## Metaclass and Class Creation



Robert Smallshire
COFOUNDER - SIXTY NORTH
@robsmallshire



Austin Bingham
COFOUNDER - SIXTY NORTH
@austin\_bingham

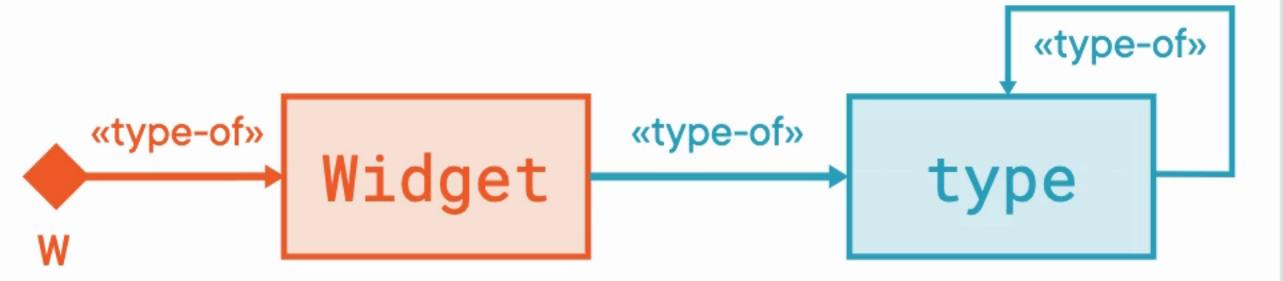
```
class Widget: pass
```

Default Base Class

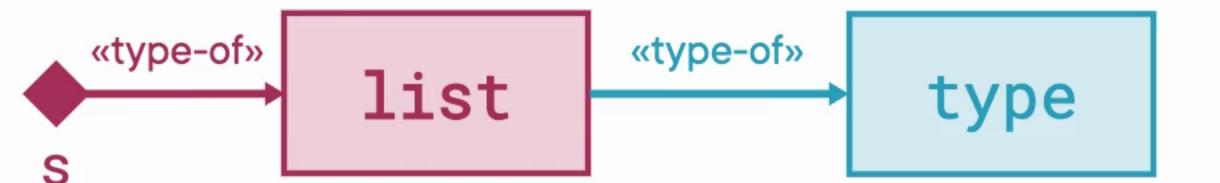
Class Widget(object, metaclass=type):

pass

# type is its own metaclass



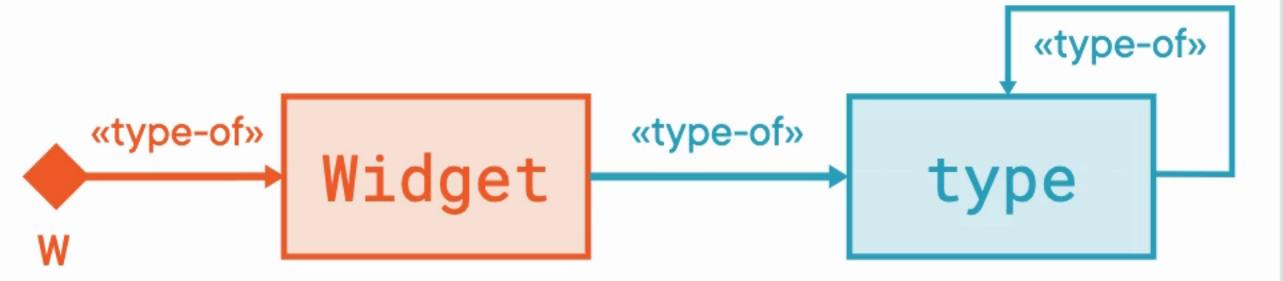
# type is the metaclass



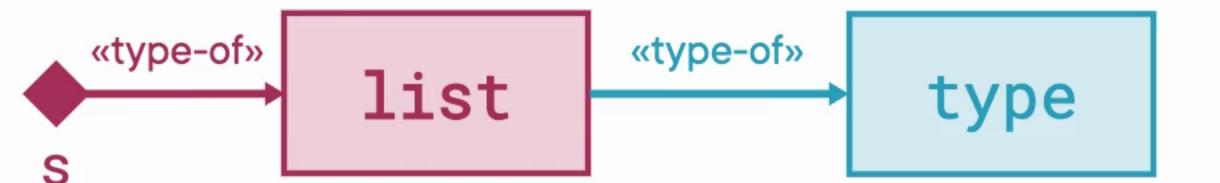
```
__class__
Widget type
```

```
>>> w.__class__
<class '__main__.Widget'>
>>> w.__class__._class__
<class 'type'>
>>> w.__class__._class__._class__
<class 'type'>
>>>
```

# type is its own metaclass



# type is the metaclass



```
__class__
Widget type
```

```
>>> w.__class__
<class '__main__.Widget'>
>>> w.__class__._class__
<class 'type'>
>>> w.__class__._class__._class__
<class 'type'>
>>>
```

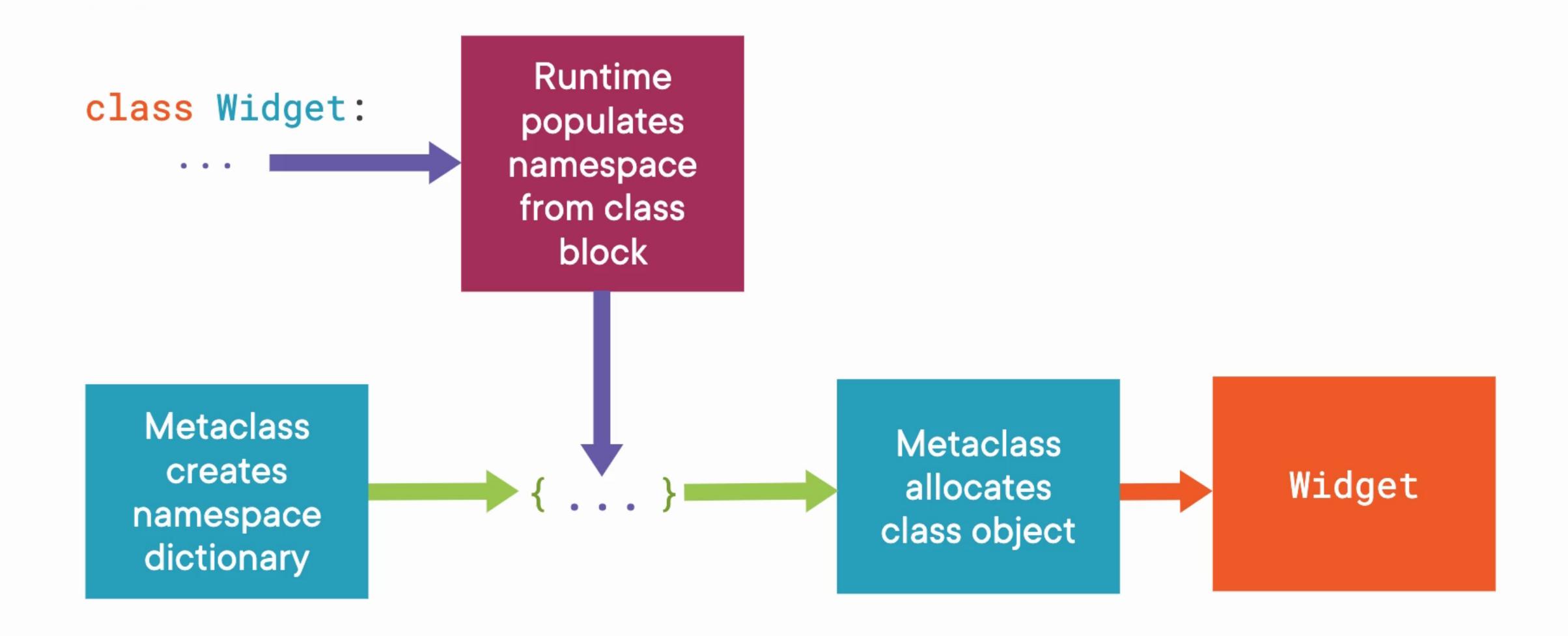
```
class Widget: pass
```

Default Base Class

Class Widget(object, metaclass=type):

pass





```
class Widget:
    ...
```

```
name = 'Widget'
metaclass = type
bases = ()
kwargs = {}
namespace = metaclass.__prepare__(name, bases, **kwargs)
Widget = metaclass.__new__(metaclass, name, bases, namespace, **kwargs)
metaclass.__init__(Widget, name, bases, namespace, **kwargs)
```

```
Add Configuration...
tracing-meta ) 📠 tracing.py
tracing.py
              print(T" (kwargs = }")
              cls = super().__new__(mcs, name, bases, namespace)
              print(f"-> {cls = }")
23
              print()
25
              return cls
          def __init__(cls, name, bases, namespace, **kwargs):
27
              print("TracingMeta.__init__(cls, name, bases, namespace)")
              print(f" {cls = }")
              print(f" {name = }")
30
              print(f" {bases = }")
31
              print(f" {namespace = }")
              print(f" {kwargs = }")
33
              super().__init__(name, bases, namespace)
34
              print()
35
                                                                                                                                                                                           Python Console
TracingMeta.__prepare__(name, bases, **kwargs)
       mcs = <class 'tracing.TracingMeta'>
        We could modify the sequence of base
                               classes
$ O
     Tracin meta.__new__(mes, name, bases, namespace)
      mcs___<class 'tracing.TracingMeta'>
       name = 'Widget'
       bases = ()
                                                                                                                                                     We could modify
       namespace = {'__module__': '__main__', '__qualname__': 'Widget', 'the_answer': 42, 'action': <function Widget.action at 0x101ca7880>}
                                                                                                                                                         namespace
       kwargs = {}
     -> cls = <class '__main__.Widget'>
     TracingMeta.__init_(cls, name, bases, namespace)
       cls = <class | Din Widget's
       name = 'Wi
       bases = ()
                       We could allocate a different class
                                                                              er': 42, 'action': <function Widget.action at 0x101ca7880>}
       namespace
                                          entirely
       kwargs = {
     >>>
                                                                                                                                                           36:1 LF UTF-8 4 spaces Python 3.10 (tracing-meta) 🦫 🔾
```

## Which Metaclass Methods to Override?

#### \_\_prepare\_\_

Customize the type or initial value of the namespace mapping

#### \_\_new\_\_

Allocate and optionally configure a new class object

#### \_\_init\_\_

Configure the class object