

Lecture 9 - Python Class

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

CSC-1004: Computational Laboratory Using Java

Course Page: [\[Click\]](#)

Python Classes/Objects

Python is an object-oriented programming language.

- Almost everything in Python is an object, with its properties and methods.
- A Class is like an object constructor, or a "blueprint" for creating objects.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Python Classes/Objects

Python is an object-oriented programming language.

- To create a class, use the keyword class:

```
class MyClass:  
    x = 5
```

- We can use the class named MyClass to create objects:

```
p1 = MyClass()  
print(p1.x)
```



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Python Classes/Objects

Python is an **object-oriented** programming language.

- All classes have a function called `__init__()`, which is always executed when the class is **being initiated**.

```
class Person:  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age
```

```
p1 = Person("John", 36)  
print(p1.name)  
print(p1.age)
```



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Python Classes/Objects

Python is an object-oriented programming language.

- The **self** parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def myfunc(self):
        print("Hello my name is " + self.name)
p1 = Person("John", 36)
p1.myfunc()
```



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Python Classes/Objects

Python is an object-oriented programming language.

- The underscore `__` parameter is a indicator private attributes. It cannot be accessed directly outside the class.

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
        self.__deposit = 1000
    def print_deposit(self):
        print("The deposit is " + self.__deposit)
```

"p1.__deposit()" is not allowed, but "p1.print_deposit()" is ok.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Python Classes/Objects

Python is an object-oriented programming language.

- The `__str__()` function controls what should be returned when the class object is represented as a string.

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def __str__(self):
        return f"self.name(self.age)"
p1 = Person("John", 36)
print(p1)
```



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Python Classes/Objects

Python is an object-oriented programming language.

- You can modify properties on objects like this:

```
p1.age = 40
```

- You can delete properties on objects by using the del keyword:

```
del p1.age
```

- You can delete objects by using the del keyword:

```
del p1
```



Python Inheritance

Inheritance allows us to define a class that inherits **all the methods and properties** from another class.

- **Parent class** is the class **being inherited from**, also called base class.
- **Child class** is the class that **inherits from another class**, also called derived class.



Python Inheritance

Create a **Parent Class**.

```
class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname
    def printname(self):
        print(self.firstname, self.lastname)
x = Person("John", "Doe")
x.printname()
```



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Python Inheritance

Create a **Child Class**.

```
class Student(Person):  
    def __init__(self, fname, lname):  
        super().__init__(fname, lname)
```

- Add the **`__init__()`** function to the child class.
- Use the **`super()`** function to make the child class **inherit all the methods and properties** from its parent.



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Python Inheritance

Add **properties** to the child class.

```
class Student(Person):  
    def __init__(self, fname, lname):  
        super().__init__(fname, lname)  
        self.graduationyear = 2019  
x = Student("Mike", "Olsen", 2019)
```

- Add a **property** called graduationyear to the Student class.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Python Inheritance

Add **methods** to the child class.

```
class Student(Person):  
    def __init__(self, fname, lname):  
        super().__init__(fname, lname)  
        self.graduationyear = 2019  
    def welcome(self):  
        print("Welcome", self.firstname, self.lastname, self.graduationyear)
```

- Add a method called "welcome" to the Student class.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Python Try Except

- The `try` block lets you test a block of code for errors.
- The `except` block lets you handle the error.
- The `else` block lets you execute code when there is no error.
- The `finally` block lets you execute code, regardless of the result of the try- and except blocks.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Python Try Except

Exception Handling. When an **error** occurs, or **exception** as we call it, Python will normally **stop and generate an error message**.

```
f = open("demofile.txt")
try:
    f.write("Lorum Ipsum")
except:
    print("Something went wrong when writing to the file")
finally:
    f.close()
```



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Python Random

The **random module** gives access to various useful functions one of them being able to **generate random integers**, which is **randint()**.

```
import random  
r1 = random.randint(0, 10)  
print("Random number between 0 and 10 is % d" % (r1))
```

"Why we need random numbers?"

Generate the **location of foods** in the game snake.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Question and Answering (Q&A)



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen