

Assessment on Python Decorators:

In Python, a **decorator** is a special function that allows us to modify or extend the behavior of another function — without changing its actual code.

For example :

```
practise.py X
C: > Users > DELL > Desktop > practise.py > ...
1  def add_decorator(func):
2      def wrapper(a, b):
3          print("Starting calculation...")
4          result = func(a, b)
5          print("Calculation finished.")
6          return result
7      return wrapper
8  @add_decorator
9  def add(a, b):
10     return a + b
11  print(add(10, 5))

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SPELL CHECKER

PS C:\Users\DELL> & C:/Users/DELL/AppData/Local/Programs/Python/Python314/python.exe c:/Users/DELL/Desktop/practise.py
Starting calculation...
Calculation finished.
15
PS C:\Users\DELL>
```

=> When `add(10, 5)` is called, it doesn't directly jump into `add`. Instead, it first runs the decorator's function which is `@add_decorator`.

```
practise.py X
C: > Users > DELL > Desktop > practise.py > ...
1  def android_decorator(version):
2      def actual_decorator(func):
3          def wrapper(*args, **kwargs):
4              print(f"Android {version} starting compatibility check...")
5              result = func(*args, **kwargs)
6              print(f"Android {version} finished compatibility check.")
7              return result
8          return wrapper
9      return actual_decorator
10
11  @android_decorator("4.0")
12  @android_decorator("3.0")
13  def check_compatibility(app_name, required_version):
14      print(f"Checking if '{app_name}' is compatible with Android {required_version}...")
15
16  check_compatibility("Snapchat", "3.0")
17

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SPELL CHECKER

PS C:\Users\DELL> & C:/Users/DELL/AppData/Local/Programs/Python/Python314/python.exe c:/Users/DELL/Desktop/practise.py
Android 4.0 starting compatibility check...
Android 3.0 starting compatibility check...
Checking if 'Snapchat' is compatible with Android 3.0...
Android 3.0 finished compatibility check.
Android 4.0 finished compatibility check.
PS C:\Users\DELL>
```

=> When you call → `check_compatibility("Snapchat", "3.0")`

=> Python reads it from the bottom up. That means, The function `check_compatibility` is first passed into `android_decorator("3.0")`.

=> Then it takes that result and **wraps** it again with `android_decorator("4.0")`.

=> Android 3.0's wrapper starts and prints, Android 3.0 starting compatibility check...

=> Inside Android 3.0's wrapper, the real function `check_compatibility` finally runs and prints, Checking if 'Snapchat' is compatible with Android 3.0...

=> Once the inner function finishes, Android 3.0's wrapper continues. It prints, Android 3.0 finished compatibility check. Then, it gives back its result to Android 4.0.

=> Now, control comes back to Android 4.0's wrapper. After receiving the inner result, it prints: Android 4.0 finished compatibility check.