Day10_PyMySQL_闭包_装饰器

- PyMySQL ㊼
- 闭包
  - 是一种特殊的函数定义形式
  - 定义的规则：
    1. 外函数中定义一个内函数
    2. 内函数中使用外函数中的局部变量
    3. 外函数返回内函数的引用(函数名)
  - 作用
    - 提高代码复用率
    - 隐藏代码实现细节
  - 本质
    - 闭包就是内函数的引用+外函数的执行环境
- 装饰器
  - 作用
    - 用来为已有函数在不改变已有函数代码和调用方式的前提下,为该函数添加额外的功能
  - 原理
    - 利用闭包,将被装饰的函数做为参数传递闭包的外函数的参数中, 让被装饰函数重新指向外函数的返回值(内函数的引用)
    - 让在使用被装饰函数名进行调用函数时,实际调用的就是闭包中的内函数
  - 结论
    - 无论如何使用装饰器,被装饰函数永远指向内函数
  - 语法糖
    - @闭包外函数名
    - 注意: 当使用闭包来定义装饰器时,外函数必须且只能有一个参数
  - 通用装饰器

    -
      ```
      def  outer(func):
      def inner(*args, **kwargs):
      # 装饰代码 ...
      ret = func(*args, **kwargs)
      # 装饰代码...
      return ret
      return inner
      ```

    -
      ```
      @outer
      def show():
      pass
      ```
      - show = outer(show)
  - 类装饰器
    - 使用类也可以做为装饰器使用
      - 但是的格式相同

    -
      ```
      @Decorate
      def show():
      pass
      ```
      - 如果是闭包
        - 返回的是内函数引用
      - 如果是类
        - 返回的是一个Decorate类的对象
    - 实现

      -
        ```
        class Decorate(object):
        def __init__(self, func):
        self.__func = func

        def __call__(self, *args, **kwargs):
        # 装饰代码 ...
        ret = func(*args, **kwargs)
        # 装饰代码...
        return ret
        ```
      - 原理:
        - 当使用一个对象加括号形式时,会自动调用 该对象的所属类中实现的 __call__方法

        -
          ```
          obj = Decorate()
          obj()
          ```
        - print(obj)

XMind:ZEN | 试用模式