继承

继承指的是一个对象直接使用另一个对象的属性或方法

继承的格式: class 子类名(父类名): ...

继承定义

```python
# 父类
class Person:

    def __init__(self):
        # 构造，初始化属性
        self.name = 'itcast'
        self.age = 20

    def say_hello(self):
        print('hello {}'.format(self.name))

# 子类
class Student(Person):
    pass


if __name__ == '__main__':
    stu = Student()

    print(stu.name)
    print(stu.age)

    stu.say_hello()
```

!!!note
    子类继承父类属性和方法

构造函数

```python
# 父类
class Person:

    def __init__(self, name, age):
        # 构造，初始化属性
        self.name = name
        self.age = age

    def say_hello(self):
        print('hello {}'.format(self.name))


# 子类
class Student(Person):
    pass
```

```
16
17 if __name__ == '__main__':
18     stu = Student('itheima', 30)
19
20     print(stu.name)
21     print(stu.age)
22
23     stu.say_hello()
```

!!!note
    父类中如果构造改变，子类实例需要传入对应的参数。

子类覆写构造

```
1  # 父类
2  class Person:
3
4      def __init__(self, name, age):
5          # 构造，初始化属性
6          self.name = name
7          self.age = age
8
9      def say_hello(self):
10         print('hello {}'.format(self.name))
11
12
13 # 子类
14 class Student(Person):
15
16     def __init__(self, name, age, id):
17         # 需要去加载父类的构造
18         super(Student, self).__init__(name, age)
19         self.id = id
20
21
22 if __name__ == '__main__':
23     stu = Student('itheima', 30, '110')
24
25     print(stu.name)
26     print(stu.age)
27     print(stu.id)
28
29     stu.say_hello()
```

!!! note
    由于业务功能原因，如果子类需要覆写父类的构造，那么：

```
1 * 定义自己的`__init__`构造函数
2 * 在自己的`__init__`中加载父类的构造，见18行
```

Qt窗口继承

继承QWidget

```
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
import sys


class MyWindow(QWidget):

    def __init__(self):
        super(MyWindow, self).__init__()
        self.setWindowTitle('title')


if __name__ == '__main__':
    app = QApplication(sys.argv)

    window = MyWindow()
    window.show()

    sys.exit(app.exec_())
```

!!!note
通过继承`QWidget`来实现窗体

```
在构造中，必须实现`super`函数的调用，否则将出行错误
```

私有化

在python的类中，通常采用`__名称`来定义私有化的属性和函数。

私有化变量

```
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
import sys


class MyWindow(QWidget):

    def __init__(self):
        super(MyWindow, self).__init__()
        self.setWindowTitle('title')

        self.__hello = 'hello'
        self.hi = 'hi'


if __name__ == '__main__':
    app = QApplication(sys.argv)
```

```
19
20      window = MyWindow()
21      window.__hello
22      window.hi
23
24      window.show()
25
26      sys.exit(app.exec_())
```

!!!note
    在类中，定义了`__hello`和`_hi`变量.

```
1  在外部使用过程中：
2
3  * 第21行，`window.__hello`这个访问是不被允许的
4  * 第22行，`window.hi`这个是可以访问的
```

私有化函数

```
1  from PyQt5.QtWidgets import *
2  from PyQt5.QtCore import *
3  from PyQt5.QtGui import *
4  import sys
5
6
7  class MyWindow(QWidget):
8
9      def __init__(self):
10         super(MyWindow, self).__init__()
11         self.setWindowTitle('title')
12         self.__init_ui()
13
14     def __init_ui(self):
15         layout = QHBoxLayout()
16         self.setLayout(layout)
17
18
19 if __name__ == '__main__':
20     app = QApplication(sys.argv)
21
22     window = MyWindow()
23
24     window.__init_ui()
25
26     window.show()
27
28     sys.exit(app.exec_())
```

!!!note
    在类中，定义了`__init_ui`函数.

```
* 第24行，外部调用此函数时，是无法访问的
* 第12行，内部调用是可以的。
```

定义原则

- 当属性只是当前类使用时，就将其作为私有的
- 当函数只是当前类使用时，就将其作为私有的