Classes and OO

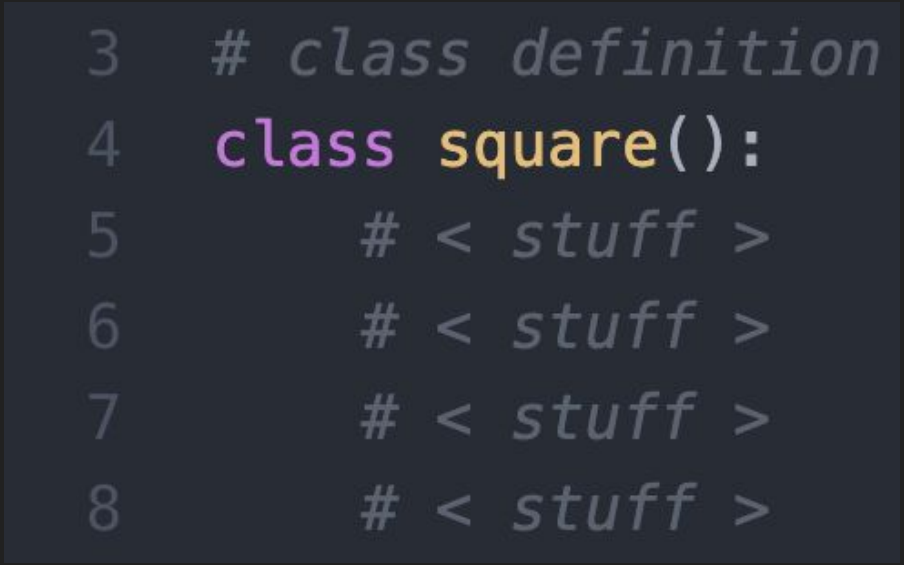
Chelsea Parlett Pelleriti

Classes

A template to make an object

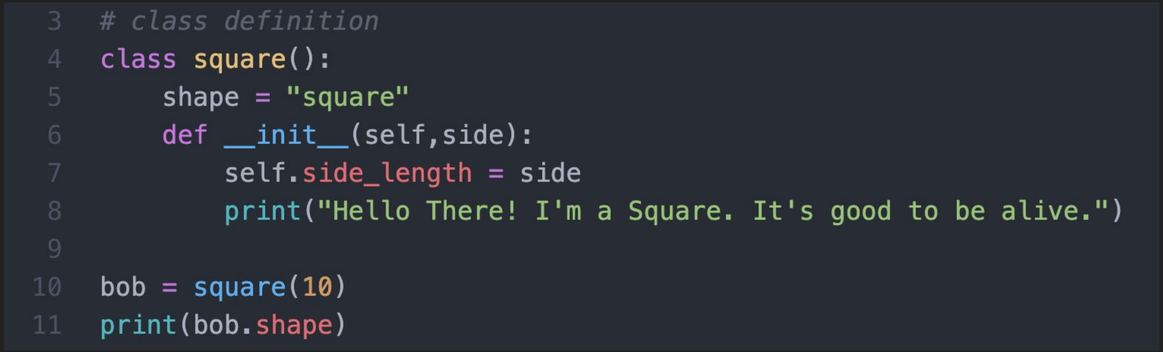
- Attributes
- Methods

Class Definition



```
# class definition
class square():
    # < stuff >
    # < stuff >
    # < stuff >
    # < stuff >
```

__init__



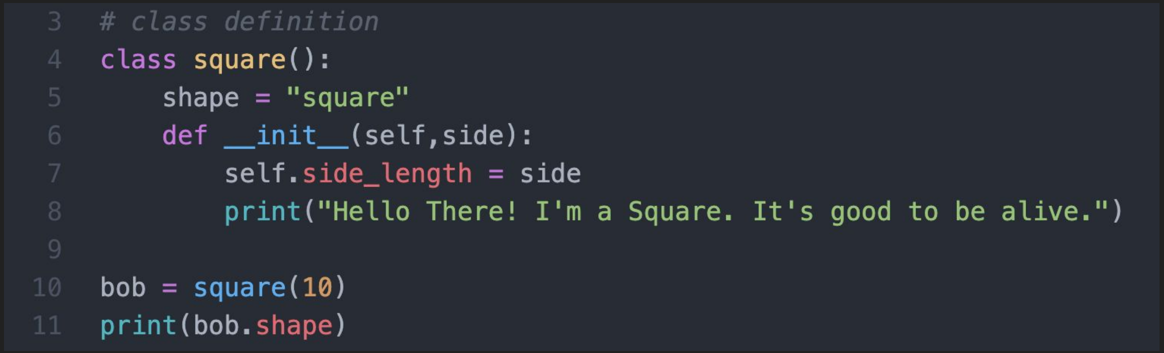
```python
# class definition
class square():
    shape = "square"
    def __init__(self,side):
        self.side_length = side
        print("Hello There! I'm a Square. It's good to be alive.")

bob = square(10)
print(bob.shape)
```

Self

- A way to refer to a specific *instance* of a class.
- All class methods take in self as its first argument.

Class Attribute vs. Instance Attribute

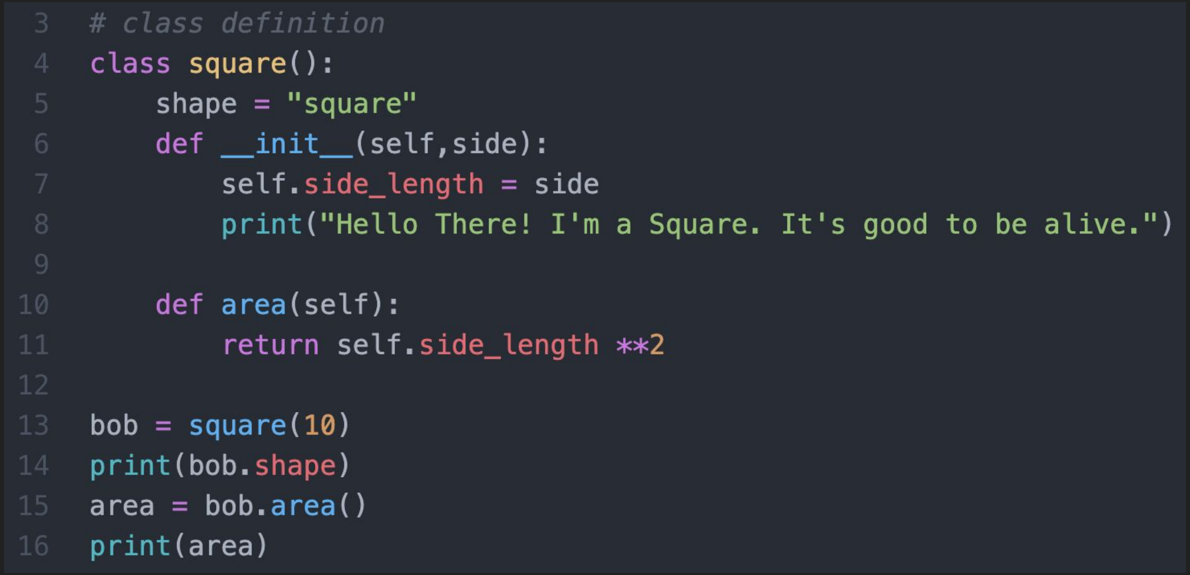


```python
# class definition
class square():
    shape = "square"
    def __init__(self,side):
        self.side_length = side
        print("Hello There! I'm a Square. It's good to be alive.")

bob = square(10)
print(bob.shape)
```

Instance Methods

- Instance methods (functions defined within the Class) have access to class attributes/instance attributes!!!!!!

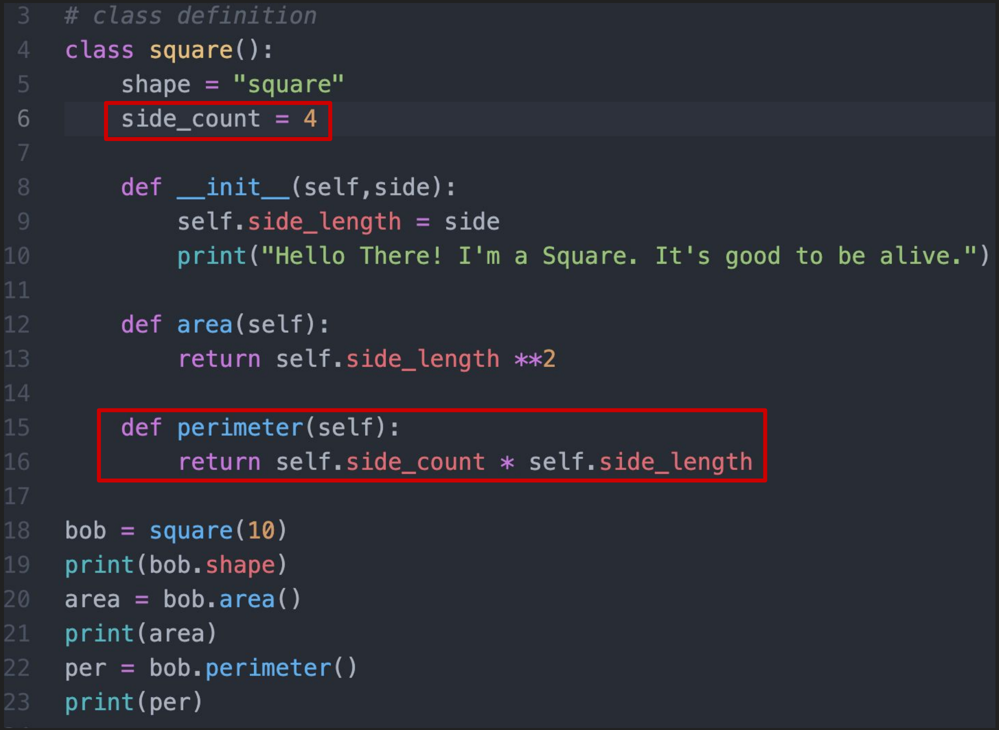


```python
# class definition
class square():
    shape = "square"
    def __init__(self,side):
        self.side_length = side
        print("Hello There! I'm a Square. It's good to be alive.")

    def area(self):
        return self.side_length **2

bob = square(10)
print(bob.shape)
area = bob.area()
print(area)
```

Instance Methods



```python
# class definition
class square():
    shape = "square"
    side_count = 4

    def __init__(self,side):
        self.side_length = side
        print("Hello There! I'm a Square. It's good to be alive.")

    def area(self):
        return self.side_length **2

    def perimeter(self):
        return self.side_count * self.side_length

bob = square(10)
print(bob.shape)
area = bob.area()
print(area)
per = bob.perimeter()
print(per)
```