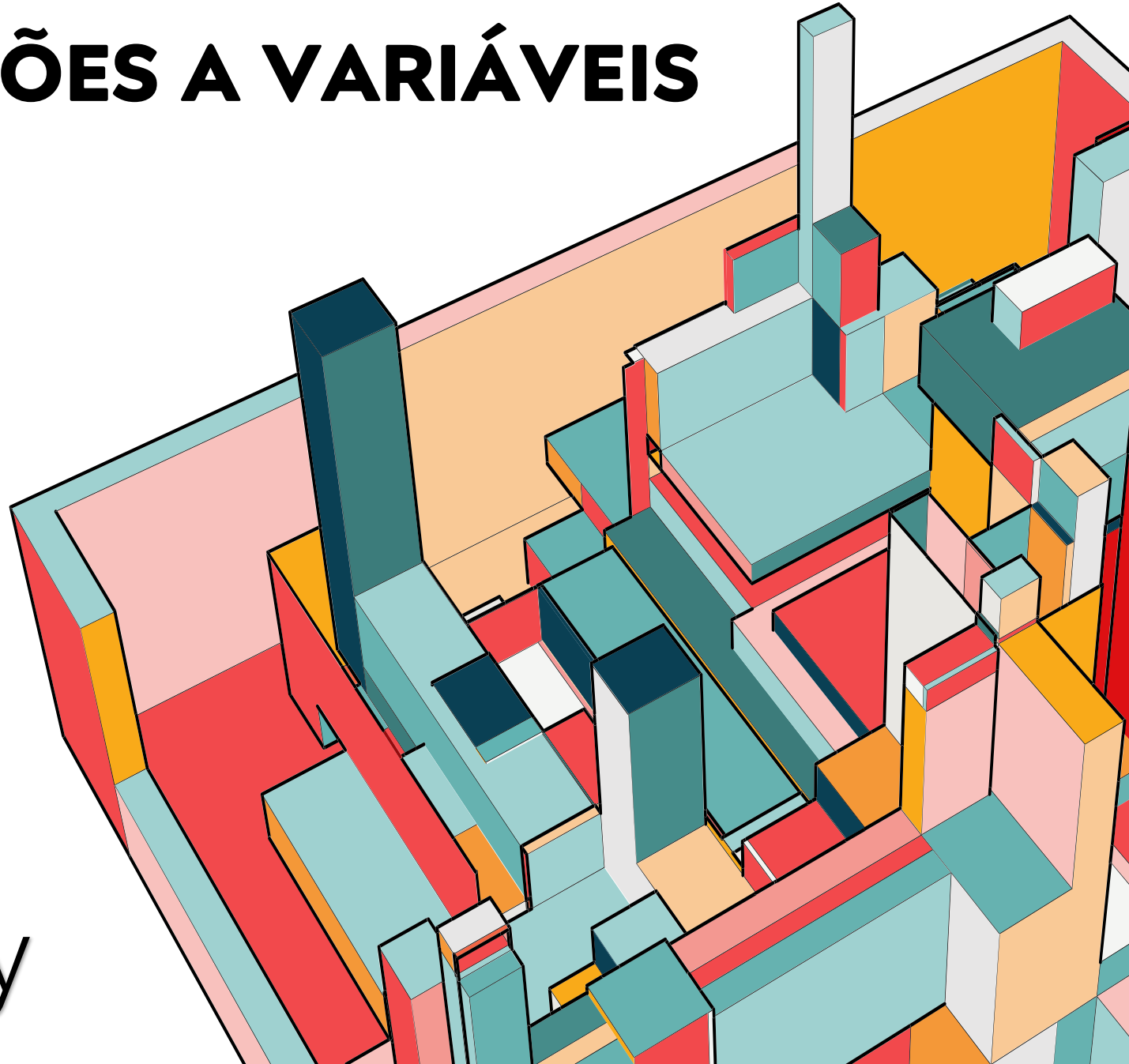
ATRIBUINDO FUNÇÕES A VARIÁVEIS

Funções em Python suportam operações como:

- serem passados como um argumento,
- serem retornados por uma função,
- serem modificados e atribuídos a uma variável.

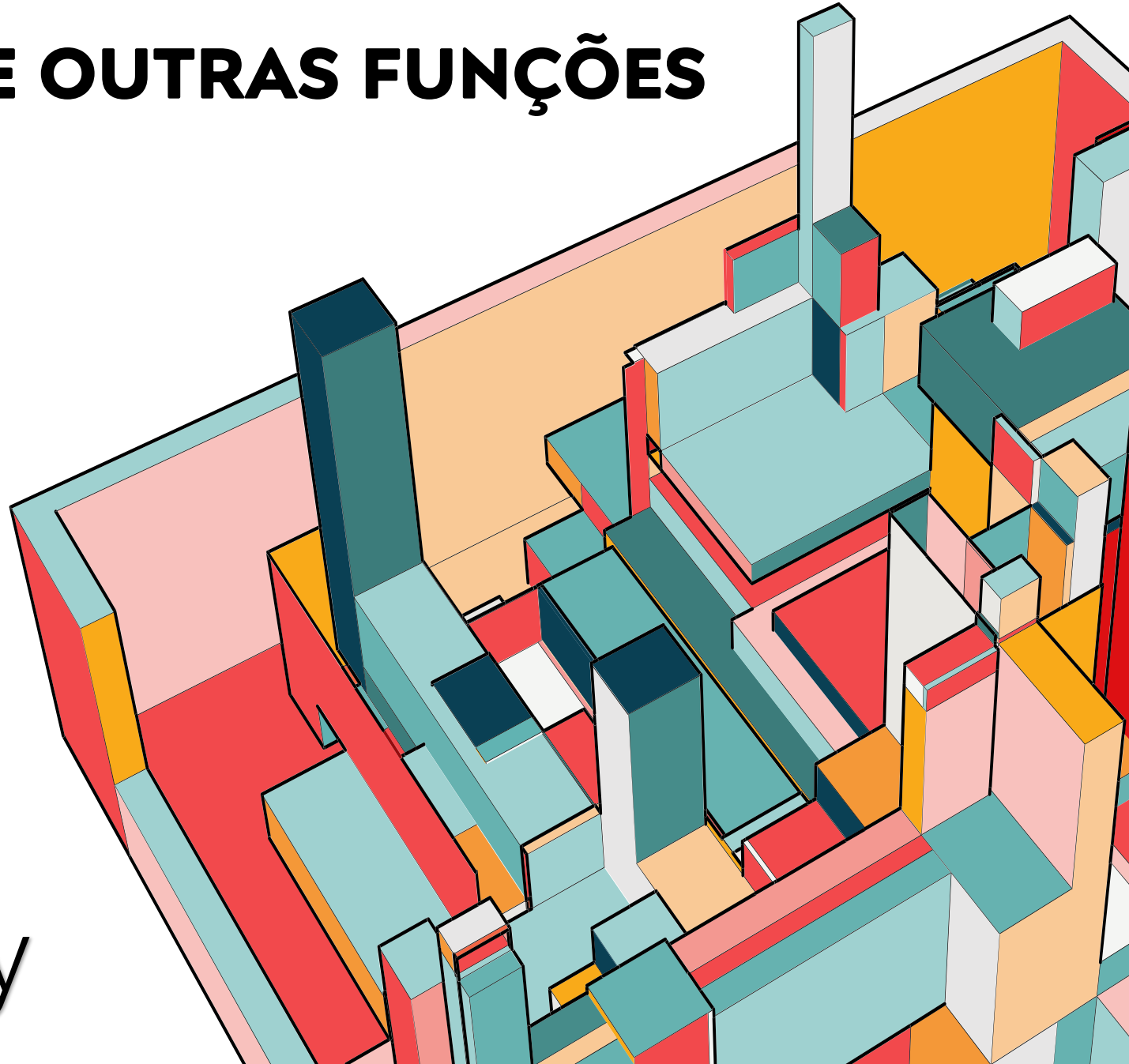
O Python permite que você use decoradores de maneira simples com o símbolo @, às vezes chamado de sintaxe "torta".

Exemplo-01.py



FUNÇÕES DENTRO DE OUTRAS FUNÇÕES

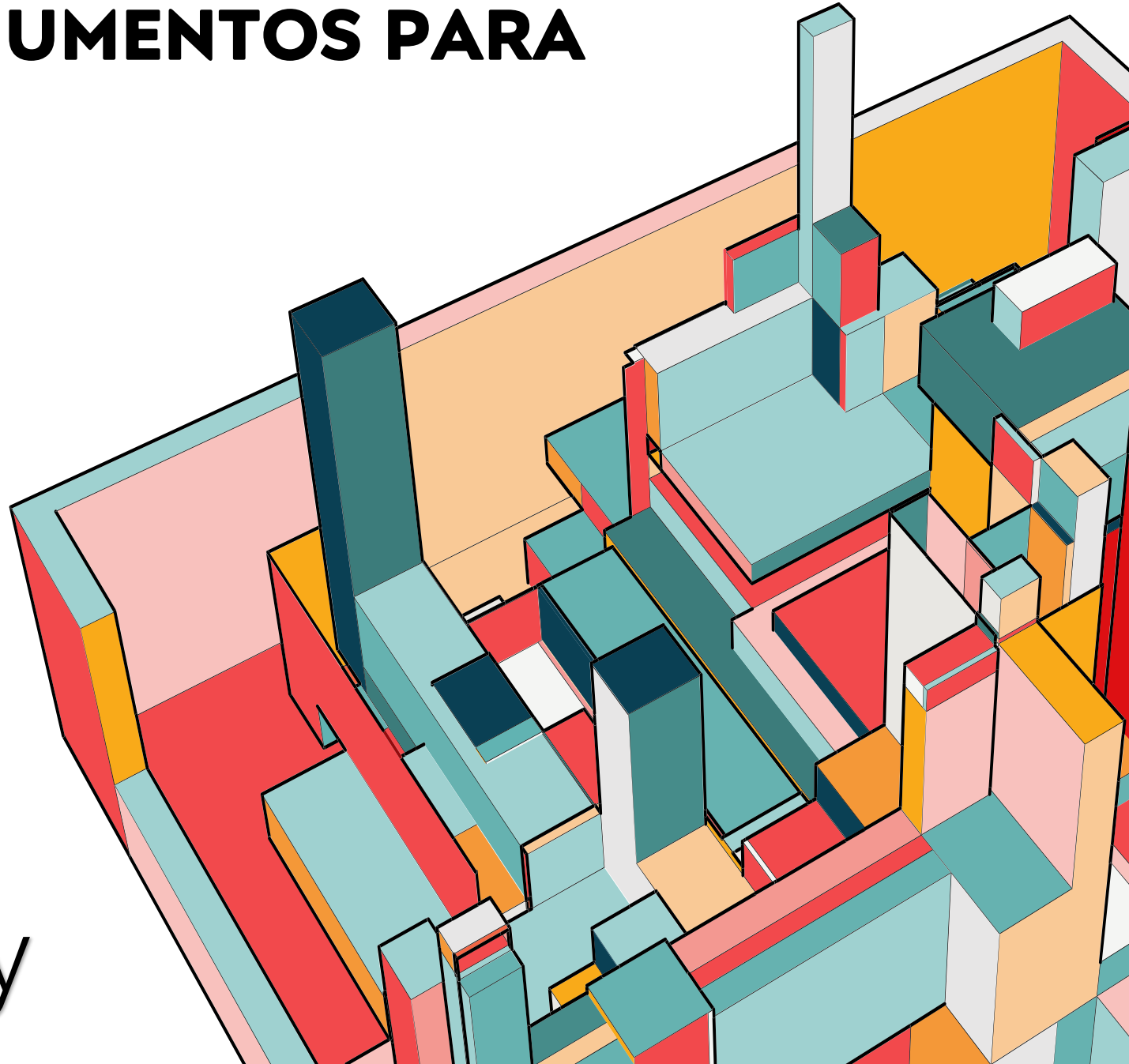
A seguir, ilustraremos como você pode definir uma função dentro de outra função em Python. Fique comigo, logo veremos como tudo isso é relevante na criação e compreensão de decoradores em Python.



Exemplo-02.py

FUNÇÕES COMO ARGUMENTOS PARA OUTRAS FUNÇÕES

As funções também podem ser passadas como parâmetros para outras funções.



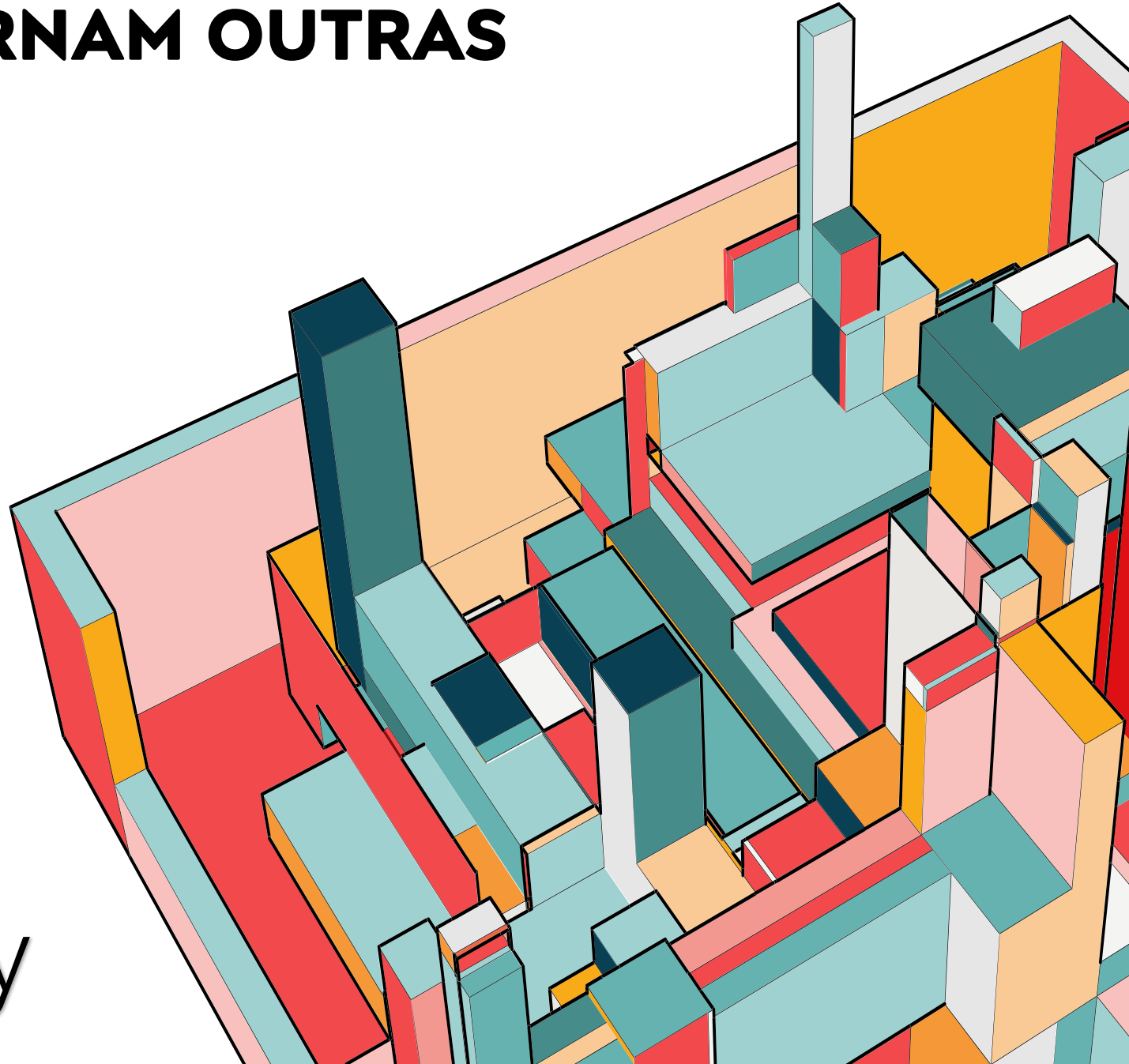
Exemplo-03.py

FUNÇÕES QUE RETORNAM OUTRAS FUNÇÕES

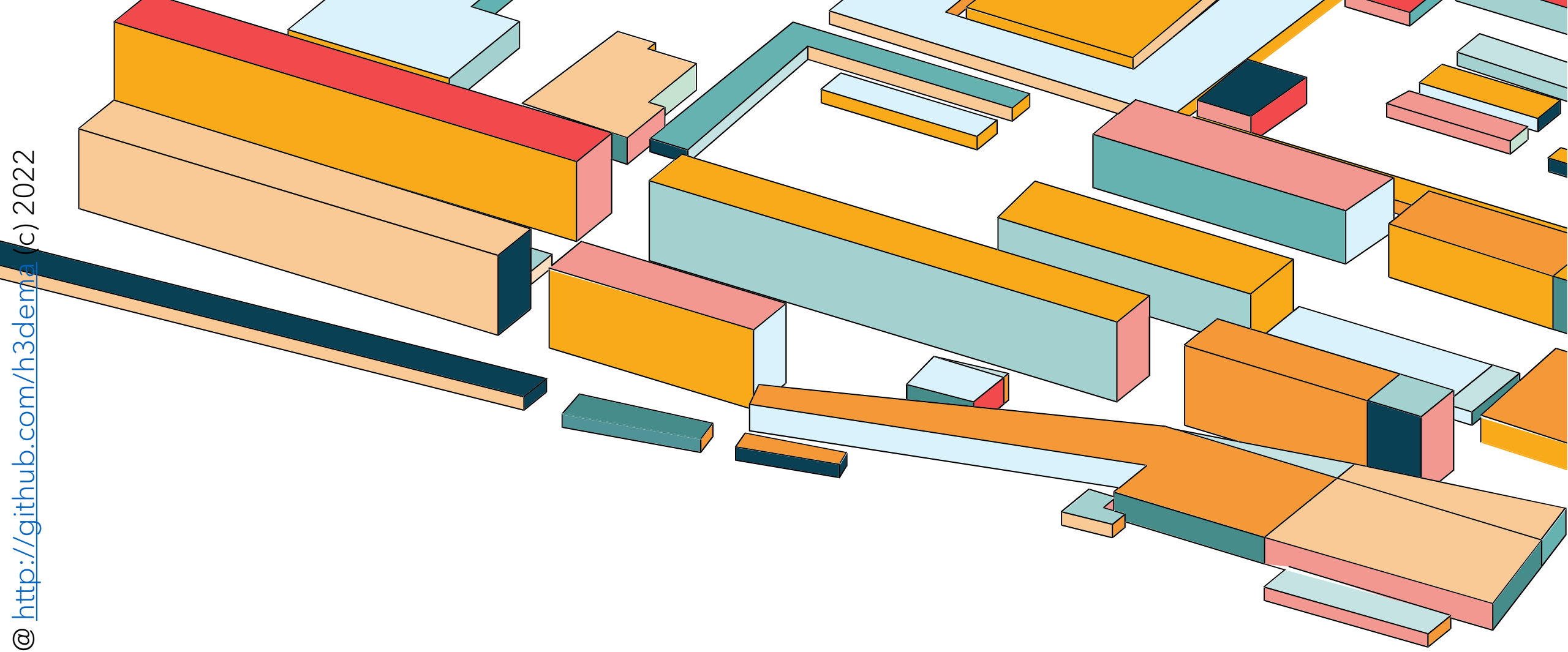
Uma função também pode gerar outra função.

Python permite que uma função aninhada acesse o escopo externo da função envolvente.

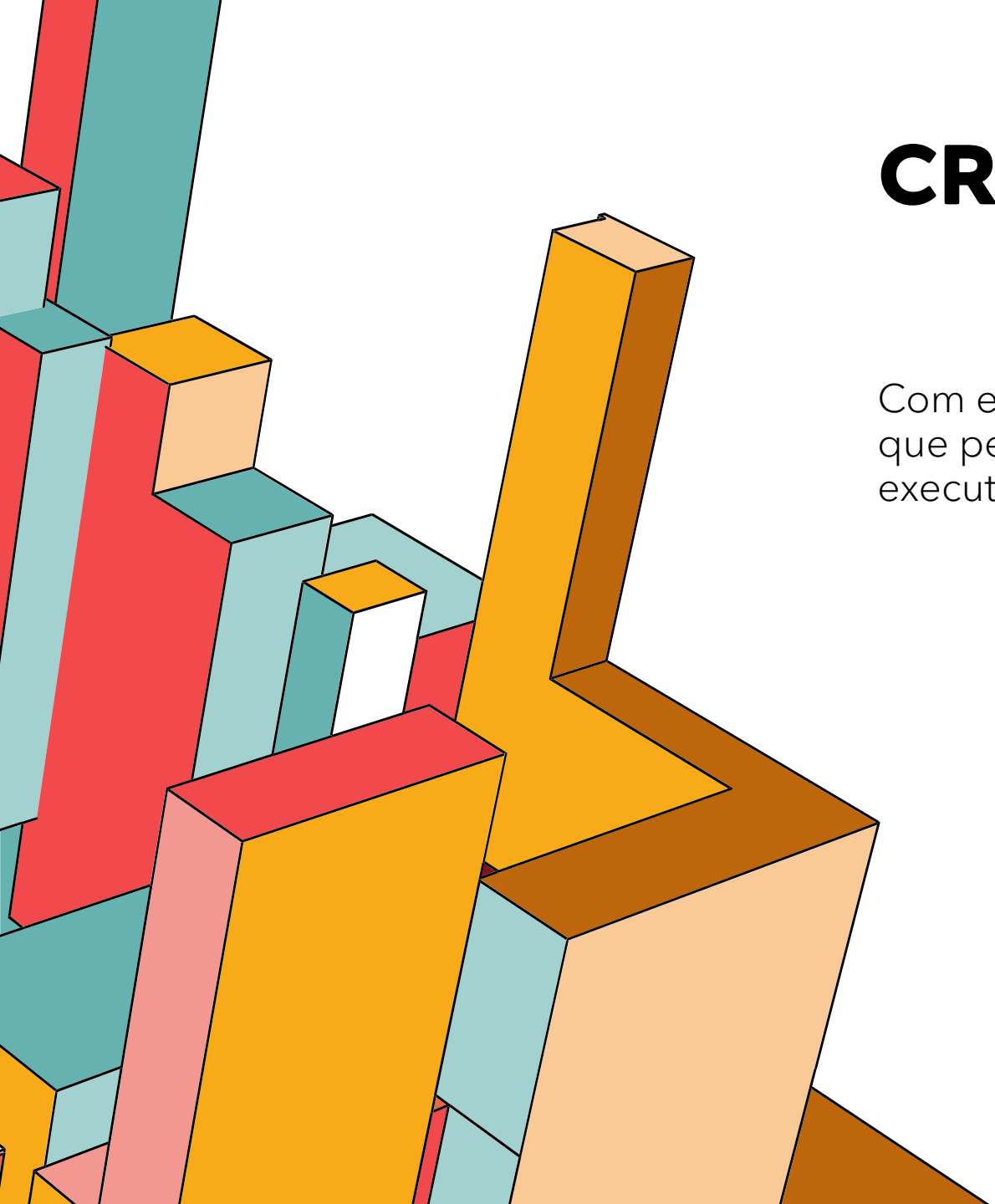
- Este é um conceito crítico para os decorators
- É um padrão conhecido como Closure.



Exemplo-04.py



DECORATORS



CRIANDO DECORADORES

Com esses pré-requisitos resolvidos, vamos criar um decorador que permite criar uma função que depois de chamada, é executada a cada “x” segundos.

```
@timely(periodo=x)
my_func(*args, **kwargs)
```

PASSOS