

# Class Decorators

---



**Robert Smallshire**

COFOUNDER - SIXTY NORTH

@robsmallshire



**Austin Bingham**

COFOUNDER - SIXTY NORTH

@austin\_bingham

# Overview



Programmatically **transform** class definitions

Similar **mechanism** to function decorators

Metaprogramming - treating **programs as data**

Overlap with the **capabilities** of metaclasses

Less powerful than metaclasses, but **easier to use** 📄

**Class decorators often introspect the decorated class**

# Core Python: Introspection

on



# PLURALSIGHT

# Decorator Syntax

@decorator

def decorated\_function():

# . . .

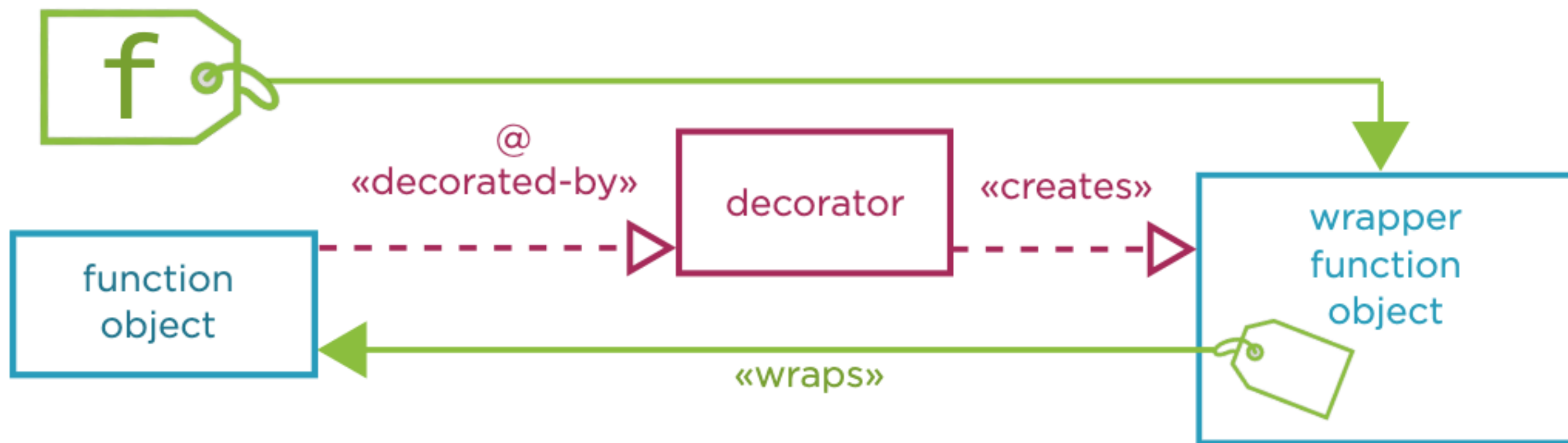
# Core Python: Functions and Functional Programming

on



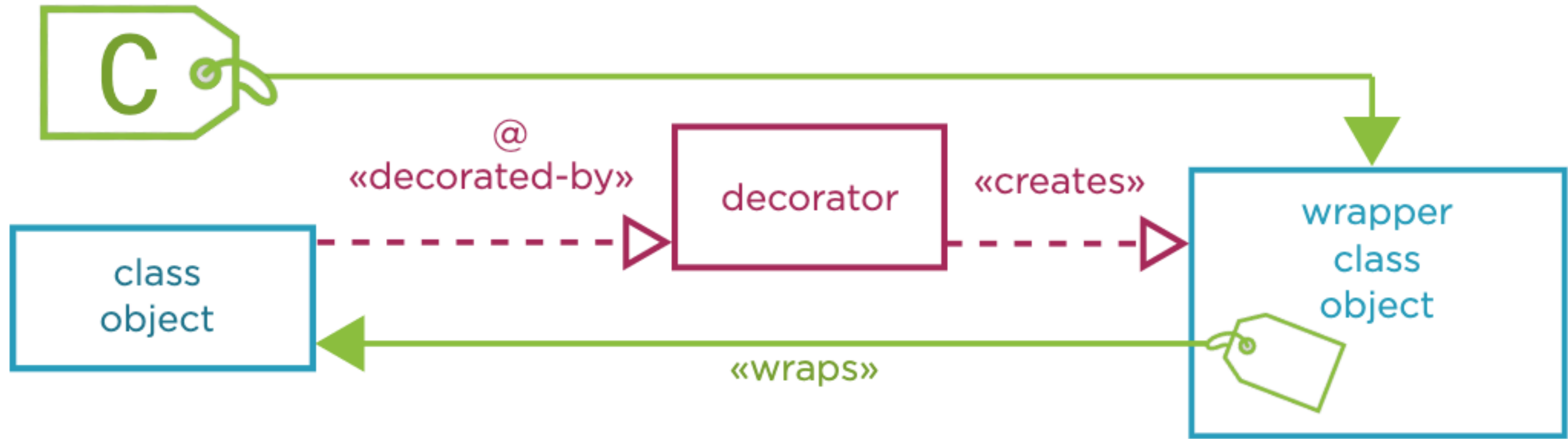
**PLURALSIGHT**

# Decorator Recap



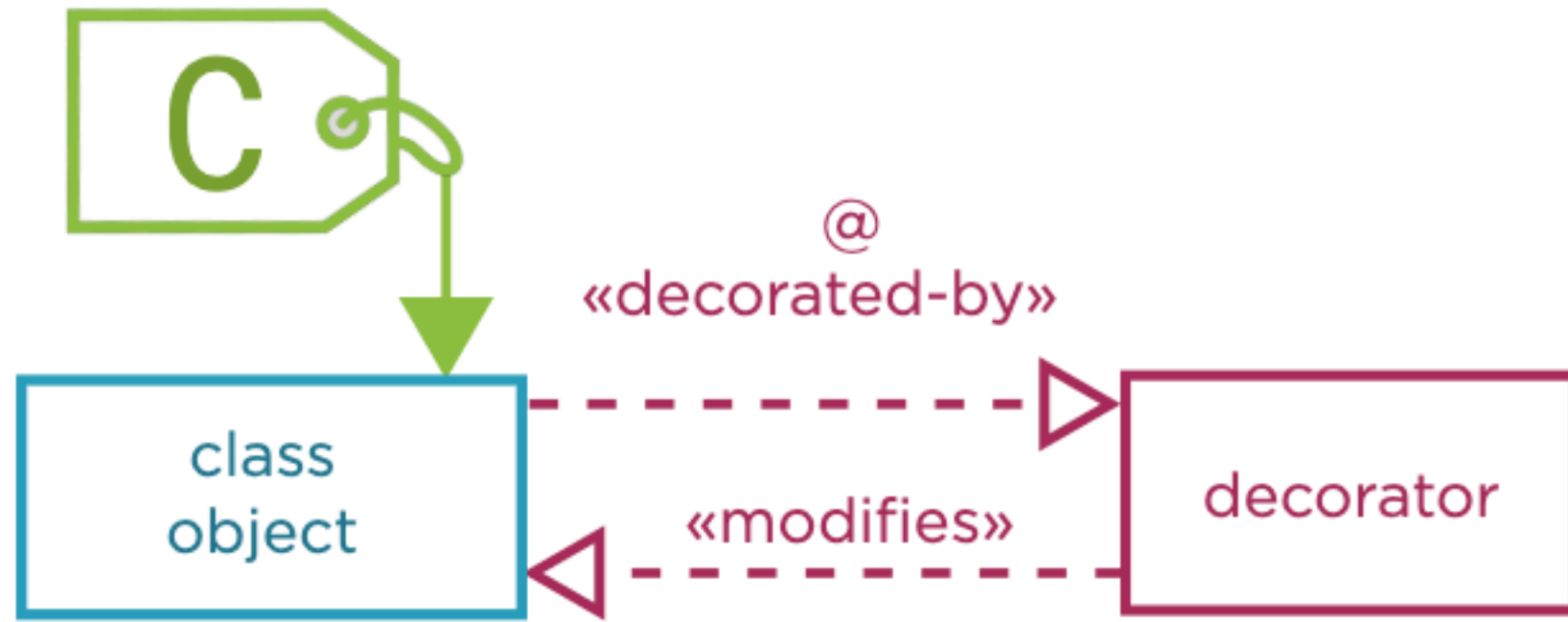
```
@decorator
def f():
    do_something()
```

# Class Decorator



```
@decorator
class C:
    # methods here
```

# Class Decorator Transforms in-place



```
@decorator  
class C:  
    # methods here
```



Can We Synthesize a Method?

---

```
location.py x
7 class Location:
8
9     def __init__(self, name, position):
10         self._name = name
11         self._position = position
12
13     @property
14     def name(self):
15         return self._name
16
17     @property
18     def position(self):
19         return self._position
20
21     def __str__(self):
22         return self.name
23
24
25 hong_kong = Location("Hong Kong", EarthPosition(22.29, 114.16))
26 stockholm = Location("Stockholm", EarthPosition(59.33, 18.06))
27 cape_town = Location("Cape Town", EarthPosition(-33.93, 18.42))
28 rotterdam = Location("Rotterdam", EarthPosition(51.92, 4.47))
```

```
def auto_repr(cls):
    members = vars(cls)

    if "__repr__" in members:
        raise TypeError(f"{cls.__name__} already defines __repr__")

    if "__init__" not in members:
        raise TypeError(f"{cls.__name__} does not override __init__")
```

auto\_repr()

```
import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['/var/folders/bb/w8tlddfn2fz1lhtlj_rxp_000000gn/T/tmpv9rin7kg/build/decorators'])
```

## Python Console

```
>>> from location import *
```

```
>>> hong_kong
```

```
Location(name='Hong Kong', position=EarthPosition(latitude=22.29, longitude=114.16))
```



# Class Decorator Factories

---

location.py x itinerary.py x

```

55 @auto_repr
56 @invariant(no_duplicates)
57 @invariant(at_least_two_locations)
58 class Itinerary:
59
60     @classmethod
61     def from_locations(cls, *locations):
62         return cls(locations)
63
64     def __init__(self, locations):
65         self._locations = list(locations)
66
67     def __str__(self):

```

Itinerary

Python Console x

```

File "/var/folders/bb/w8tlddfn2fz1lhtlj_rxp_000000gn/T/tmpto0b36ip/build/decorators/itinerary.py", line 17, in wrapper
    raise RuntimeError(
RuntimeError: Post-condition no_duplicates not maintained for Itinerary(locations=(Location(name='Rotterdam',
position=EarthPosition(latitude=51.96, longitude=4.47)), Location(name='Rotterdam', position=EarthPosition(latitude=51
.96, longitude=4.47))))
>>>

```



## Summary



Class decorators **transform** class definitions

Class decorators are unary functions which accept a **class object**, cls

Class decorators should return a class object, often the **same** one they accept

Class decorators are a **simpler** alternative to metaclasses

Class decorator **factories** facilitate parameterization

**Multiple** class decorators can be applied