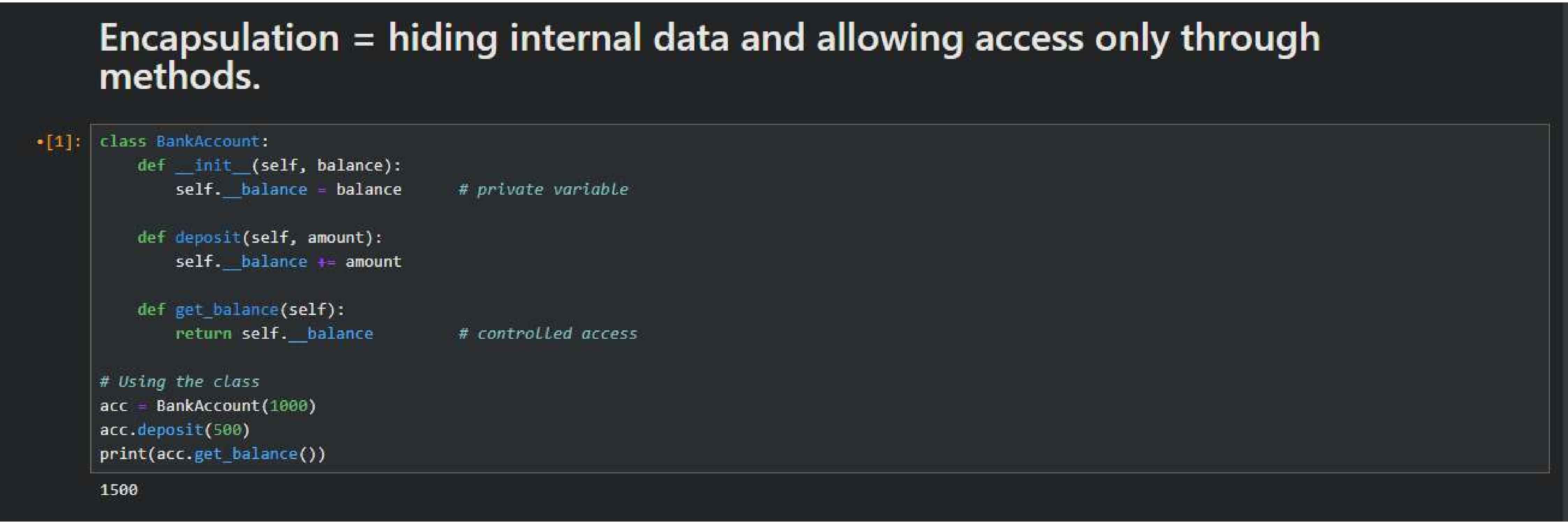


## Encapsulation = hiding internal data and allowing access only through methods.

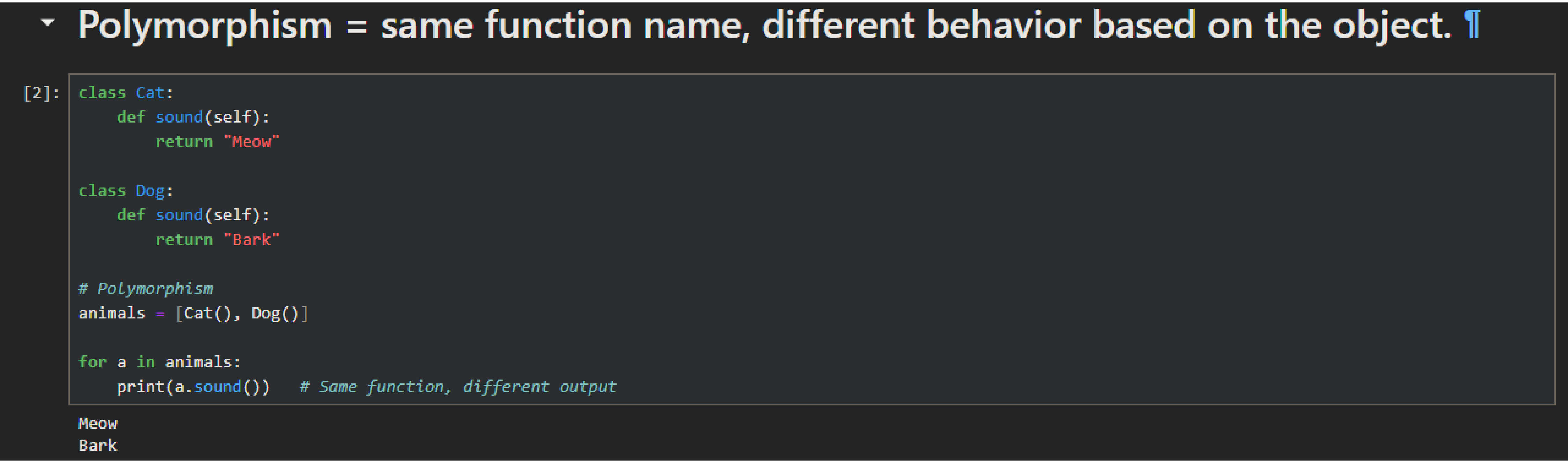
•[1]:
```python
class BankAccount:
    def __init__(self, balance):
        self.__balance = balance      # private variable

    def deposit(self, amount):
        self.__balance += amount

    def get_balance(self):
        return self.__balance         # controlled access

# Using the class
acc = BankAccount(1000)
acc.deposit(500)
print(acc.get_balance())
```

```
1500
```



## Polymorphism = same function name, different behavior based on the object. ¶

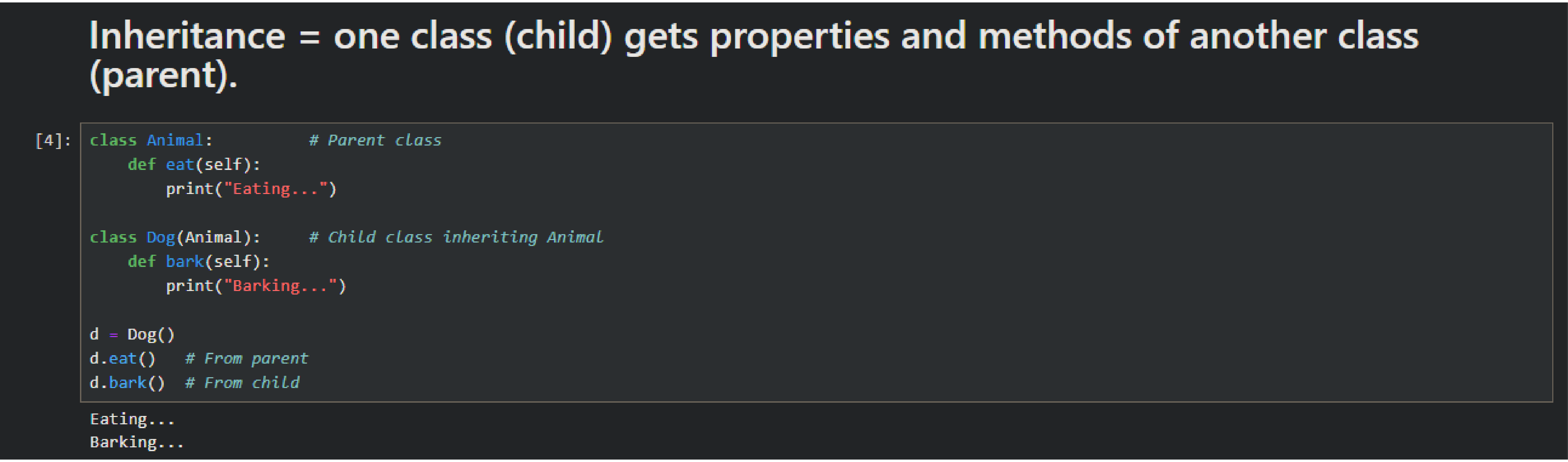
[2]:
```python
class Cat:
    def sound(self):
        return "Meow"

class Dog:
    def sound(self):
        return "Bark"

# Polymorphism
animals = [Cat(), Dog()]

for a in animals:
    print(a.sound())   # Same function, different output
```

```
Meow
Bark
```



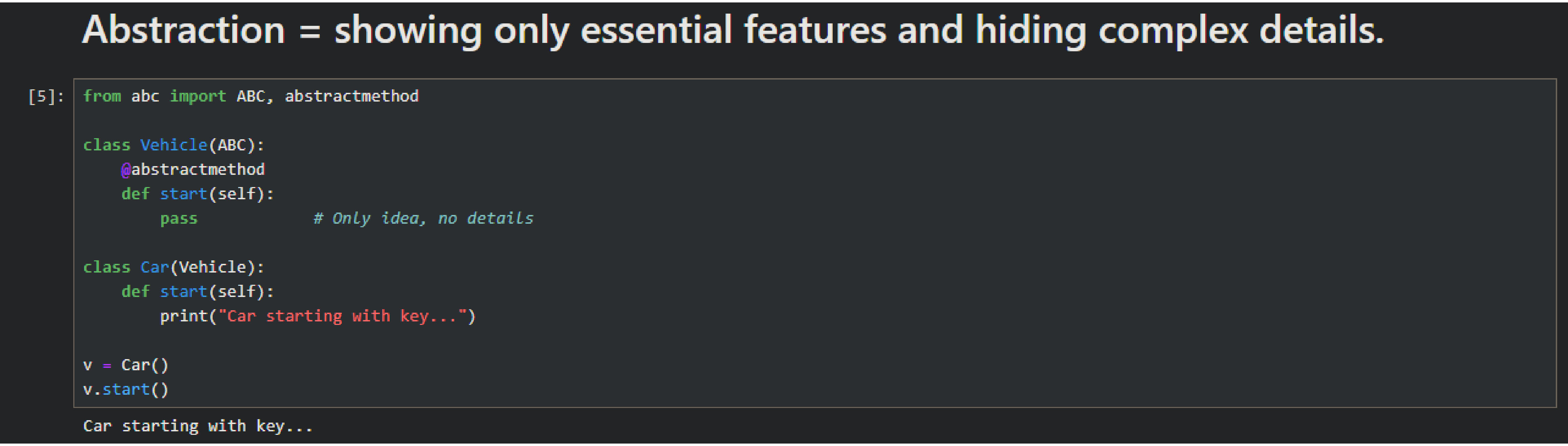
## Inheritance = one class (child) gets properties and methods of another class (parent).

[4]:
```python
class Animal:          # Parent class
    def eat(self):
        print("Eating...")

class Dog(Animal):     # Child class inheriting Animal
    def bark(self):
        print("Barking...")

d = Dog()
d.eat()   # From parent
d.bark()  # From child
```

```
Eating...
Barking...
```



## Abstraction = showing only essential features and hiding complex details.

```
[5]: from abc import ABC, abstractmethod

     class Vehicle(ABC):
         @abstractmethod
         def start(self):
             pass            # Only idea, no details

     class Car(Vehicle):
         def start(self):
             print("Car starting with key...")

     v = Car()
     v.start()
```

```
Car starting with key...
```