Object-Oriented Programming with Classes

Classes, Inheritance & More

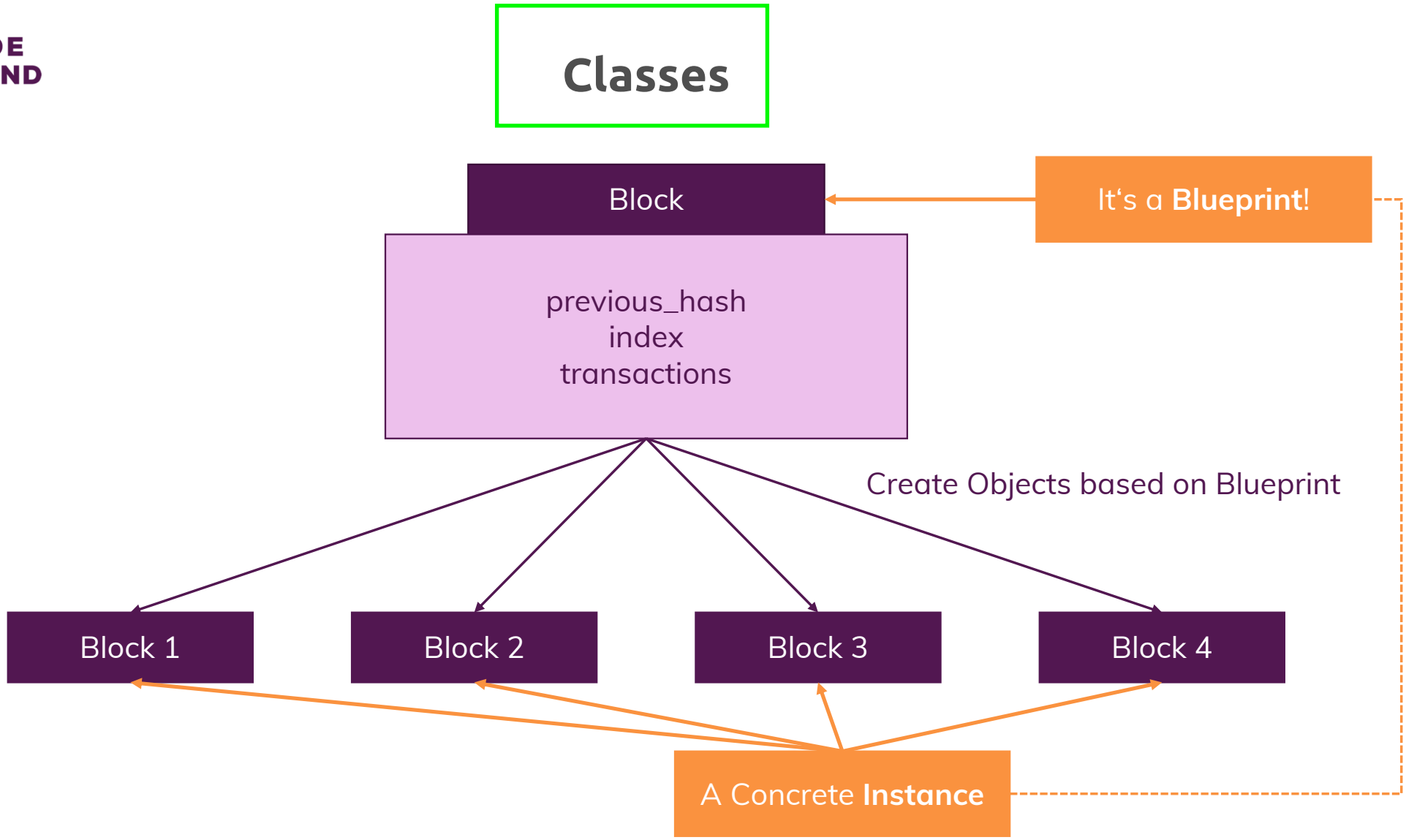
Module Overview

- Classes & Objects
- Attributes & Methods
- Inheritance

What about the Blockchain?

What Is Object-Oriented Programming?

Procedural	Object-Oriented
`blockchain = []`	`class Blockchain:`
`def load_data():`	`blockchain = Blockchain()`
`load_data():`	`blockchain.load_data()`
…	…
`while running:`	`user_interface = UI()`
Execute Steps Sequentially	Use Classes as Blueprints of Objects (Data Structures)
Code is Relatively "Unstructured"	Code is Structured in Objects


Classes
Block
It's a **Blueprint**!
previous_hash
index
transactions
Create Objects based on Blueprint
Block 1
Block 2
Block 3
Block 4
A Concrete **Instance**

Create a class

car.py

9_面向对象编程 ▸ 9-oop-examples ▸ car.py ▸ Car

```python
class Car:
    top_speed = 100

    def drive(self):
        print('I am driving but certainly not faster than {}'.format(self.top_speed))

car1 = Car()
car1.drive()
```

```
pz (master *) 9-oop-examples $ python car.py
I am driving but certainly not faster than 100
pz (master *) 9-oop-examples $
```

Classes vs Objects (Instances) vs Dictionaries

A Class

An Object

An Instance

An Object is istantiated based on a Class (the Blueprint)

A Blueprint for an Object

Doesn't do anything if no Object is created (like a Function that isn't called)

An Object which holds Data and is used in your Code

Multiple Objects can be instantiated from the same Class

A Dictionary

Can hold equal Data as an Object

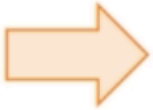
Not based on a custom Blueprint

No specific Structure is guaranteed/ enforced

Why Use Classes?

Classes are Blueprint for Objects

Objects have exactly the Structure you Need



And can be Created Quickly

An Object can also include Methods (not just Fields/ Attributes)

Object-Oriented Programming allows you to write Clean Code

Class attributes

2-car-class-attributes.py ×

9_面向对象编程 ▸ 9-oop-examples ▸ 2-car-class-attributes.py ▸ ...

```python
class Car:
    top_speed = 100
    warnings = []

    def drive(self):
        print('I am driving but certainly not faster than {}'.format(self.top_speed))

car1 = Car()
car1.drive()

Car.top_speed = 200

car2 = Car()
car2.drive()

car3 = Car()
car3.drive()
```

修改类的属性：生成对象的属性因此改变

```
pz (master *) 9-oop-examples $ python 2-car-class-attributes.py
I am driving but certainly not faster than 100
I am driving but certainly not faster than 200
I am driving but certainly not faster than 200
pz (master *) 9-oop-examples $
```

2-car-class-attributes.py ×

9_面向对象编程 ▸ 9-oop-examples ▸ 2-car-class-attributes.py ▸ ...

```python
class Car:
    top_speed = 100
    warnings = []

    def drive(self):
        print('I am driving but certainly not faster than {}'.format(self.top_speed))

car1 = Car()
car1.drive()

# Car.top_speed = 200
car1.warnings.append('New warning')

car2 = Car()
car2.drive()
print(car2.warnings)

car3 = Car()
car3.drive()
print(car3.warnings)
```

如果这么定义的话（定义的是类的属性），修改对象的属性：
新生成对象的属性会怎样？

修改对象（注意不是类）的属性：新生成对象的属性会怎样？

新生成对象的属性：也改变了　　说明：修改了类的属性。所以：所有新生成对象的属性都随之改变

```
pz (master *) 9-oop-examples $ python 2-car-class-attributes.py
I am driving but certainly not faster than 100
I am driving but certainly not faster than 100
['New warning']
I am driving but certainly not faster than 100
['New warning']
pz (master *) 9-oop-examples $
```

Constructor and instance attributes

3-car-constructor-instance-attributes.py ×

9_面向对象编程 ▸ 9-oop-examples ▸ 3-car-constructor-instance-attributes.py ▸ ...

```python
class Car:
    # top_speed = 100
    # warnings = []
    def __init__(self, starting_top_speed=100):
        self.top_speed = starting_top_speed
        self.warnings = []

    def drive(self):
        print('I am driving but certainly not faster than {}'.format(self.top_speed))

car1 = Car()
car1.drive()

# Car.top_speed = 200
car1.warnings.append('New warning')
print(car1.warnings)

car2 = Car(200)
car2.drive()
print(car2.warnings)

car3 = Car(250)
car3.drive()
print(car3.warnings)
```

Define attributes in special built-in constructor function: gives instance attributes which are in the scope this instance.
Therefore, no longer share the attributes in class, avoiding unpredictable problems.

```
pz (master *) 9-oop-examples $ python 3-car-constructor-instance-attributes.py
I am driving but certainly not faster than 100
['New warning']
I am driving but certainly not faster than 200
[]
I am driving but certainly not faster than 250
[]
pz (master *) 9-oop-examples $
```

The Constructor

```
def __init__(self):
```

If you don't define your own one, a default (empty) one gets used

Allows you to run Initialization Code

Allows you to define instance-scoped Attributes

```
__init__(self):
        self.result = 5
```

Printing classes with special methods

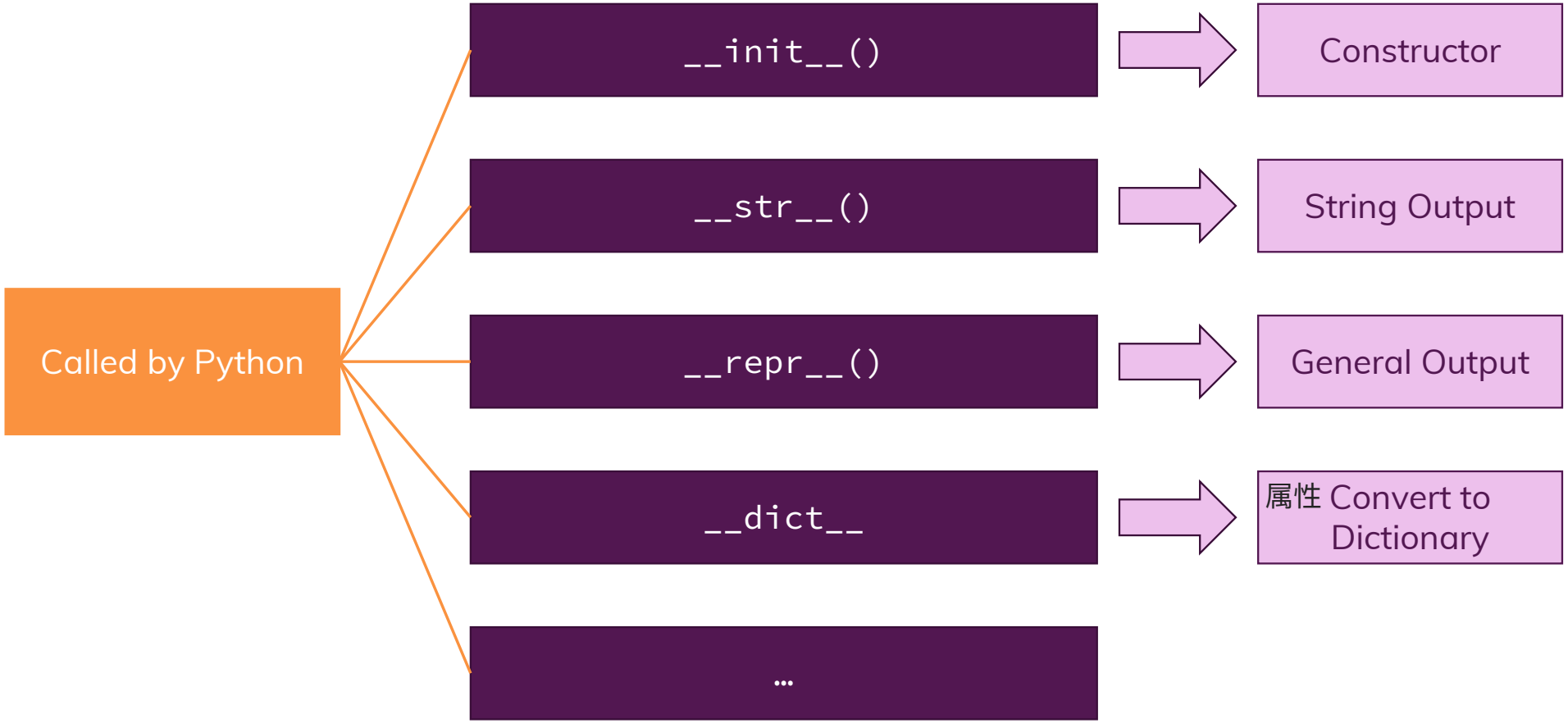
4-car-printing-classes-with-special-methods.py ×

9_面向对象编程 ▸ 9-oop-examples ▸ 4-car-printing-classes-with-special-methods.py ▸ ...

```python
class Car:
    # top_speed = 100
    # warnings = []
    def __init__(self, starting_top_speed=100):
        self.top_speed = starting_top_speed
        self.warnings = []

    def drive(self):
        print('I am driving but certainly not faster than {}'.format(self.top_speed))


car1 = Car()
car1.drive()

# Car.top_speed = 200
car1.warnings.append('New warning')
print(car1)
print(car1.__dict__)

car2 = Car(200)
car2.drive()
print(car2.warnings)

car3 = Car(250)
car3.drive()
print(car3.warnings)
```

```
pz (master *) 9-oop-examples $ python 4-car-printing-classes-with-special-methods.py
I am driving but certainly not faster than 100
<__main__.Car object at 0x1109b9cf8>
{'top_speed': 100, 'warnings': ['New warning']}
I am driving but certainly not faster than 200
[]
I am driving but certainly not faster than 250
[]
pz (master *) 9-oop-examples $
```

Special Methods



4-car-printing-classes-with-special-methods.py ×

9_面向对象编程 ▸ 9-oop-examples ▸ 4-car-printing-classes-with-special-methods.py ▸ Car

```python
class Car:
    # top_speed = 100
    # warnings = []
    def __init__(self, starting_top_speed=100):
        self.top_speed = starting_top_speed
        self.warnings = []

    def __repr__(self):
        print('Printing...')

    def drive(self):
        print('I am driving but certainly not faster than {}'.format(self.top_speed))


car1 = Car()
car1.drive()
car1.warnings.append('New warning')
# print(car1.__dict__)
print(car1)

car2 = Car(200)
car2.drive()
print(car2.warnings)

car3 = Car(250)
car3.drive()
print(car3.warnings)

```

重写 overwrite 这个通用输出的 special method

```
pz (master *) 9-oop-examples $ python 4-car-printing-classes-with-special-methods.py
I am driving but certainly not faster than 100
Printing...
Traceback (most recent call last):
  File "4-car-printing-classes-with-special-methods.py", line 19, in <module>
    print(car1)
TypeError: __str__ returned non-string (type NoneType)
pz (master *) 9-oop-examples $
```

但是，__repr__ 需要返回的是 string 类型的数据
原来不是直接打印，而是输出字符串！

4-car-printing-classes-with-special-methods.py ×

9_面向对象编程 ▸ 9-oop-examples ▸ 4-car-printing-classes-with-special-methods.py ▸ Car ▸ __repr__

```python
class Car:
    # top_speed = 100
    # warnings = []
    def __init__(self, starting_top_speed=100):
        self.top_speed = starting_top_speed
        self.warnings = []

    def __repr__(self):
        print('Printing...')
        return 'Top Speed: {}, Warnings: {}'.format(self.top_speed, len(self.warnings))

    def drive(self):
        print('I am driving but certainly not faster than {}'.format(self.top_speed))


car1 = Car()
car1.drive()
car1.warnings.append('New warning')
# print(car1.__dict__)
print(car1)

car2 = Car(200)
car2.drive()
print(car2.warnings)

car3 = Car(250)
car3.drive()
print(car3.warnings)
```

输出字符串

```
pz (master *) 9-oop-examples $ python 4-car-printing-classes-with-special-methods.py
I am driving but certainly not faster than 100
Printing...
Top Speed: 100, Warnings: 1
I am driving but certainly not faster than 200
[]
I am driving but certainly not faster than 250
[]
pz (master *) 9-oop-examples $
```

Public vs. Private attributes

5-car-public-vs-private-attributes.py

9_面向对象编程 ▸ 9-oop-examples ▸ 5-car-public-vs-private-attributes.py ▸ ...

```python
class Car:
    # top_speed = 100
    # warnings = []
    def __init__(self, starting_top_speed=100):
        self.top_speed = starting_top_speed
        self.__warnings = []

    def __repr__(self):
        print('Printing...')
        return 'Top Speed: {}, Warnings: {}'.format(self.top_speed, len(self.__warnings))

    def add_warning(self, warning_text):
        if len(warning_text) > 0:
            self.__warnings.append(warning_text)

    def get_warnings(self):
        return self.__warnings

    def drive(self):
        print('I am driving but certainly not faster than {}'.format(self.top_speed))
```

double underscore: special attributes, private attributes

操作

读取

定义了 private 的属性
Encapsulation 封装：读取和操作对象的属性，不能再直接call这个属性了，
就是说属性变成内部资料了，要读取和操作内部资料需要使用这个对象定义好的方法。

Encapsulation (computer programming) - Wikipedia翻译此页 I 中文网页
2019-6-18 · Encapsulation can be used to hide data members and member functions. Under this definition, encapsulation means that the internal representation of an object is generally hidden from view outside of the object's definition. Typically, only the object's own methods can directly inspect or manipulate its fields.

定义private属性后，试图直接操作该属性

```
21
22
23  car1 = Car()
24  car1.drive()
25
26  car1.__warnings.append([])
27
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
pz (master *) 9-oop-examples $ python 5-car-public-vs-private-attributes.py
I am driving but certainly not faster than 100
Traceback (most recent call last):
  File "5-car-public-vs-private-attributes.py", line 26, in <module>
    car1.__warnings.append([])
AttributeError: 'Car' object has no attribute '__warnings'
pz (master *) 9-oop-examples $
```

调用定义好的 member function
来操作

```
22
23  car1 = Car()
24  car1.drive()
25
26  # car1.__warnings.append([])
27  car1.add_warning('New warning')
28  # print(car1.__dict__)
29  print(car1)
30
```

OK

```
pz (master *) 9-oop-examples $ python 5-car-public-vs-private-attributes.py
I am driving but certainly not faster than 100
Printing...
Top Speed: 100, Warnings: 1
I am driving but certainly not faster than 200
[]
I am driving but certainly not faster than 250
[]
pz (master *) 9-oop-examples $
```

尝试直接读取私有属性

```
30
31    car2 = Car(200)
32    car2.drive()
33    print(car2.__warnings)
34
```

不灵
私有属性：不能直接读取

```
pz (master *) 9-oop-examples $ python 5-car-public-vs-private-attributes.py
I am driving but certainly not faster than 100
Printing...
Top Speed: 100, Warnings: 1
I am driving but certainly not faster than 200
Traceback (most recent call last):
  File "5-car-public-vs-private-attributes.py", line 33, in <module>
    print(car2.__warnings)
AttributeError: 'Car' object has no attribute '__warnings'
pz (master *) 9-oop-examples $
```

改为调用定义好的方法来读取

```
30
31  car2 = Car(200)
32  car2.drive()
33  # print(car2.__warnings)
34  print(car2.get_warnings())
35
36  car3 = Car(250)
37  car3.drive()
38  print(car3.get_warnings())
39
```

OK

```
pz (master *) 9-oop-examples $ python 5-car-public-vs-private-attributes.py
I am driving but certainly not faster than 100
Printing...
Top Speed: 100, Warnings: 1
I am driving but certainly not faster than 200
[]
I am driving but certainly not faster than 250
[]
pz (master *) 9-oop-examples $
```