Lesson 8 Python Class and File Operation

1. Class and Object

1.1 Class Introduction

In Python, all data types are considered as object, and objects can be user-defined. User-defined object data type is equal to class in object-oriented. Class is used to describes the object sets with the same properties and methods. And it defines the properties and methods shared by all the objects of this set. Object is the instance of class.

For example, create a “cat” class, and define the name, color, age, etc., as well as the way to eat, drink, sound, etc. And instantiated object “ginger cat” will have these defined properties and methods.

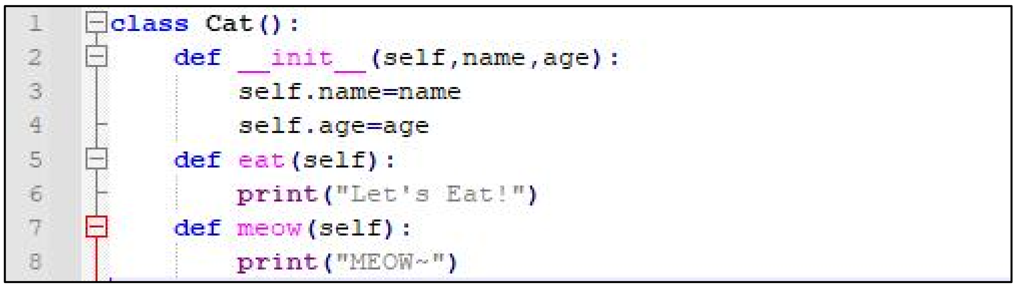
1.2 Class Definition

1) Defined by class keywords, class name usually starts with capital letter.

2) _init_ function initializes class, which is equal to constructor. Whenever creating instance based on class, this function will be ran automatically in Python.

3) self function, implicit parameter and the object itself, is equivalent to this function.

4) Properties correspond to member variables.

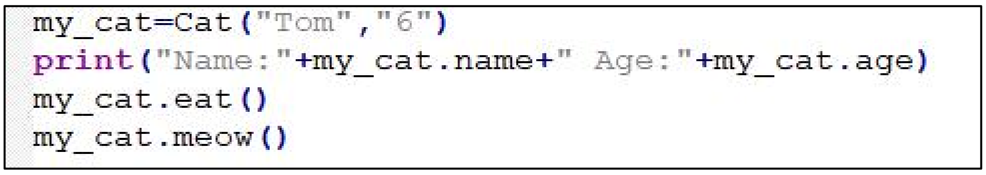


```python
class Cat():
    def __init__(self,name,age):
        self.name=name
        self.age=age
    def eat(self):
        print("Let's Eat!")
    def meow(self):
        print("MEOW~")
```

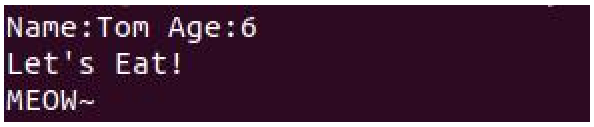
1.3 Instance Creating

1) Create the class name, and input the required properties.

2) These properties and methods can be called directly.



```python
my_cat=Cat("Tom","6")
print("Name:"+my_cat.name+" Age:"+my_cat.age)
my_cat.eat()
my_cat.meow()
```



```
Name:Tom Age:6
Let's Eat!
MEOW~
```

1.4 Property of Class and Instance

1) Instance property are defined within **__init__** of a class, and class property are defined outside __init__ of a class. An instance of a class created outside the class can define its own properties.

2) Personal variable: if don’t want the external to access some properties, we can start the property name with double subscripts (__). Therefore, these properties will not be accessed by original variable name, that is, they can not be called from an instance.

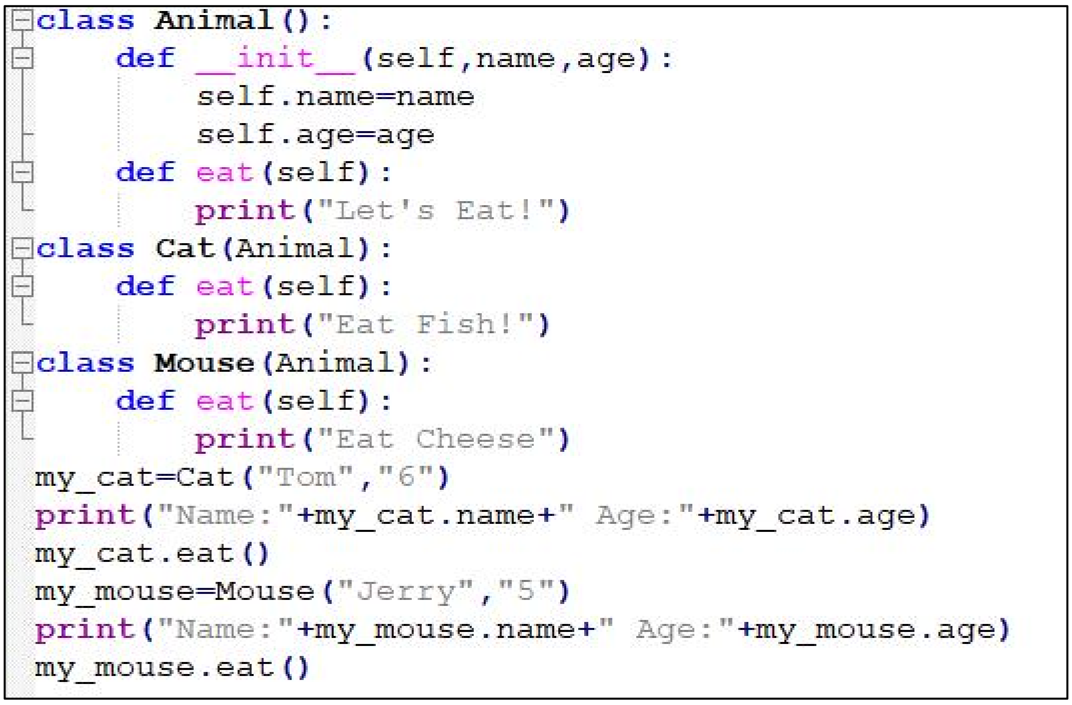
1.5 Class Inheritance and Polymorphism

1) Inheritance: when creating a class, you can write the name of the parent class in parentheses, so that the properties and methods of the parent class are inherited to the child class. If the child class does not have an

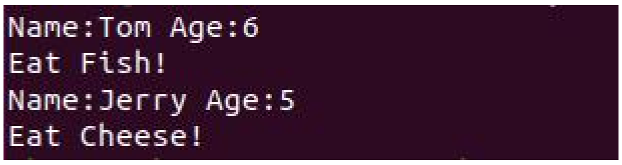
initialization method, the initialization method of the parent class will be called.

2) Method rewriting: If the name, return type and parameter list of a method defined in a child class exactly match those of a method in the parent class, the method of child class rewrites that of parent class.

3) Polymorphism: the behaviors of multiple child classes derived from one parent class are different, which is called polymorphism. In other words, child class rewrite the methods of parent class to let it has methods different from the parent class. When child class have the same method as parent class, the priority will be given to the method of child class, that is, the child class overrides the parent class.



```python
class Animal():
    def __init__(self,name,age):
        self.name=name
        self.age=age
    def eat(self):
        print("Let's Eat!")
class Cat(Animal):
    def eat(self):
        print("Eat Fish!")
class Mouse(Animal):
    def eat(self):
        print("Eat Cheese")
my_cat=Cat("Tom","6")
print("Name:"+my_cat.name+" Age:"+my_cat.age)
my_cat.eat()
my_mouse=Mouse("Jerry","5")
print("Name:"+my_mouse.name+" Age:"+my_mouse.age)
my_mouse.eat()
```



```
Name:Tom Age:6
Eat Fish!
Name:Jerry Age:5
Eat Cheese!
```

After execution, the instance of child class will automatically call the construction method of parent class, and the child class will override the parent class method with the same name.

2. File Operation

2.1 File and File Path

1) Two key properties: “file name” and “path”

2) Absolute path and relative path: “**Absolute Path**” always starts with root folder. “**Relative Path**” corresponds to the current working directory of the program. For example, the relative path of “**test.py**” file under the current working directory is “**/test.py**” and its absolute path is “**python/scripts/test.py**”.

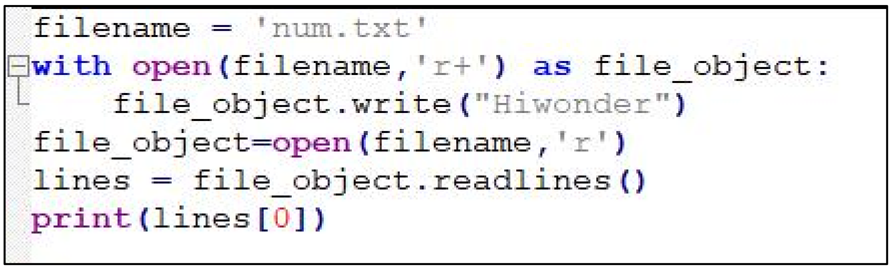
3) (.) and (..) folder: Not real folders, they are the special name that can be used in the path. When a single period (".") is used as a folder directory name, it is an abbreviation for this directory. Two periods ("..") mean the parent folder.

2.2 File Reading Operation

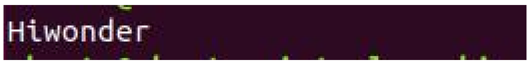
1) Open and close file: call open function to pass in file name and identifier. Open file to return a file object, and then call close() method of File object to close this file. “with” statement can also be used to close the file automatically when ending.

2) File reading: call read function to read the file content to the object. Use readline function to read one line at a time or read all lines at a time, and save them in one list.

3) File writing：When opening a file, you can add designation parameters: read mode ("r"), write mode ("w") [write from the beginning], append mode ("a") [write at the end of the file], read at the same time and write the mode of file ("r+"). You can also write directly.



```python
filename = 'num.txt'
with open(filename,'r+') as file_object:
    file_object.write("Hiwonder")
file_object=open(filename,'r')
lines = file_object.readlines()
print(lines[0])
```



```
Hiwonder
```

After execution, the first line of data just written will be read.