abs_properties_interfaces

February 16, 2024

[7]:
```
# Can you force a subclass to implement a property by combining two decorators?
# @abstractmethod and @property
```

[8]:
```
# Example:
from abc import ABC, abstractmethod

class Base(ABC):
    def __init__(self, number):
        self._number = number

    @property
    @abstractmethod
    def number(self):
        pass

class Implementation(Base):
    @property
    def number(self):
        return self._number

impl = Implementation(number=42)
print(impl.number)
```

```
42
```

[9]:
```
# Oops, applying the @property decorator is not enforced. Hence, forgetting it␣
 ↪won't raise an error,
# but number is now a regular method and not a property.
```

[10]:
```
class Base(ABC):
    def __init__(self, number):
        self._number = number

    # Please define it as a property!
    @abstractmethod
    def number(self):
        pass
```

```
class Implementation(Base):
    @property
    def number(self):
        return self._number

impl = Implementation(number=42)
print(impl.number)

# We can rely on other means like code reviews to enforce it. However, I am not␣
 ↪aware of how to enforce it
# through the language.
```

```
42
```

[11]:
```
# Interfaces: requiring methods and class attributes to be defined, set.
# Just for reference:

from abc import ABCMeta, abstractmethod


class Shape(metaclass=ABCMeta):
    @classmethod
    def __subclasshook__(cls, subclass):
        return (hasattr(subclass, "sides") and
                hasattr(subclass, "area") and
                callable(subclass.area))

    @abstractmethod
    def area(self):
        raise NotImplementedError


class Square(Shape):
    sides = 4

    def __init__(self, length: int) -> None:
        self.__length = length

    def area(self):
        return self._Square__length * self._Square__length


square = Square(length=5)
print(issubclass(Square, Shape))
print(isinstance(square, Shape))
```

```
True
True
```

[12]:
```
# By using ABCMeta as metaclass of our Shape interface, we can ensure that␣
 ↪subclasses of Shape set a class
# attribute called sides. If you don't you will be able to instantiate it, but␣
 ↪it won't be recognized as a
# subclass of our interface which is nice for testing. Making sure our␣
 ↪implementations adhere to what the
# interface dictates.
```

[ ]:
```

```