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():

. . .

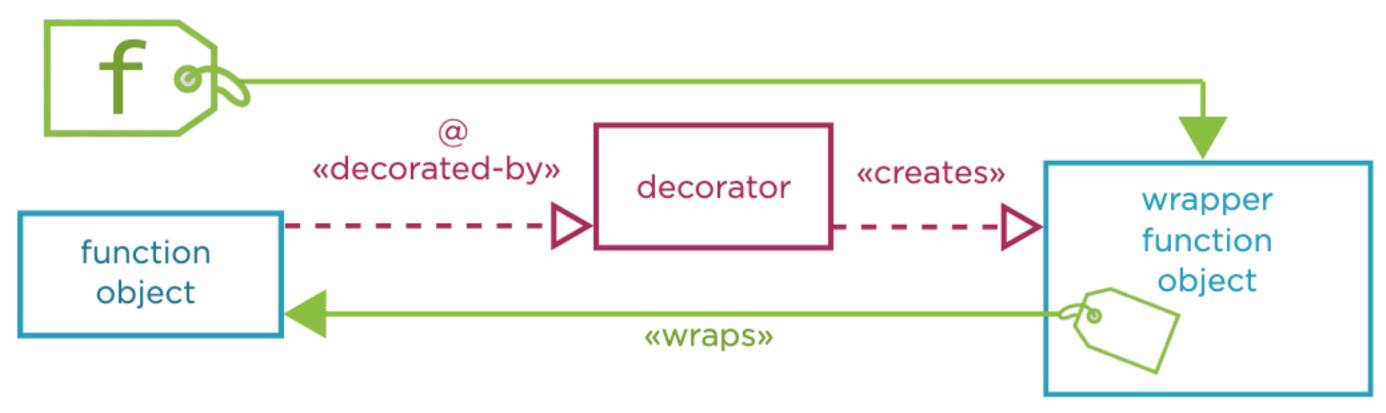
Core Python: Functions and Functional Programming



on

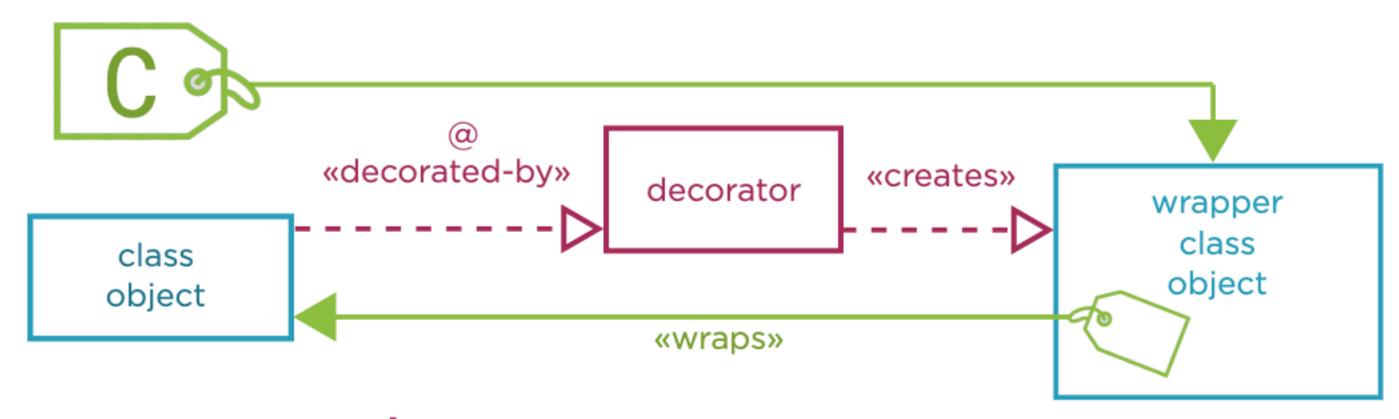
PLURALSIGHT

Decorator Recap



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

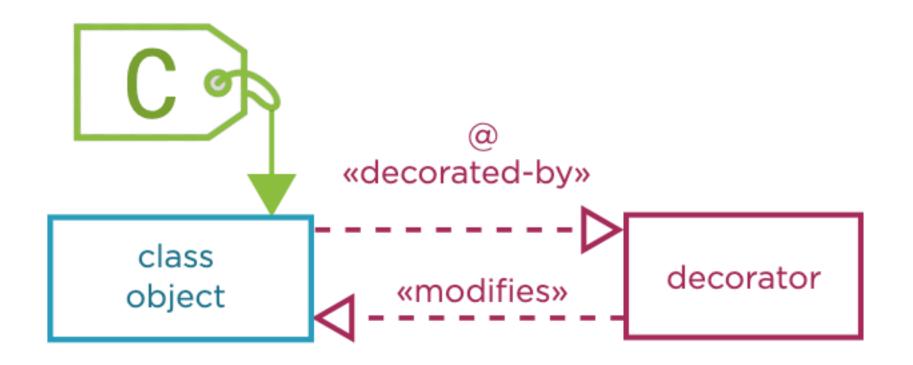
Class Decorator



@decorator class C:

methods here

Class Decorator Transforms in-place



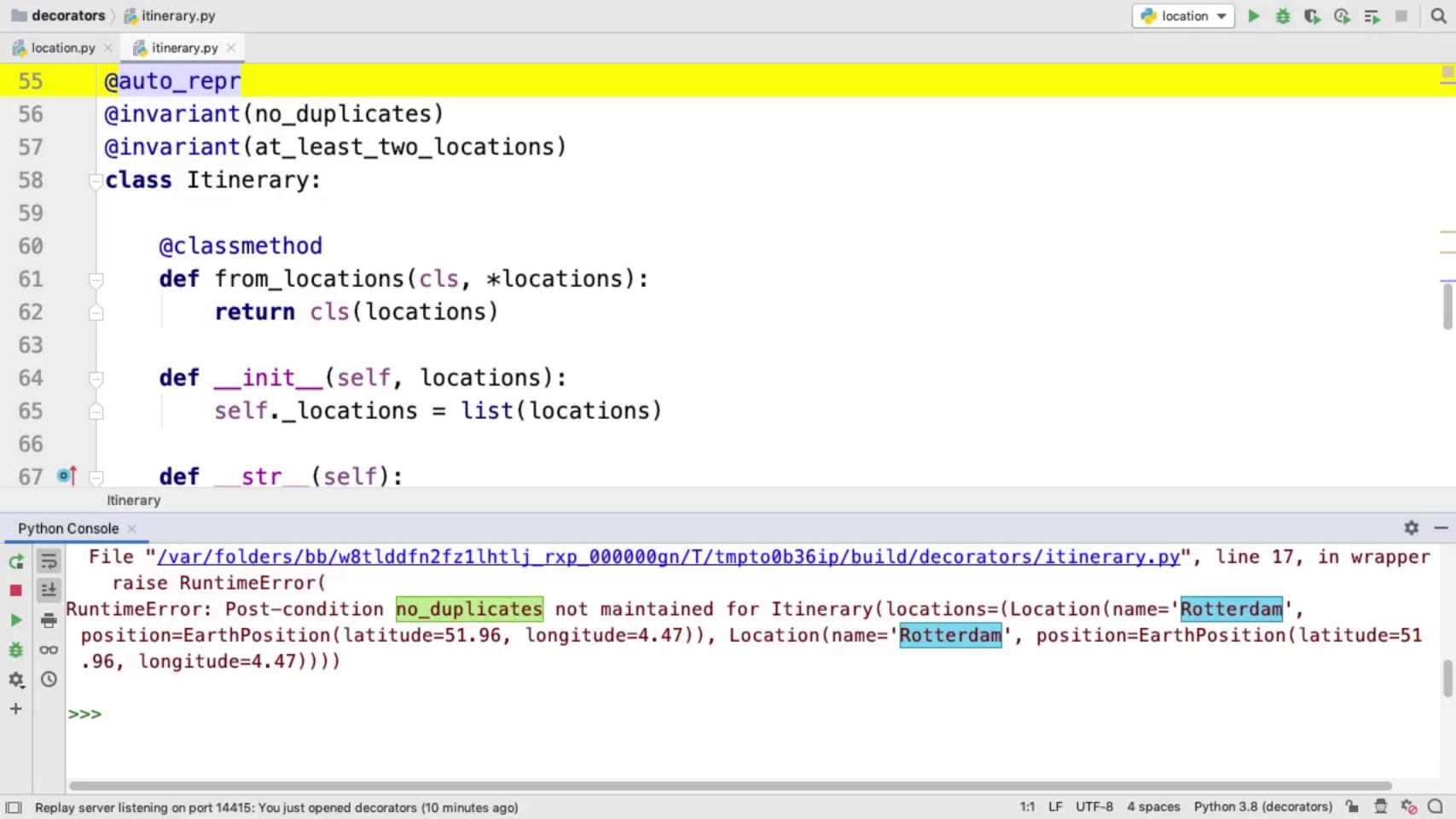
@decorator
class C:
 # methods here

Can We Synthesize a Method?

```
decorators \ 6 location.py
                                                                                                  Plocation ▼ ▶ # □ Q
location.py ×
       class Location:
            def __init__(self, name, position):
                 self._name = name
10
                 self._position = position
13
            @property
            def name(self):
14
                 return self._name
15
16
            @property
17
            def position(self):
18
                 return self._position
19
20
21 💇 🖢 🖁
            def __str__(self):
                 return self.name
22
23
24
       hong_kong = Location("Hong Kong", EarthPosition(22.29, 114.16))
25
       stockholm = Location("Stockholm", EarthPosition(59.33, 18.06))
26
27
       cape_town = Location("Cape Town", EarthPosition(-33.93, 18.42))
       Location
                                                                                       21:1 LF UTF-8 4 spaces Python 3.8 (decorators) 🚡 👼 🔾
Replay server listening on port 14415: You just opened decorators (3 minutes ago)
```

```
decorators \ location.py
                                                                                                     location.py X
       def auto_repr(cls):
            members = vars(cls)
            if "__repr__" in members:
10
                 raise TypeError(f"{cls.__name__} already defines __repr__")
            if "__init__" not in members:
13
                 raise TypeError(f"{cls.__name__} does not override __init__")
14
15
       auto_repr()
Python Console >
      Support/JetBrains/Toolbox/apps/PyCharm-P/ch-0/192.6817.19/PyCharm.app/Contents/helpers/pydev/pydevconsole.py"
      --mode=client --port=54250
     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))
     >>>
Replay server listening on port 14415: You just opened decorators (10 minutes ago)
                                                                                          1:1 LF UTF-8 4 spaces Python 3.8 (decorators) 🚡 👼
```





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