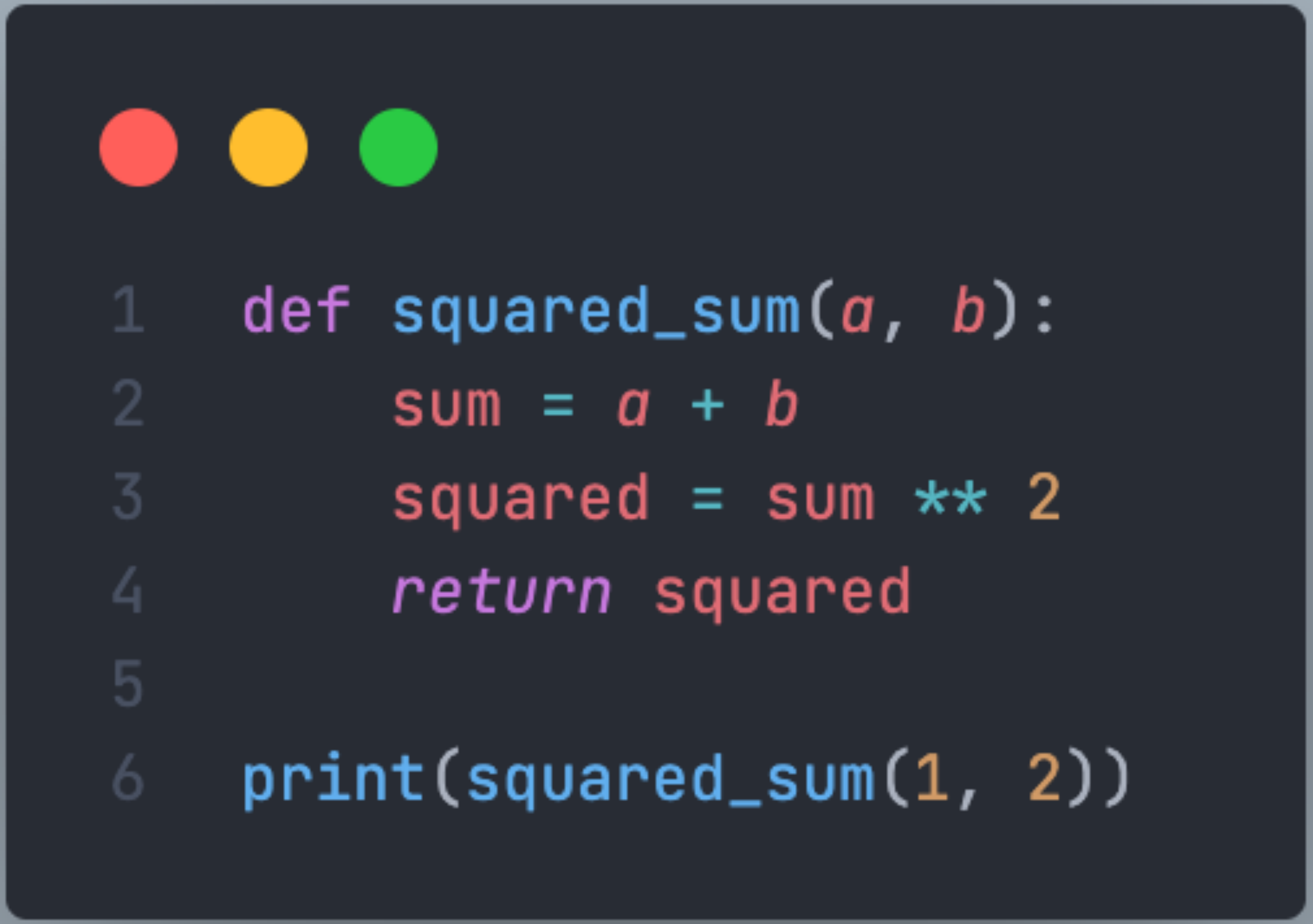
FP vs OOP

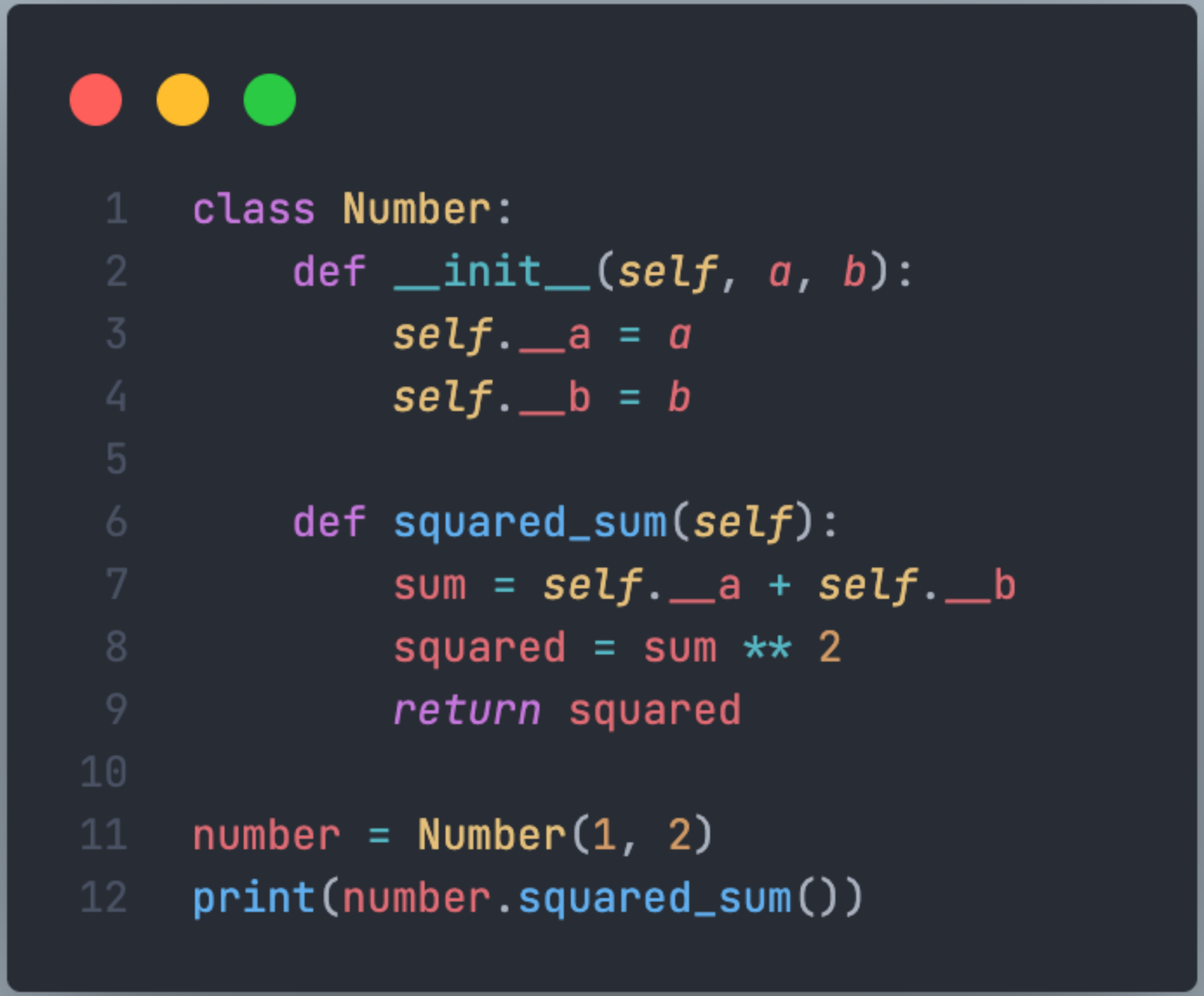
Part 2

Phan Anh - 02/07/2022

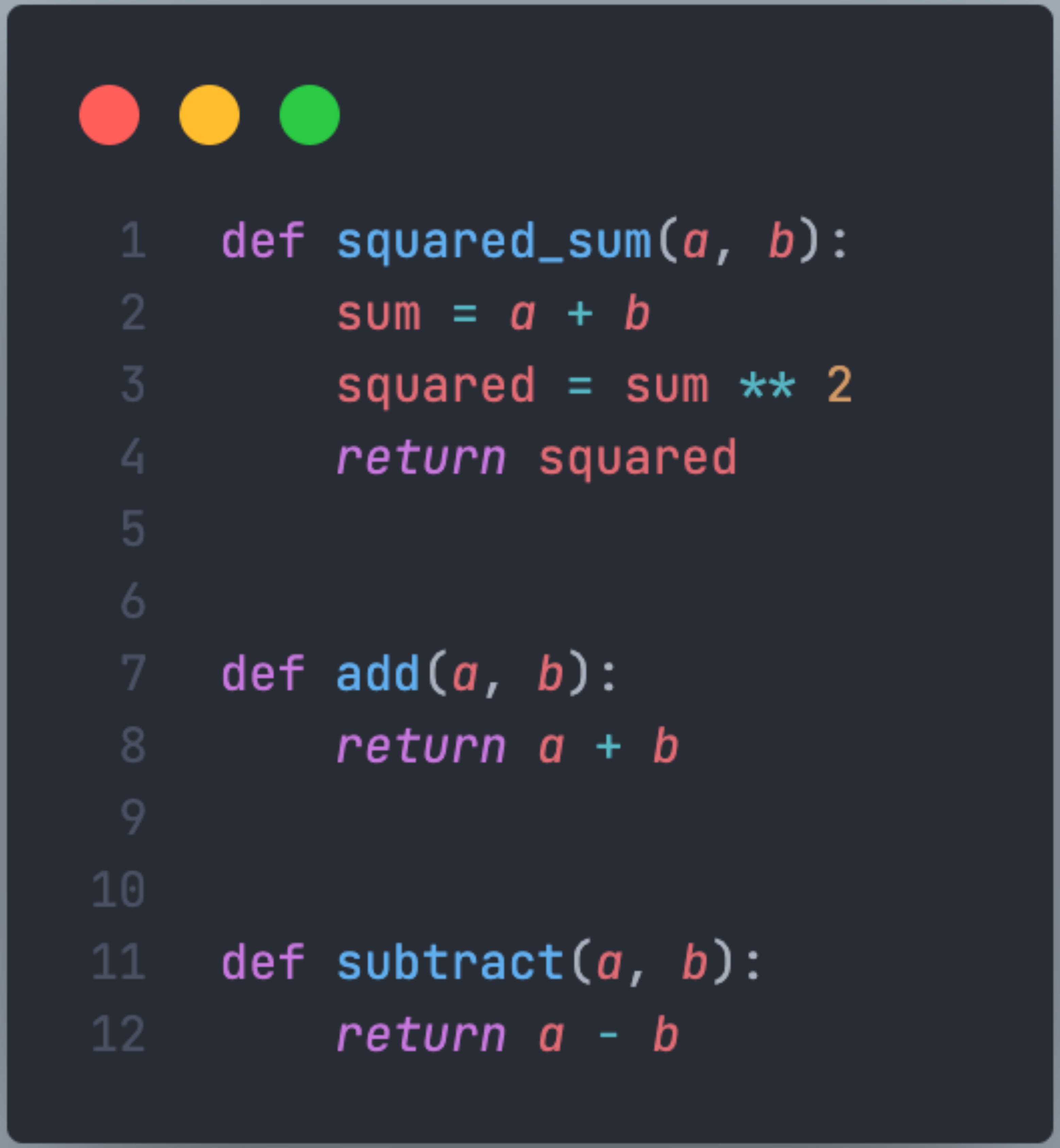


```python
def squared_sum(a, b):
    sum = a + b
    squared = sum ** 2
    return squared

print(squared_sum(1, 2))
```

Less code

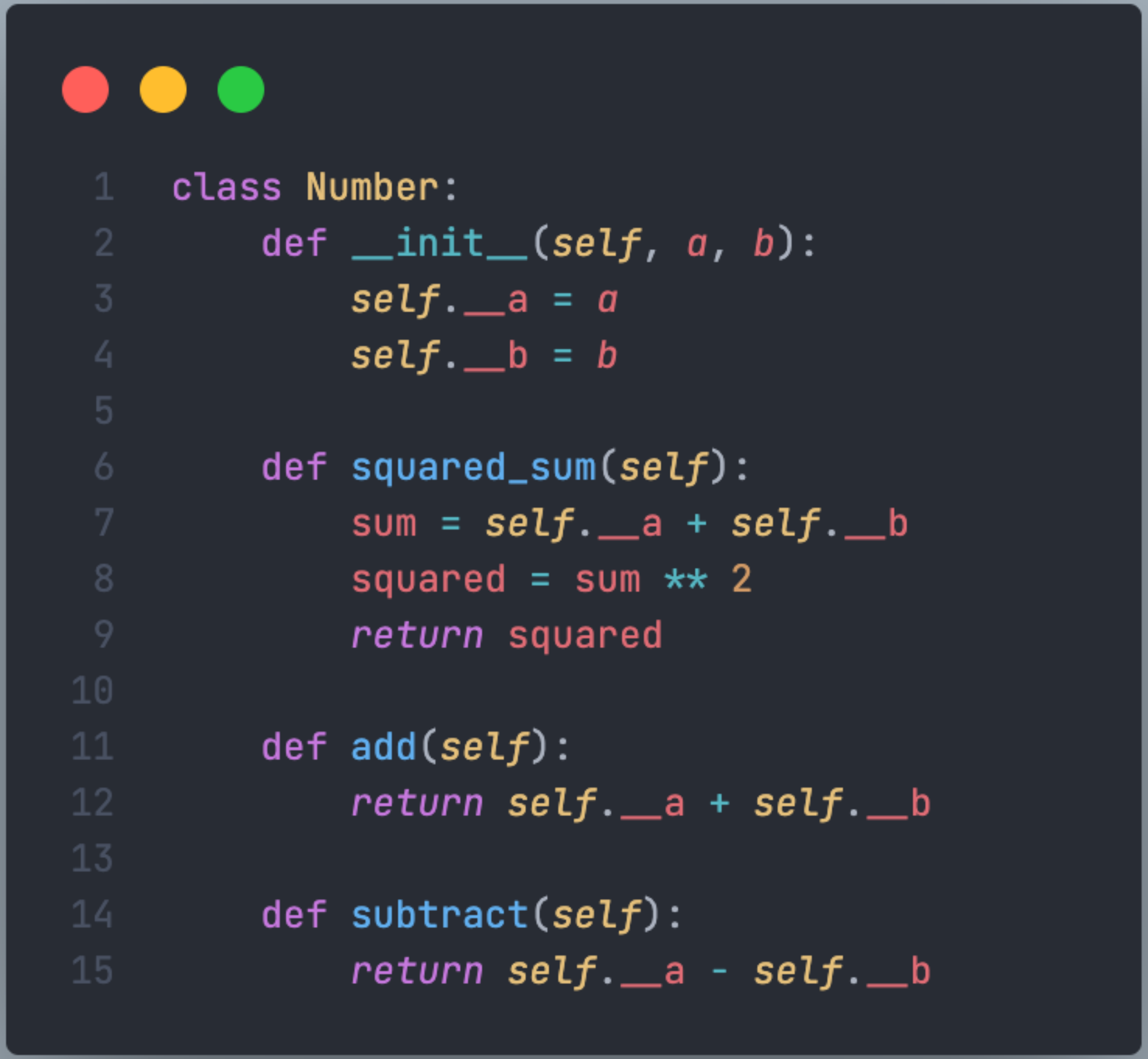


```python
class Number:
    def __init__(self, a, b):
        self.__a = a
        self.__b = b

    def squared_sum(self):
        sum = self.__a + self.__b
        squared = sum ** 2
        return squared

number = Number(1, 2)
print(number.squared_sum())
```

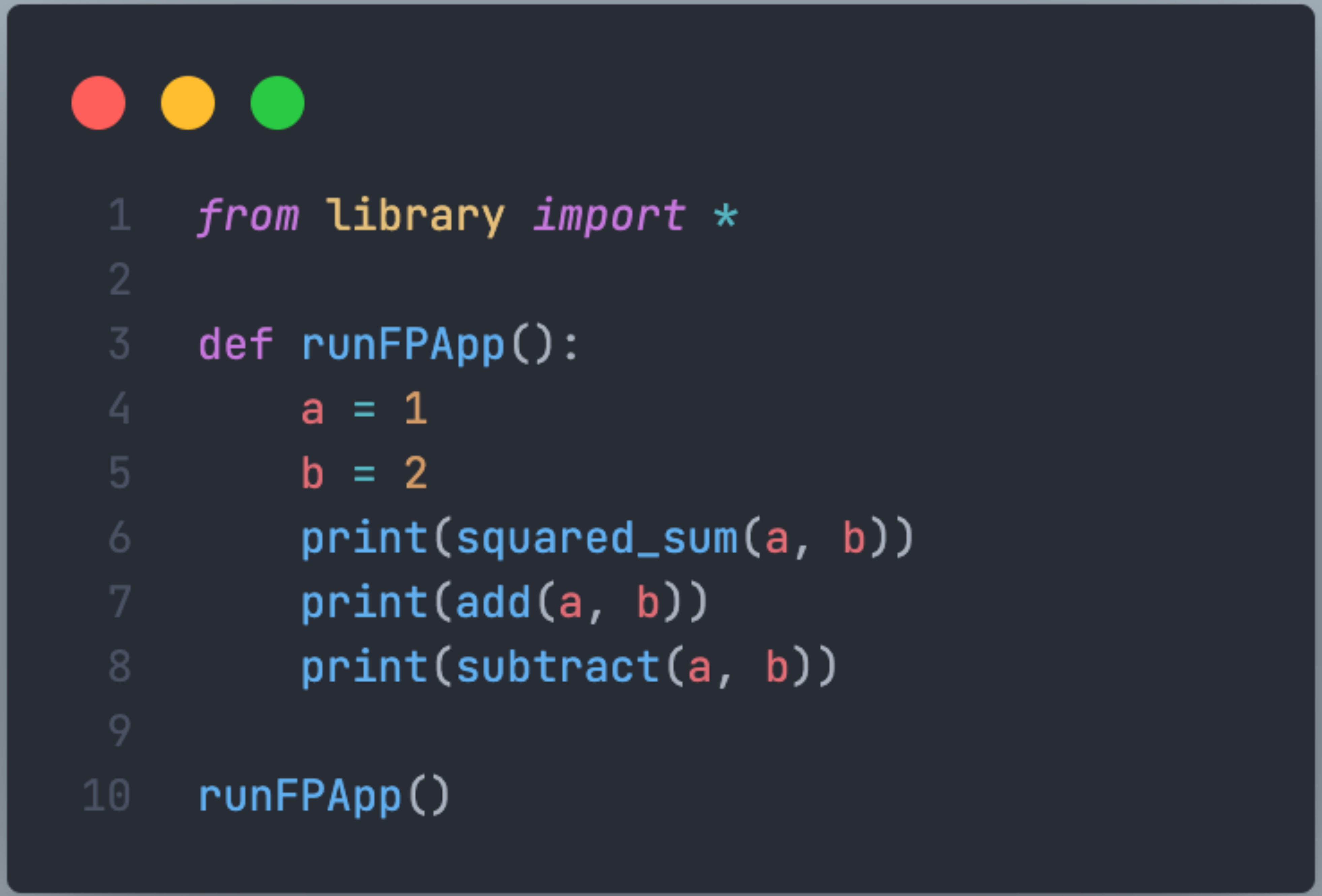


```python
def squared_sum(a, b):
    sum = a + b
    squared = sum ** 2
    return squared


def add(a, b):
    return a + b


def subtract(a, b):
    return a - b
```

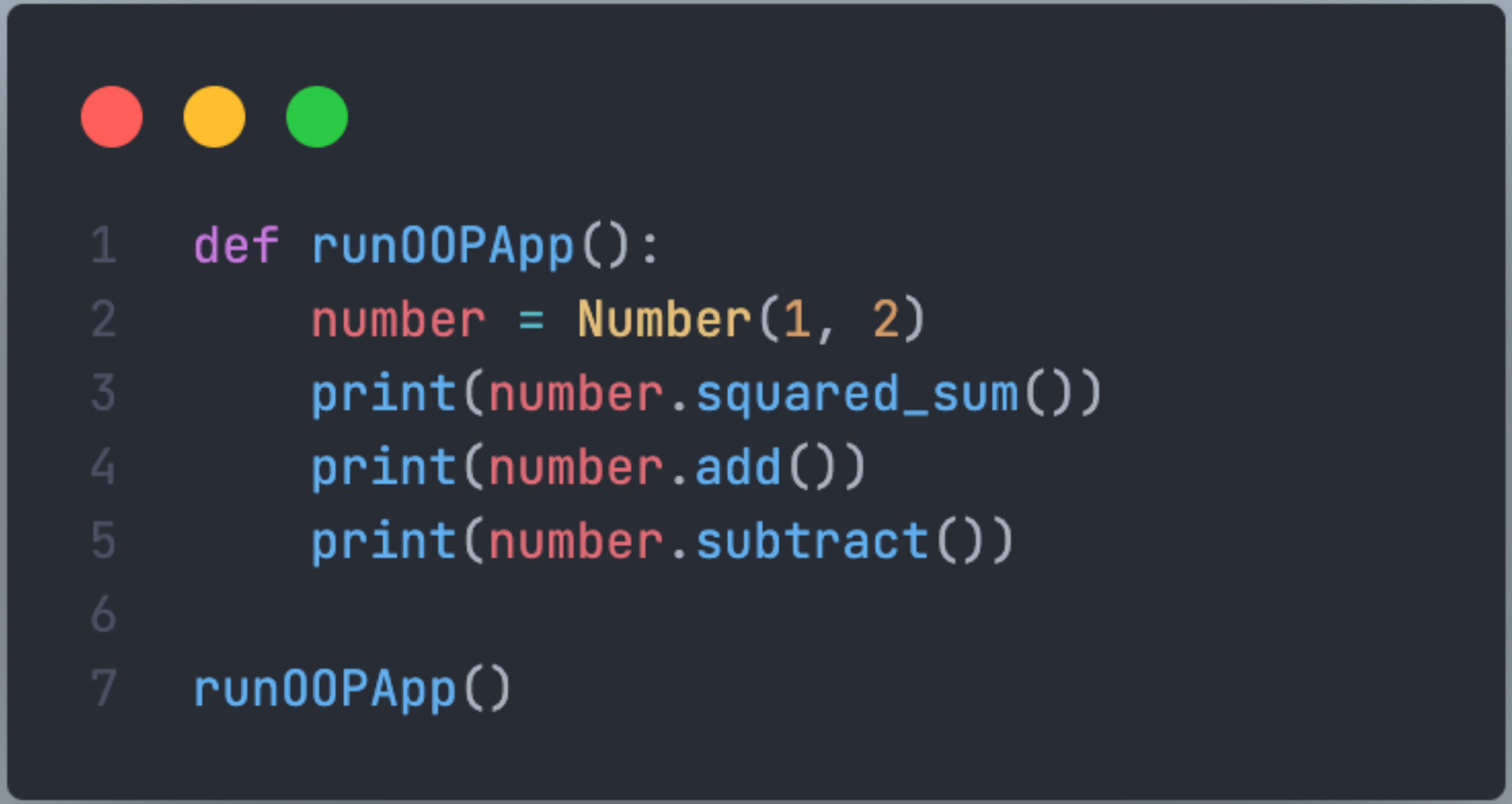
Less code



```python
class Number:
    def __init__(self, a, b):
        self.__a = a
        self.__b = b

    def squared_sum(self):
        sum = self.__a + self.__b
        squared = sum ** 2
        return squared

    def add(self):
        return self.__a + self.__b

    def subtract(self):
        return self.__a - self.__b
```



```python
from library import *

def runFPApp():
    a = 1
    b = 2
    print(squared_sum(a, b))
    print(add(a, b))
    print(subtract(a, b))

runFPApp()
```



```python
def runOOPApp():
    number = Number(1, 2)
    print(number.squared_sum())
    print(number.add())
    print(number.subtract())

runOOPApp()
```

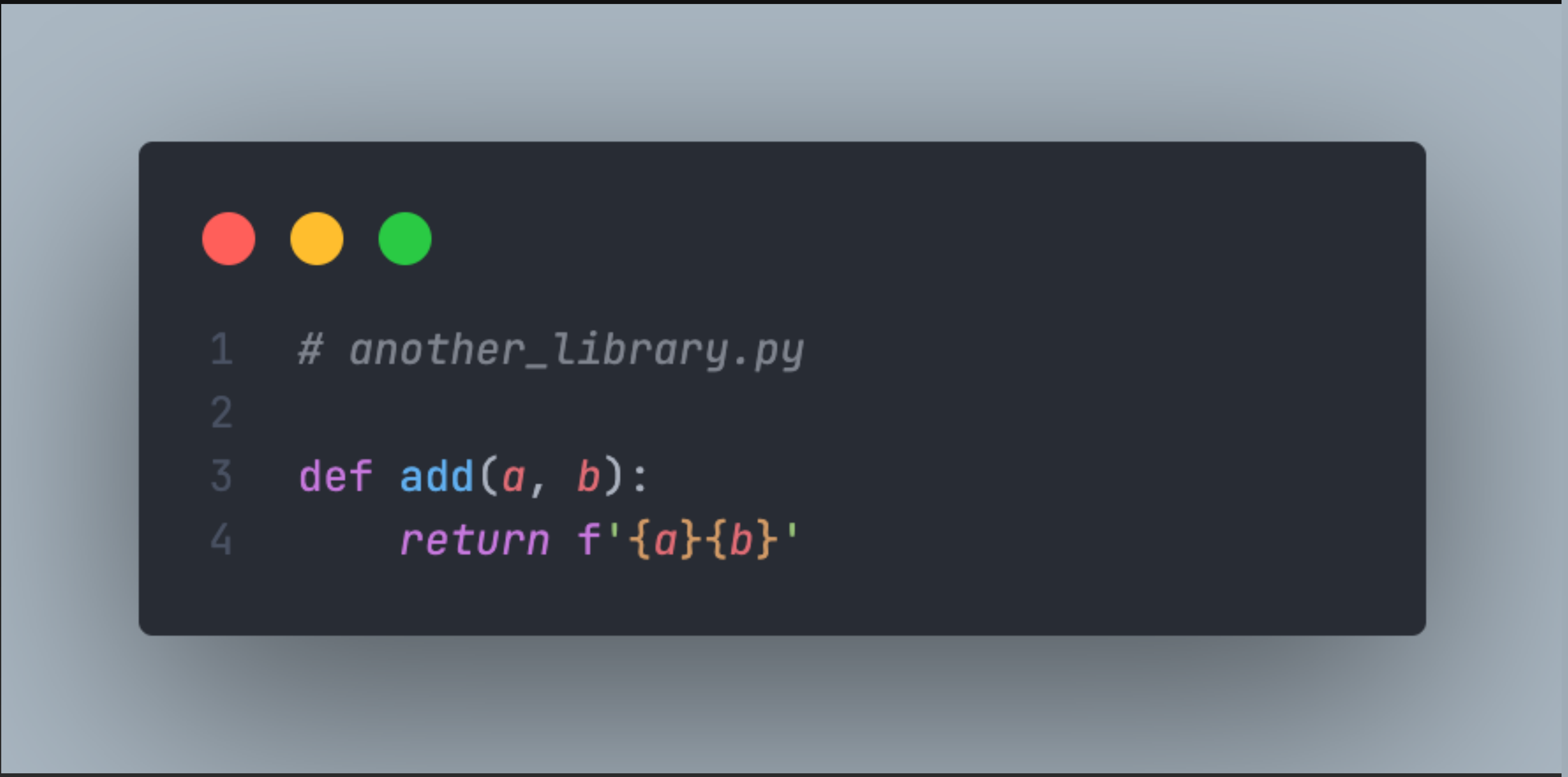
Easier to read

```
9
3
-1
```

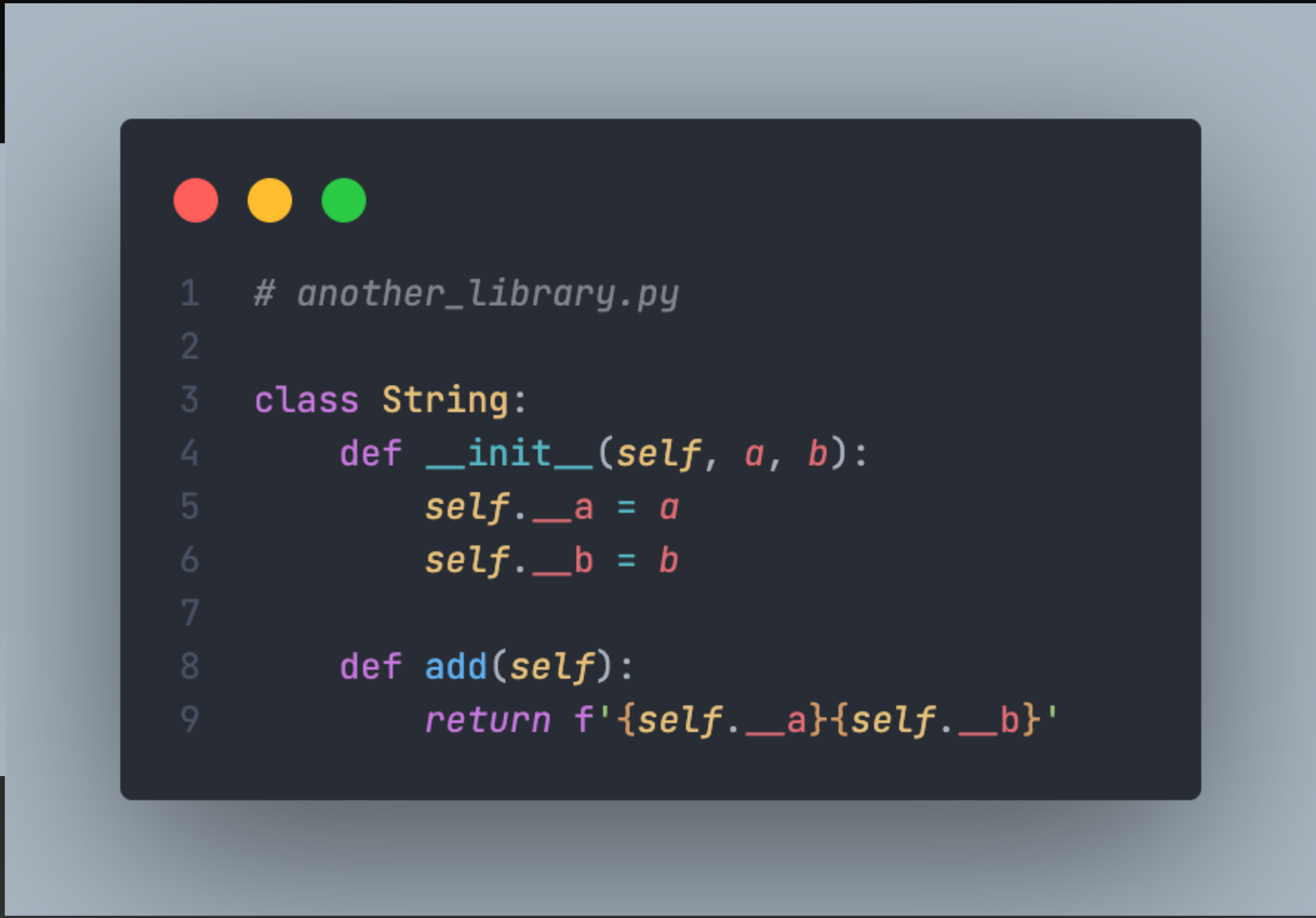
```
9
3
-1
```

runApp.ipynb

Add another library

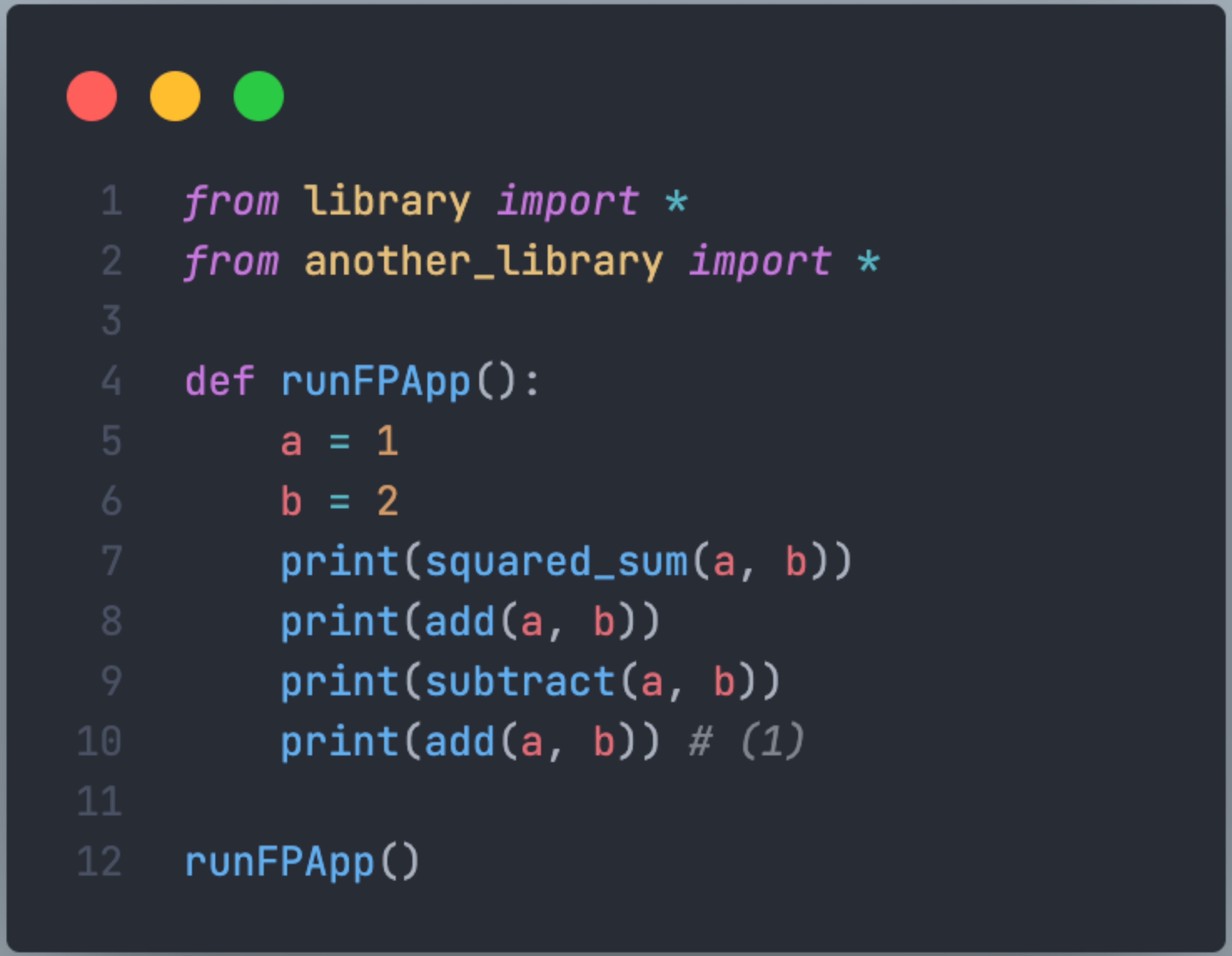


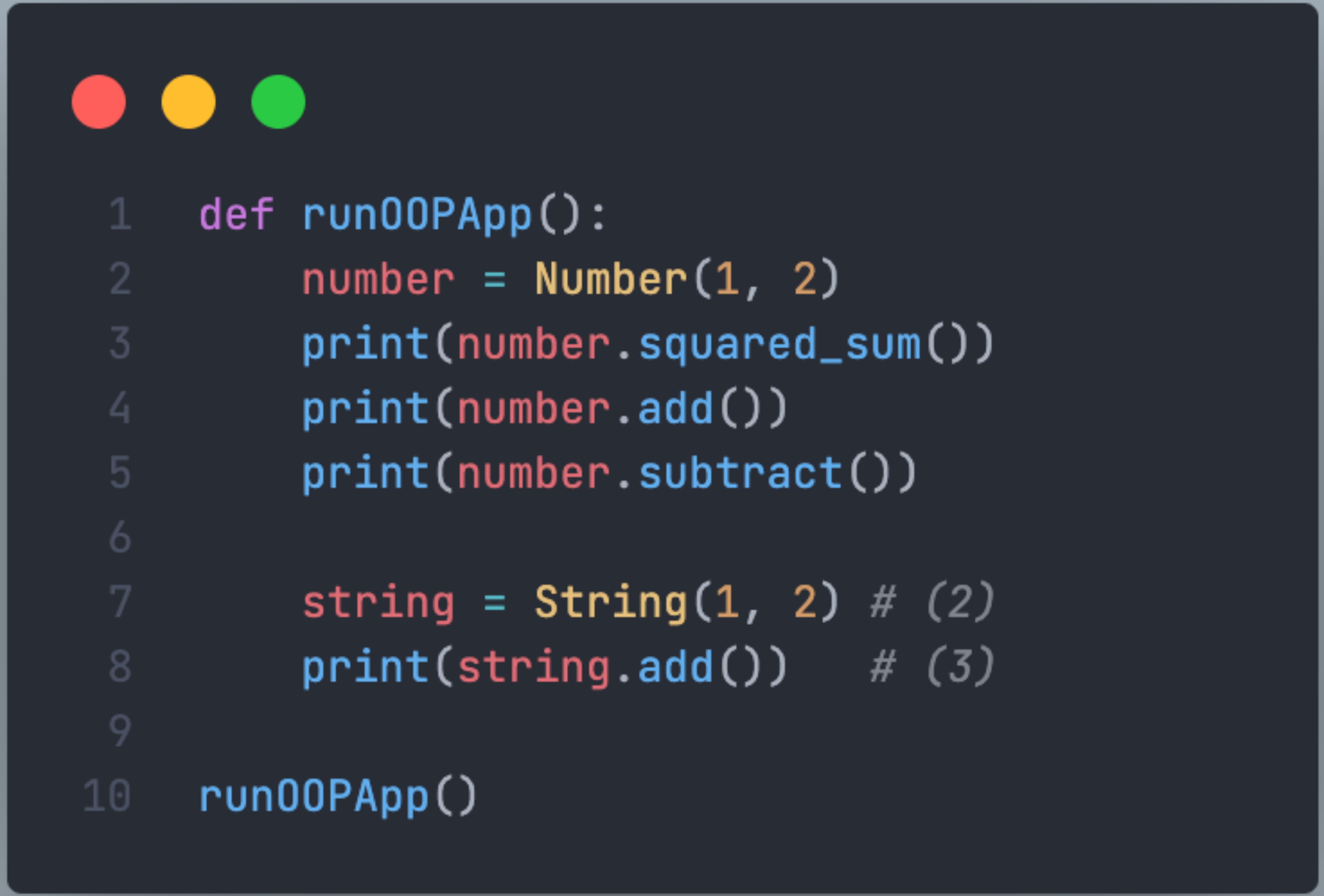
```python
# another_library.py

def add(a, b):
    return f'{a}{b}'
```



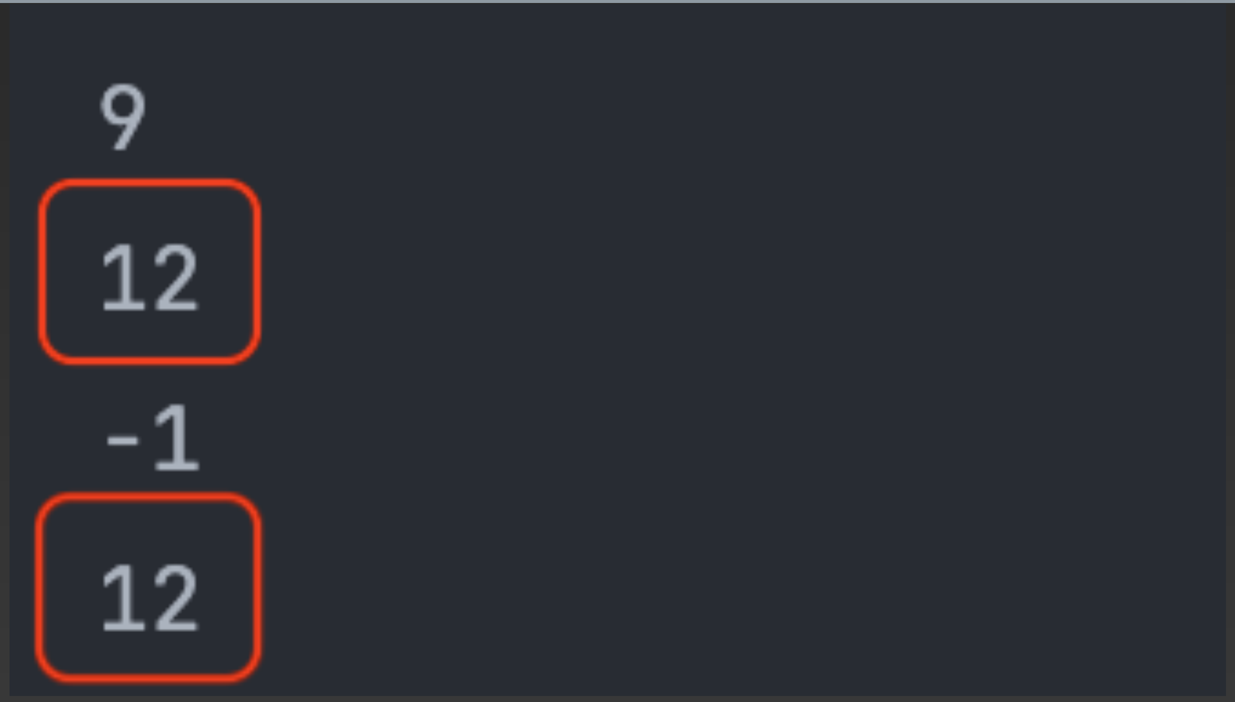
```python
# another_library.py

class String:
    def __init__(self, a, b):
        self.__a = a
        self.__b = b

    def add(self):
        return f'{self.__a}{self.__b}'
```

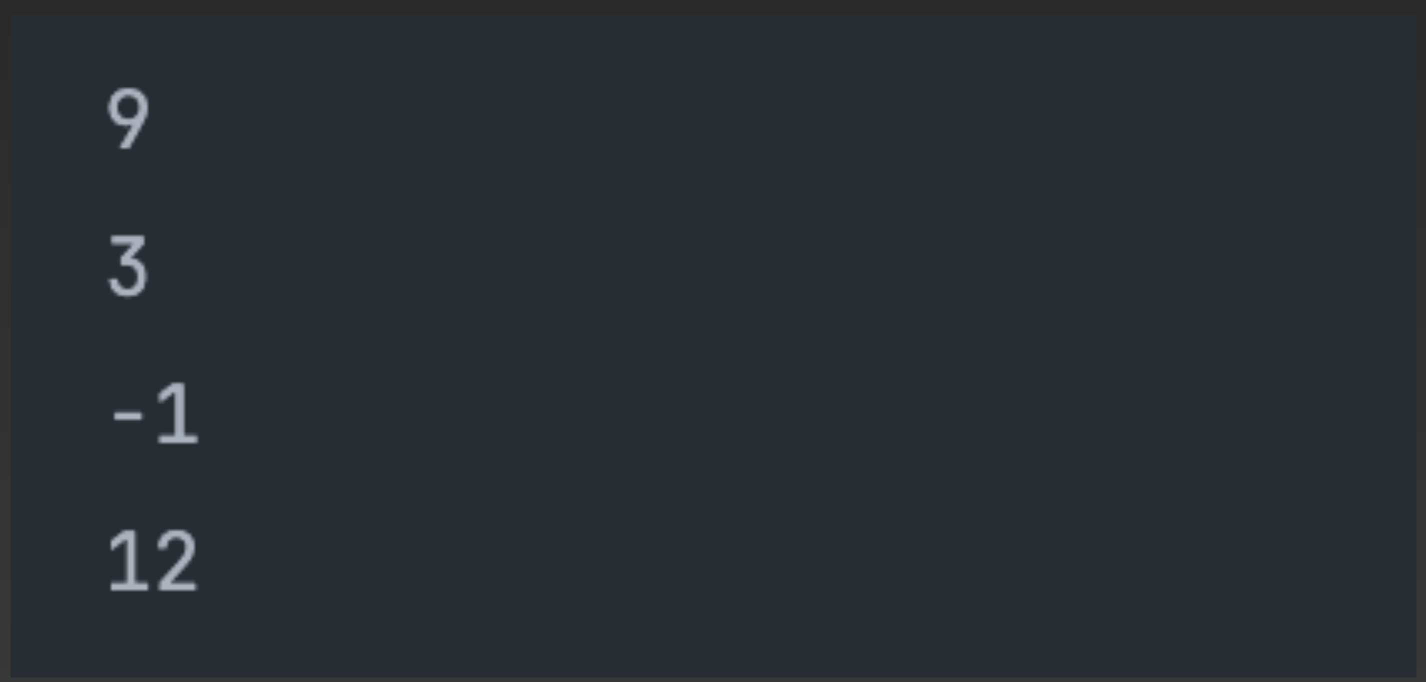
another_library.py


```python
from library import *
from another_library import *

def runFPApp():
    a = 1
    b = 2
    print(squared_sum(a, b))
    print(add(a, b))
    print(subtract(a, b))
    print(add(a, b)) # (1)

runFPApp()
```


```python
def runOOPApp():
    number = Number(1, 2)
    print(number.squared_sum())
    print(number.add())
    print(number.subtract())

    string = String(1, 2) # (2)
    print(string.add())   # (3)

runOOPApp()
```

Affected

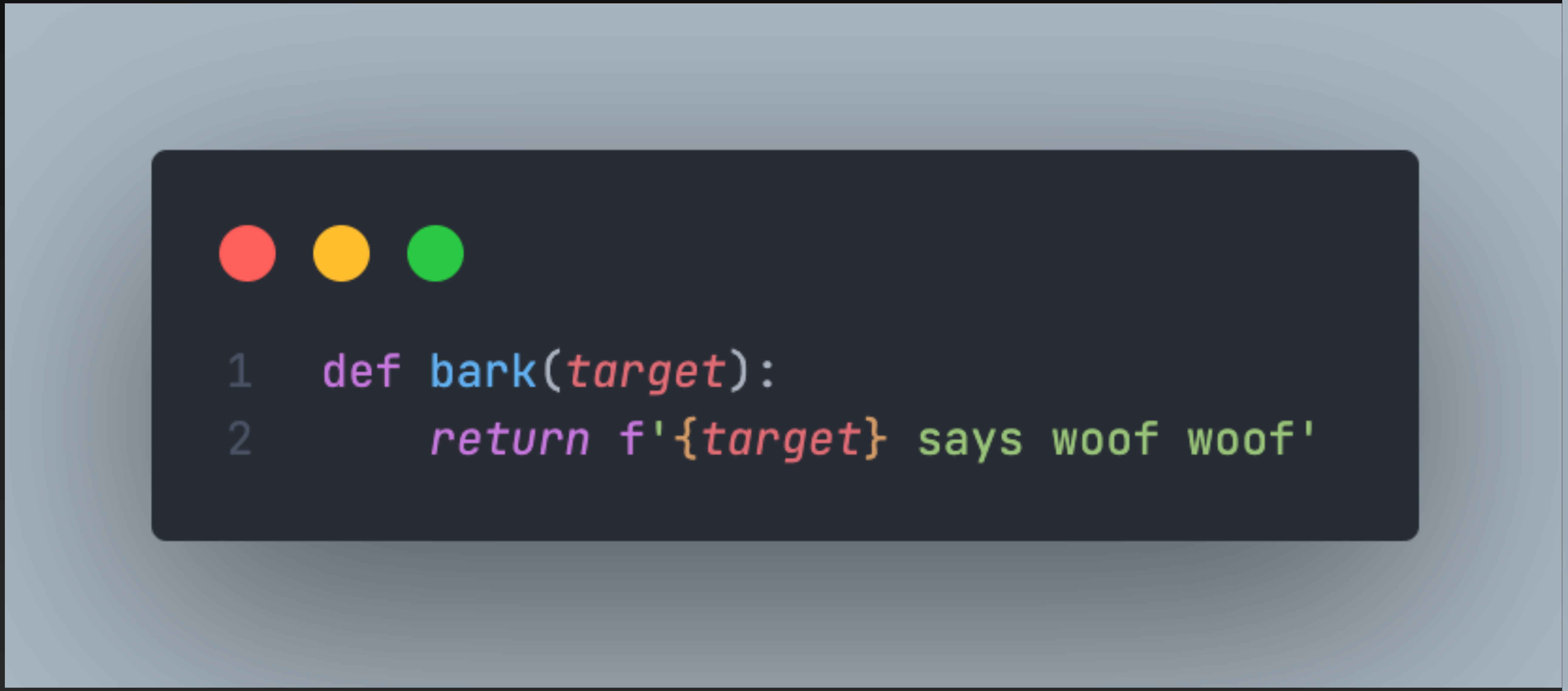

```
9
12
-1
12
```


```
9
3
-1
12
```

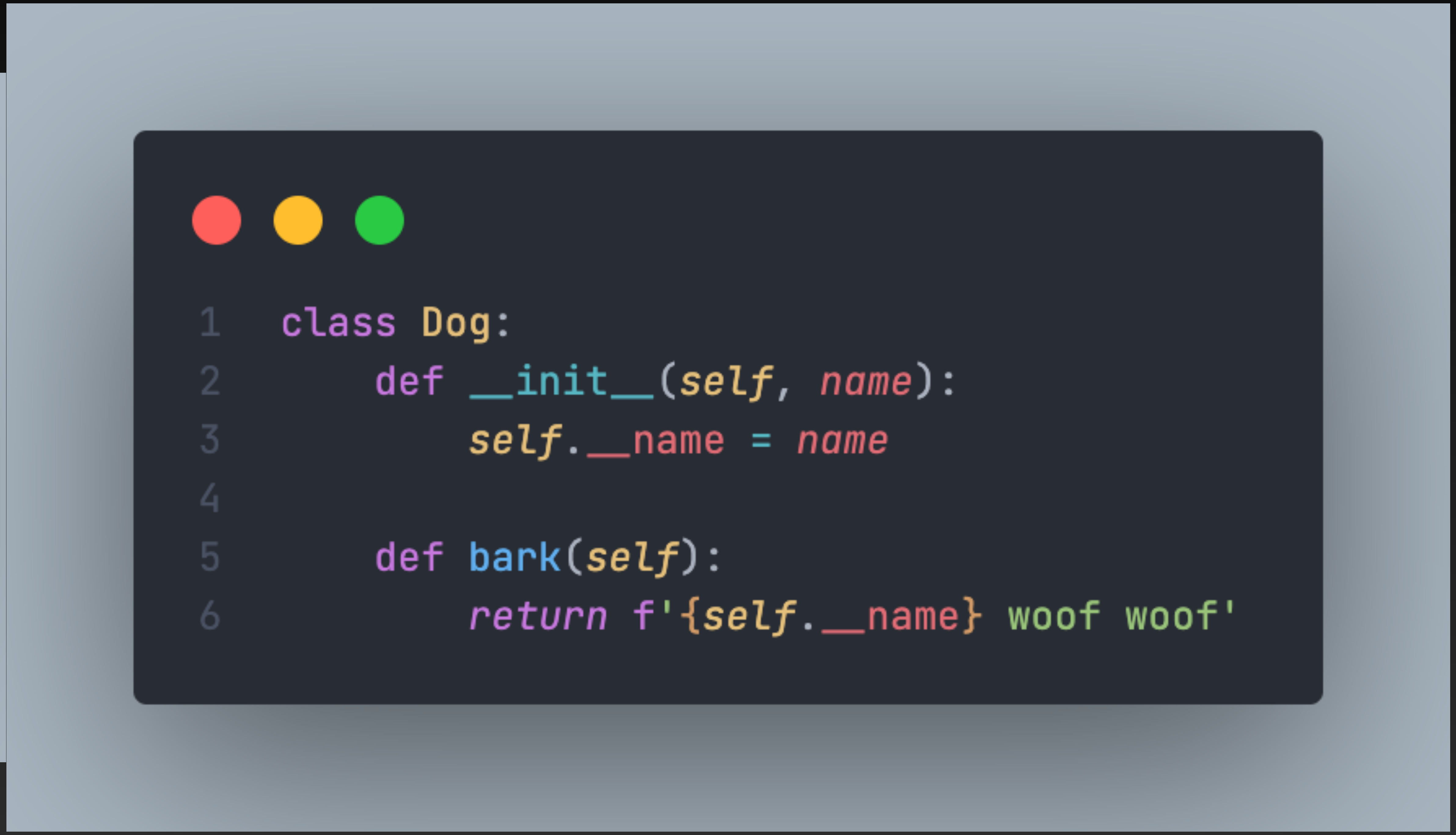
runApp.ipynb

Function vs Method

What & Why

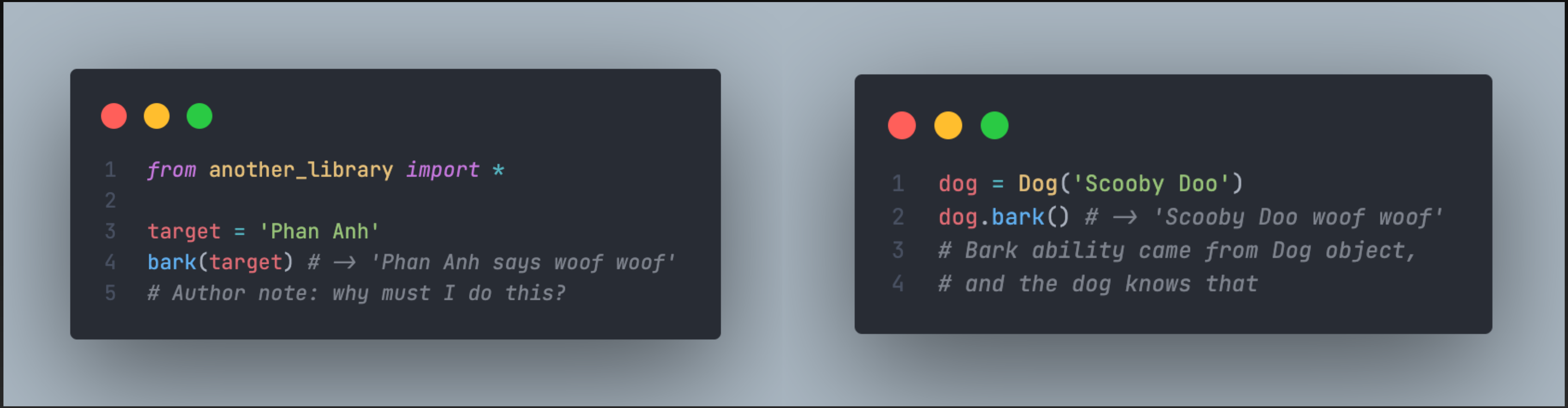


```python
def bark(target):
    return f'{target} says woof woof'
```



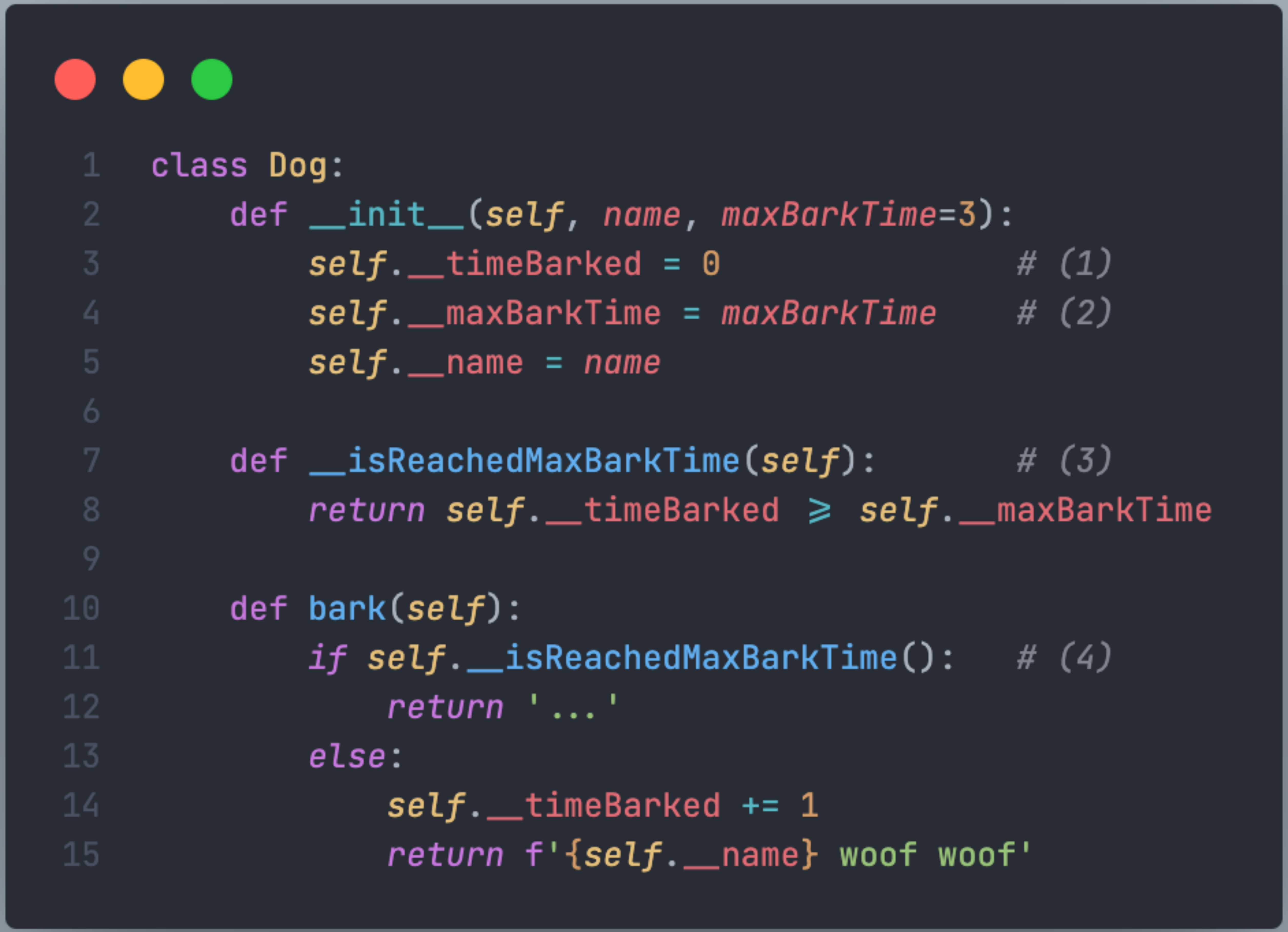
```python
class Dog:
    def __init__(self, name):
        self.__name = name

    def bark(self):
        return f'{self.__name} woof woof'
```

another_library.py



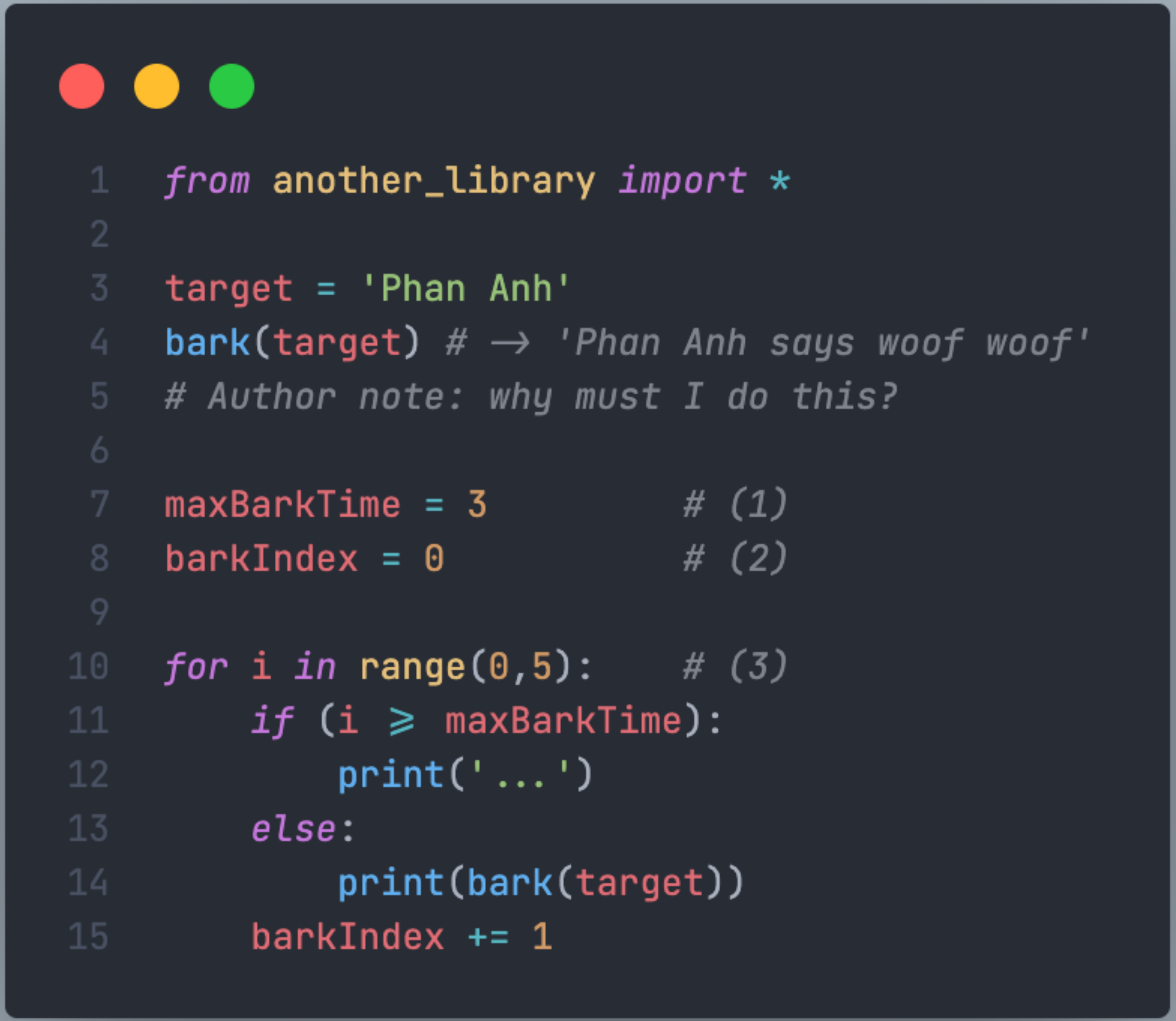
```python
from another_library import *

target = 'Phan Anh'
bark(target) # → 'Phan Anh says woof woof'
# Author note: why must I do this?
```

```python
dog = Dog('Scooby Doo')
dog.bark() # → 'Scooby Doo woof woof'
# Bark ability came from Dog object,
# and the dog knows that
```



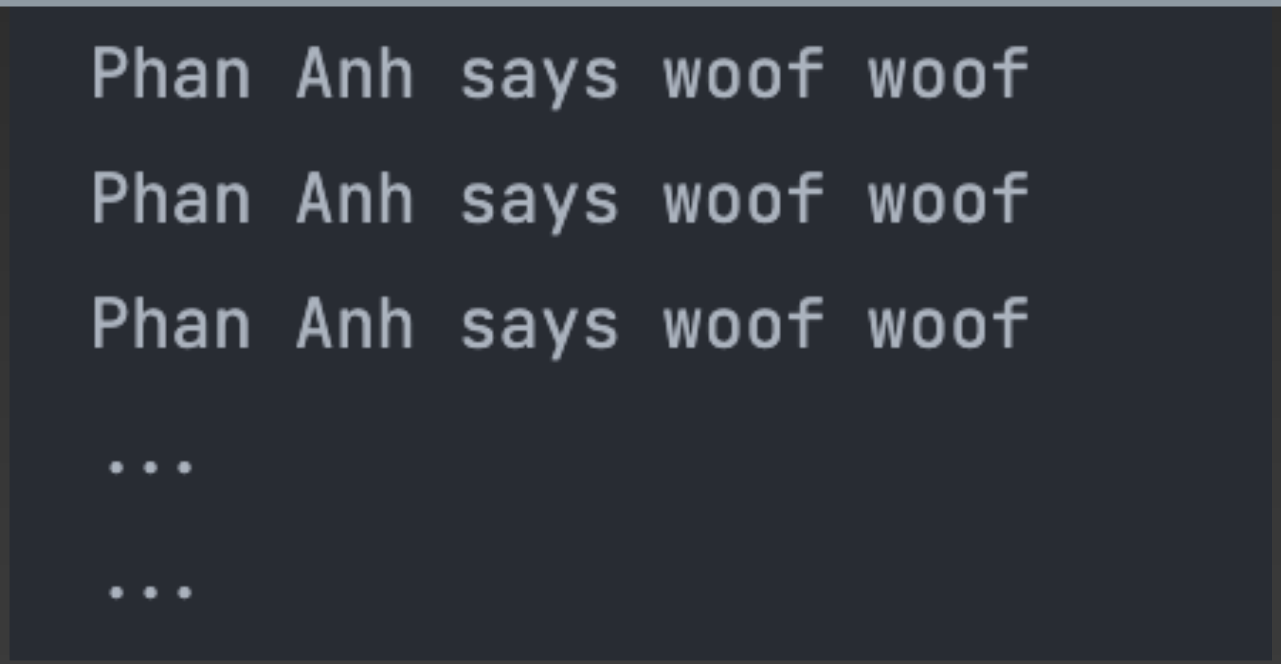
```python
class Dog:
    def __init__(self, name, maxBarkTime=3):
        self.__timeBarked = 0                   # (1)
        self.__maxBarkTime = maxBarkTime        # (2)
        self.__name = name

    def __isReachedMaxBarkTime(self):           # (3)
        return self.__timeBarked >= self.__maxBarkTime

    def bark(self):
        if self.__isReachedMaxBarkTime():       # (4)
            return '...'
        else:
            self.__timeBarked += 1
            return f'{self.__name} woof woof'
```

another_library.py

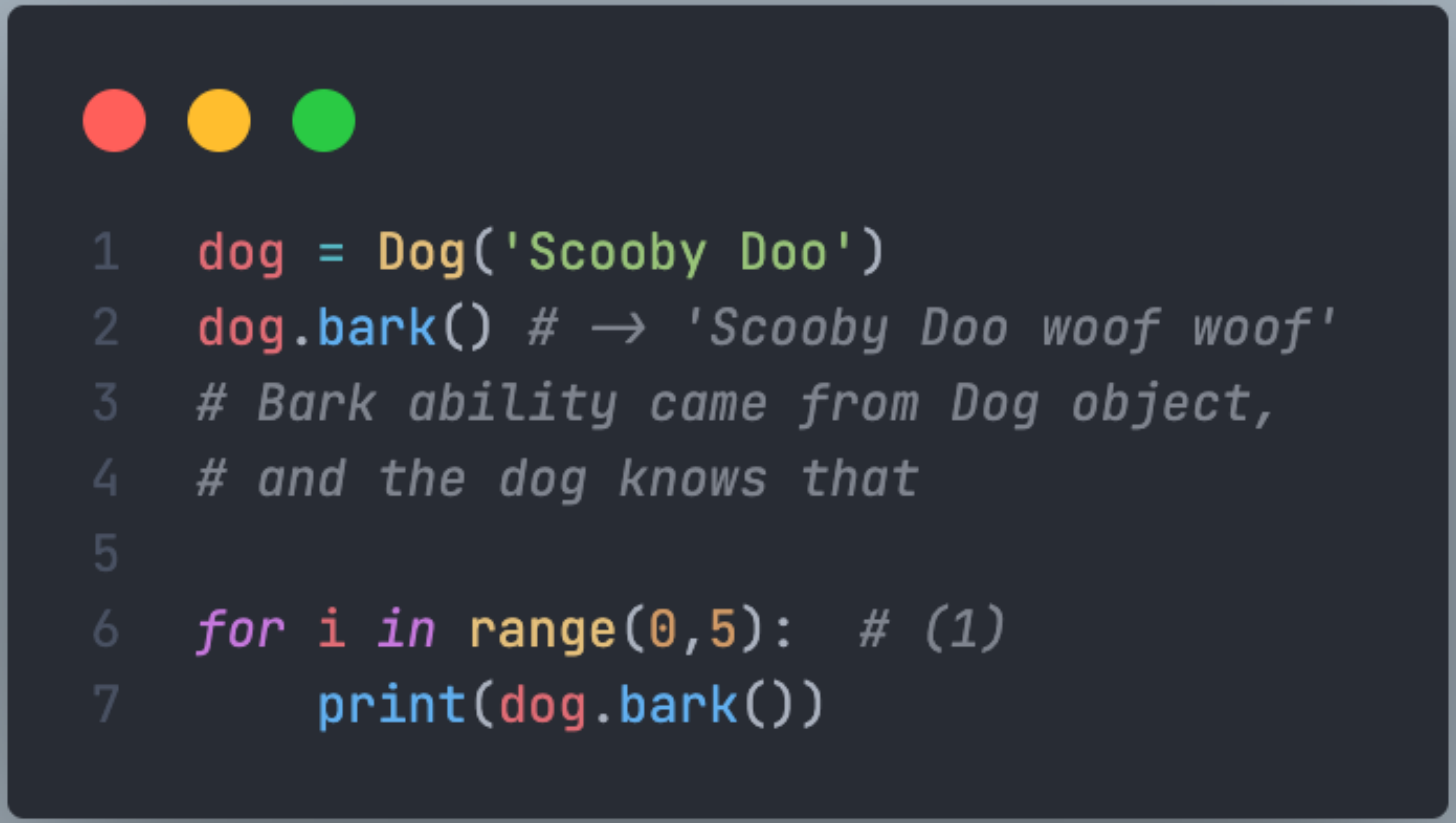


```python
from another_library import *

target = 'Phan Anh'
bark(target) # → 'Phan Anh says woof woof'
# Author note: why must I do this?

maxBarkTime = 3          # (1)
barkIndex = 0            # (2)

for i in range(0,5):     # (3)
    if (i ≥ maxBarkTime):
        print('...')
    else:
        print(bark(target))
    barkIndex += 1
```

Manually; No state

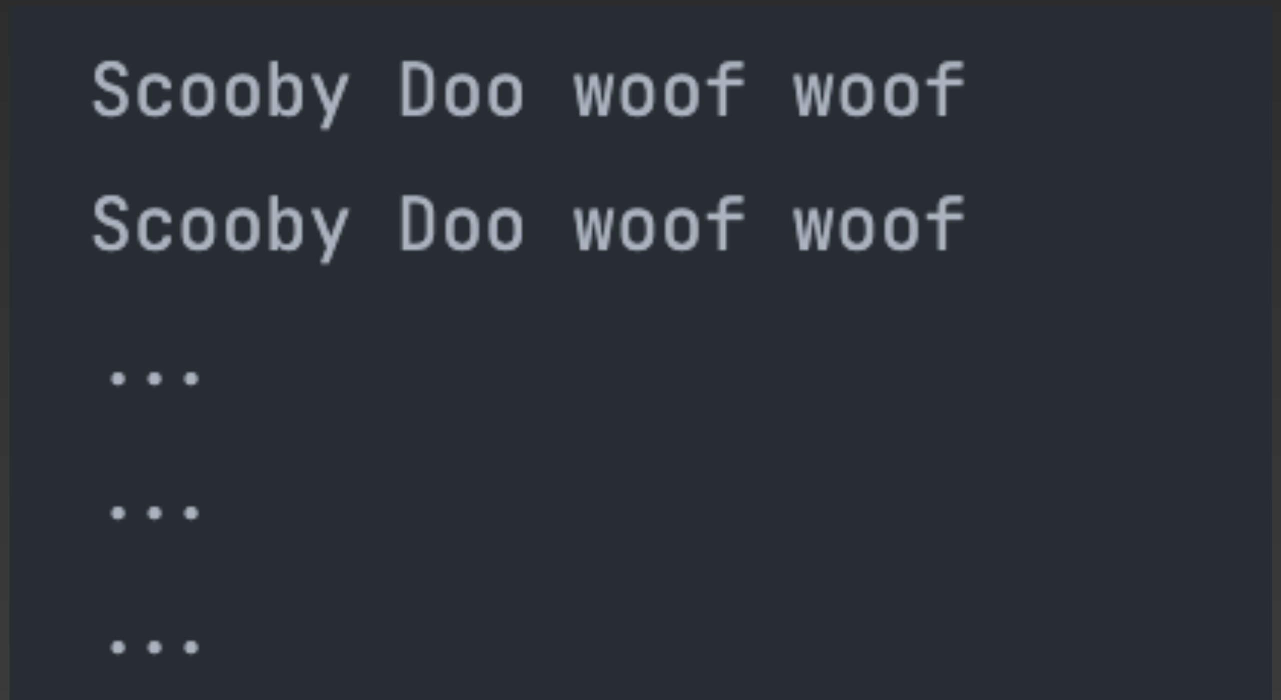


```
Phan Anh says woof woof
Phan Anh says woof woof
Phan Anh says woof woof
...
...
```



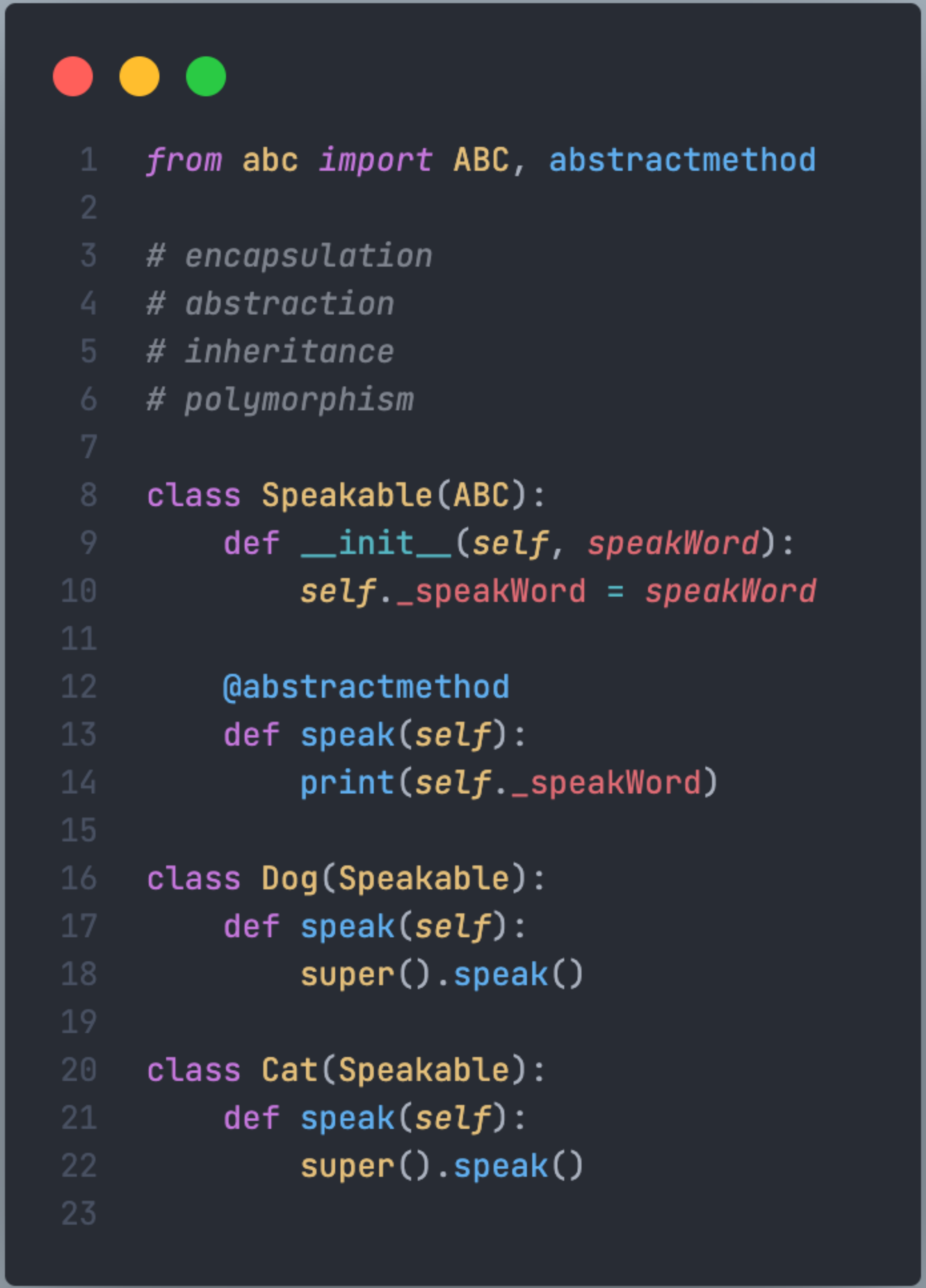
```python
dog = Dog('Scooby Doo')
dog.bark() # → 'Scooby Doo woof woof'
# Bark ability came from Dog object,
# and the dog knows that

for i in range(0,5):  # (1)
    print(dog.bark())
```

Has state



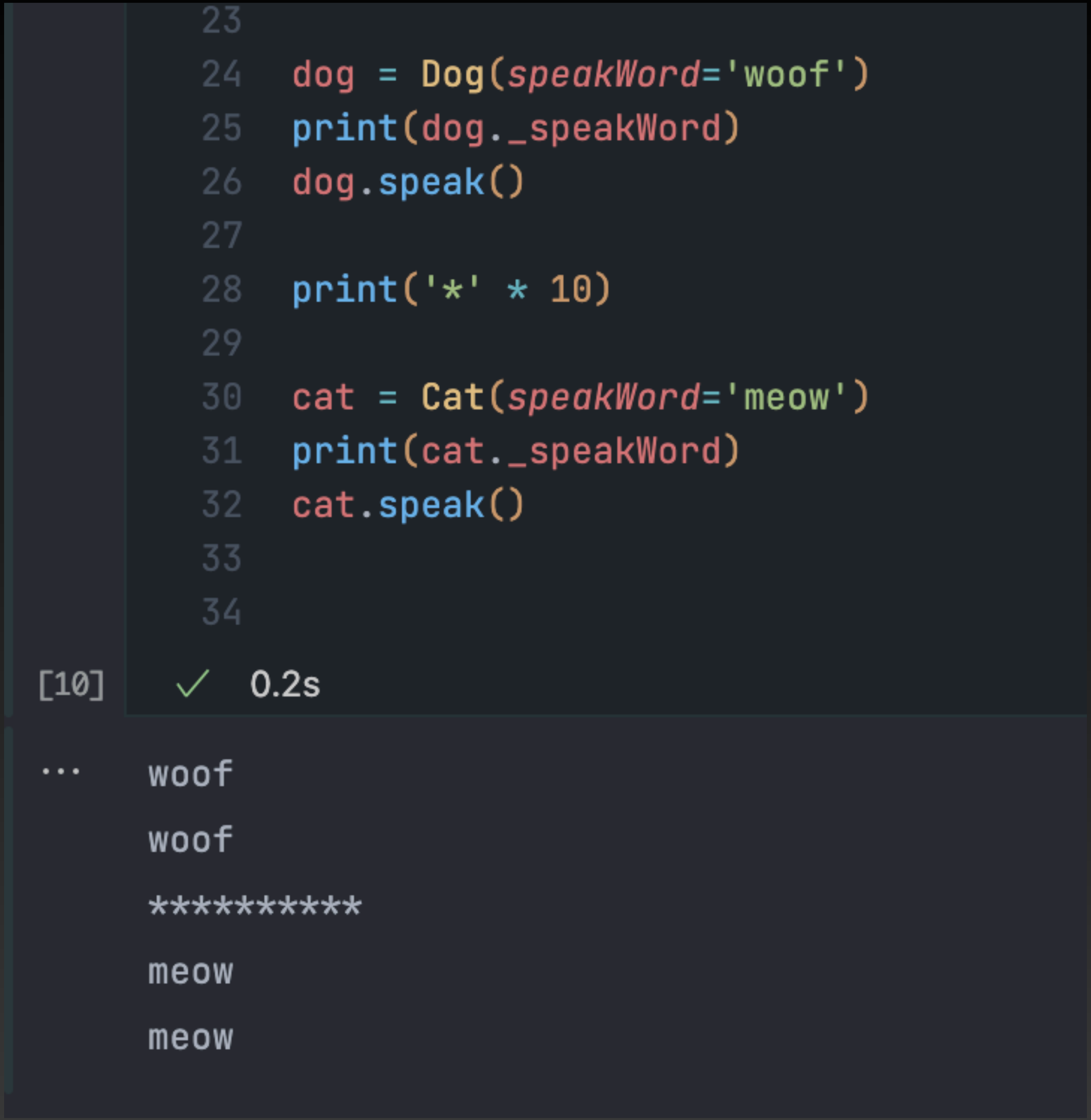
```
Scooby Doo woof woof
Scooby Doo woof woof
...
...
...
```

playground.ipynb



```python
from abc import ABC, abstractmethod

# encapsulation
# abstraction
# inheritance
# polymorphism

class Speakable(ABC):
    def __init__(self, speakWord):
        self._speakWord = speakWord

    @abstractmethod
    def speak(self):
        print(self._speakWord)

class Dog(Speakable):
    def speak(self):
        super().speak()

class Cat(Speakable):
    def speak(self):
        super().speak()

```

oop.ipynb



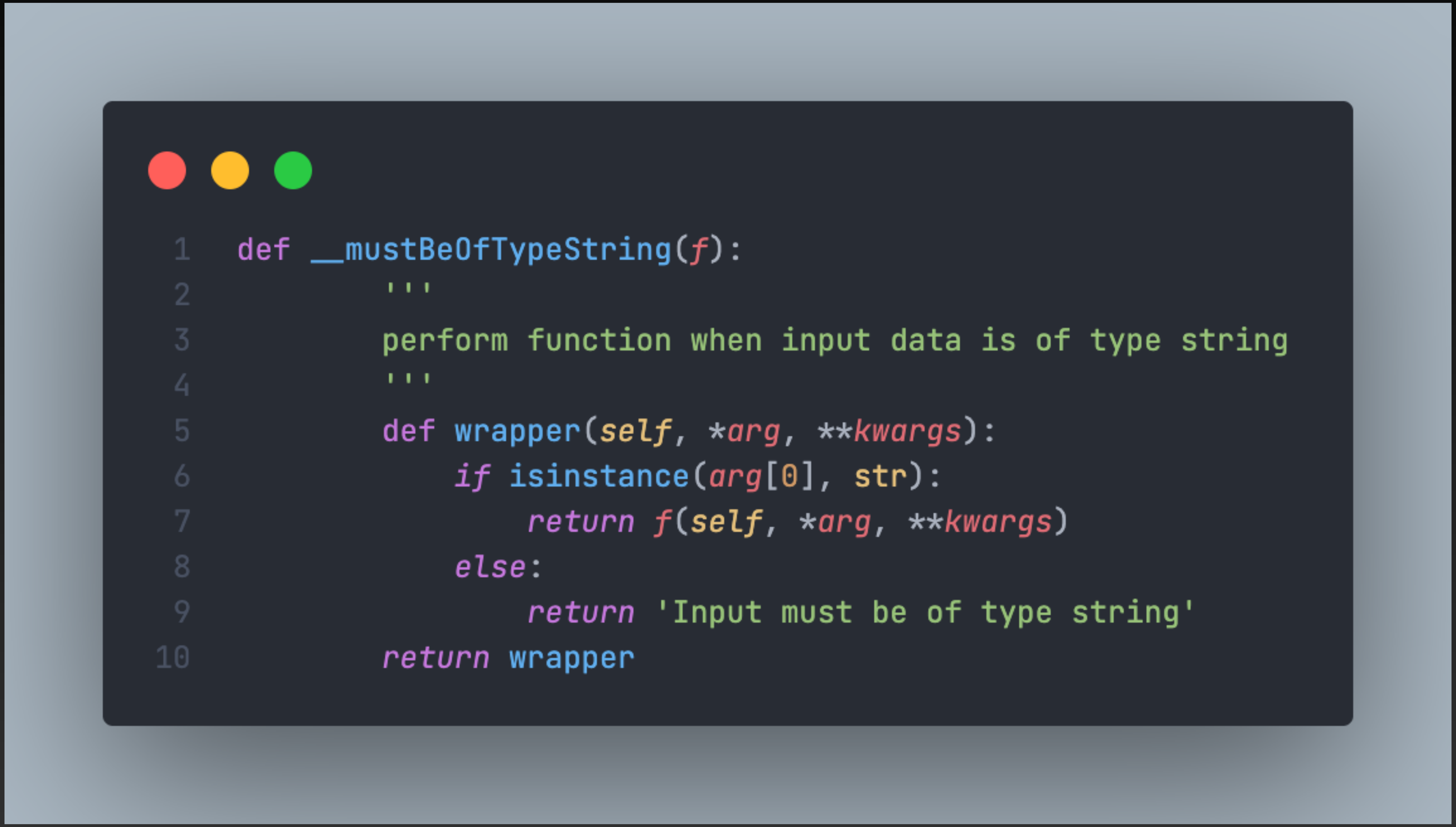
```python

dog = Dog(speakWord='woof')
print(dog._speakWord)
dog.speak()

print('*' * 10)

cat = Cat(speakWord='meow')
print(cat._speakWord)
cat.speak()


```

[10] ✓ 0.2s

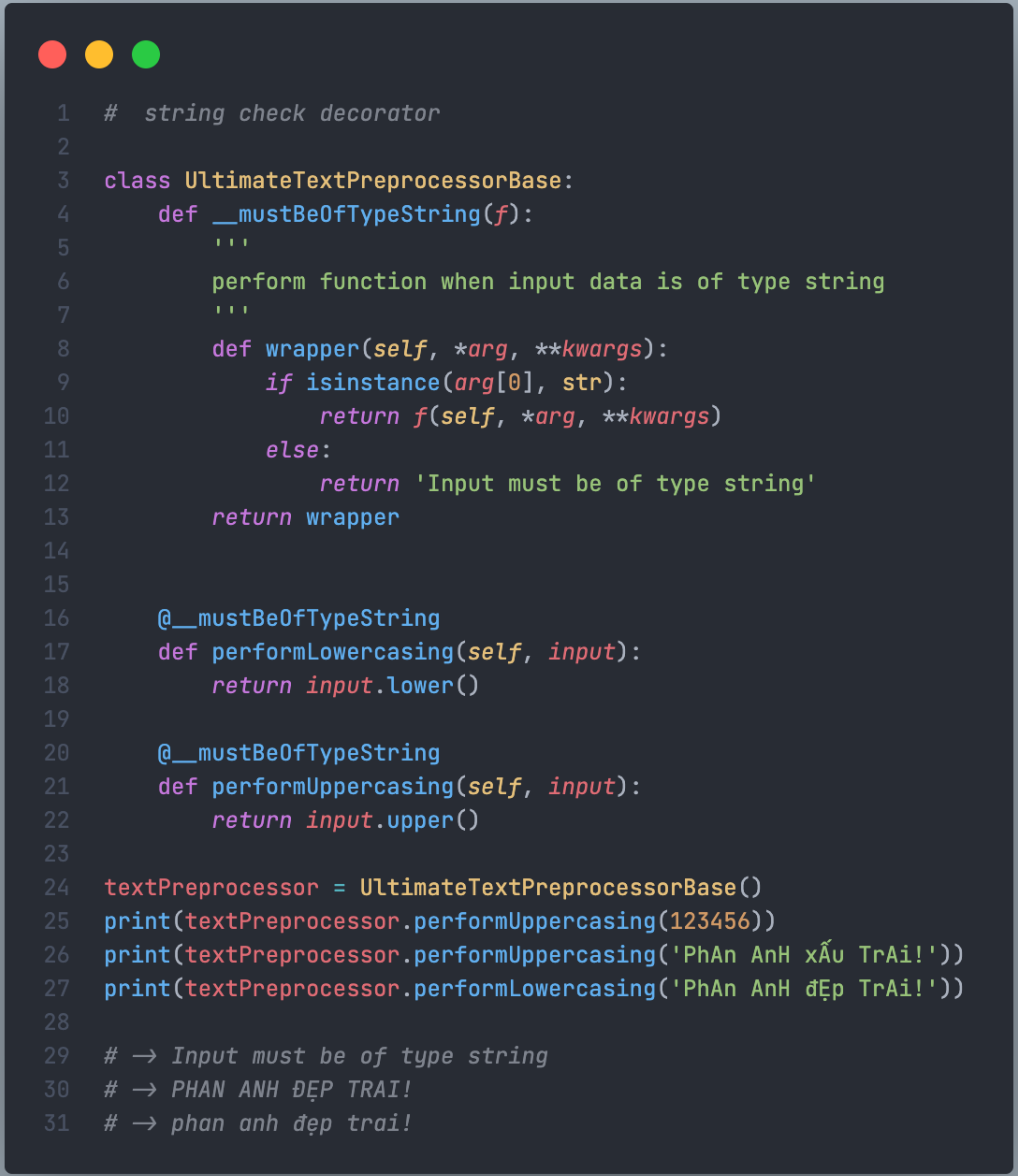
```
woof
woof
**********
meow
meow
```

Decorator

Is it Christmas already?



```python
def __mustBeOfTypeString(f):
        '''
        perform function when input data is of type string
        '''
        def wrapper(self, *arg, **kwargs):
            if isinstance(arg[0], str):
                return f(self, *arg, **kwargs)
            else:
                return 'Input must be of type string'
        return wrapper
```



```python
#  string check decorator

class UltimateTextPreprocessorBase:
    def __mustBeOfTypeString(f):
        '''
        perform function when input data is of type string
        '''
        def wrapper(self, *arg, **kwargs):
            if isinstance(arg[0], str):
                return f(self, *arg, **kwargs)
            else:
                return 'Input must be of type string'
        return wrapper


    @__mustBeOfTypeString
    def performLowercasing(self, input):
        return input.lower()

    @__mustBeOfTypeString
    def performUppercasing(self, input):
        return input.upper()

textPreprocessor = UltimateTextPreprocessorBase()
print(textPreprocessor.performUppercasing(123456))
print(textPreprocessor.performUppercasing('PhAn AnH xẤu TrAi!'))
print(textPreprocessor.performLowercasing('PhAn AnH đẸp TrAi!'))

# -> Input must be of type string
# -> PHAN ANH ĐẸP TRAI!
# -> phan anh đẹp trai!
```

decorator.ipynb