Day53_functions_to_decorators

July 28, 2025

Python Functions, Lambda & Decorators

1 Python Functions

A function is a reusable block of code that performs a specific task.

[1]:
```
# Basic Example
def greet(name):
    return f"Hello, {name}!"

print(greet("Akshay"))  # Output: Hello, Akshay
```

```
Hello, Akshay!
```

Example ## Login System Used in websites to verify login info.

[4]:
```
def login(username, password):
    if username == "admin" and password == "123":
        return "Login successful"
    return "Invalid credentials"

print(login("admin", "123"))  # Output: Login successful

print(login("admin", "1223"))# Output: Invalid credentials
```

```
Login successful
Invalid credentials
```

1.1 Pizza Order Function

Apps like Domino's, Zomato use similar logic to take and customize orders.

[5]:
```
def order_pizza(size, toppings):
    return f"You ordered a {size} pizza with {', '.join(toppings)}!"

print(order_pizza("large", ["cheese", "pepperoni", "olives"]))
```

```
You ordered a large pizza with cheese, pepperoni, olives!
```

1.2 Movie Ticket Booking

Platforms like BookMyShow, PVR use similar functions for ticketing.

[6]:
```python
def book_ticket(name, movie, seats):
    return f"{name} booked {seats} seat(s) for '{movie}'"

print(book_ticket("Akshay", "Avengers", 2))
```

```
Akshay booked 2 seat(s) for 'Avengers'
```

2 Lambda Functions

A lambda function is a small, anonymous function used for short operations.

[7]:
```python
# Basic lambda example
add = lambda a, b: a + b
print(add(5, 3))  # Output: 8
```

```
8
```

2.1 Get Even Numbers

Used in filtering user data, like active vs inactive users.

[8]:
```python
nums = [10, 15, 20, 25, 30]
evens = list(filter(lambda x: x % 2 == 0, nums))
print(evens)  # Output: [10, 20, 30]
```

```
[10, 20, 30]
```

2.2 Sort by Salary

Used in HR systems to sort records.

[9]:
```python
employees = [("Amit", 50000), ("Neha", 65000), ("Ravi", 40000)]
employees.sort(key=lambda x: x[1])  # Sort by salary
print(employees)
```

```
[('Ravi', 40000), ('Amit', 50000), ('Neha', 65000)]
```

3 Python Decorators

A decorator is a special function in Python that allows you to add new features to an existing function without changing its code.

You “wrap” your original function inside another function — and this wrapper can do something before or after the original function runs.

Think of it like this:

Imagine you have a plain cake (your function).

You want to add chocolate on top (extra feature) — but without baking the cake again.

So you “wrap” the cake in chocolate — this is what a decorator does.

[10]:
```
def decorator_func(func):
    def wrapper():
        print("Before function")
        func()
        print("After function")
    return wrapper

@decorator_func
def welcome():
    print("Welcome to the system!")

welcome()
```

```
Before function
Welcome to the system!
After function
```

3.1 Logging Every Function Call

Used in payment systems to log every transaction.

[11]:
```
def log_decorator(func):
    def wrapper(*args, **kwargs):
        print(f"LOG: Running {func.__name__} with {args}")
        return func(*args, **kwargs)
    return wrapper

@log_decorator
def pay(amount):
    print(f"Paid  {amount}")

pay(200)
```

```
LOG: Running pay with (200,)
Paid  200
```

3.2 Access Control (Login Required)

Used on websites like Amazon, Gmail to protect user dashboards.

[12]:
```
def login_required(func):
    def wrapper(user):
        if user.get("logged_in"):
            return func(user)
        else:
            return "  Please log in first."
    return wrapper

@login_required
def view_dashboard(user):
    return f"Welcome {user['name']} to your dashboard."

user1 = {"name": "Akshay", "logged_in": True}
user2 = {"name": "Swara", "logged_in": False}

print(view_dashboard(user1))  #  Welcome message
print(view_dashboard(user2))  #  Login required
```

```
Welcome Akshay to your dashboard.
 Please log in first.
```