11中的Lambda表达式 **用于定义并创建匿名的函数对象** ，以此简化编程工作。

基本构成

```
[capture](parameters)mutable->return-type
{
  statement;
}
```

QT5默认支持Lambda表达式

老版本需要在pro文件中加上一句代码 : CONFIG +=C++11

Lambda表达式的声明

```
[ = ](  )
{
  btn=>setText("aaaa");
}(函数调用);
//    =    相当于用值传递的方式让表达式认识了局部成员
```

[ ] 不可缺少，它是一个Lambda表达式的开始

在 [ ] 填充以标识函数对象参数

1. 空。 没有任何函数对象参数
2. =。 函数体内可以使用Lambda所在作用范围内所有可见的局部变量（包括Lambda所在类的this），并且是以值传递的方式。
3. &。 引用传递
4. this。 类似等号
5. a。 将按a的值进行传递（只传递btn）。按值进行传递时，函数体不能修改传递进来的a的拷贝，因为默认情况下函数被const修饰。如果使用该方式时需要修改传递进来的a的拷贝，可以在前面添加 mutable 修饰符。
6. &a。以引用的方式传递a。
7. a,&b。
8. =，&a,&b。
9. &,a,b。 略

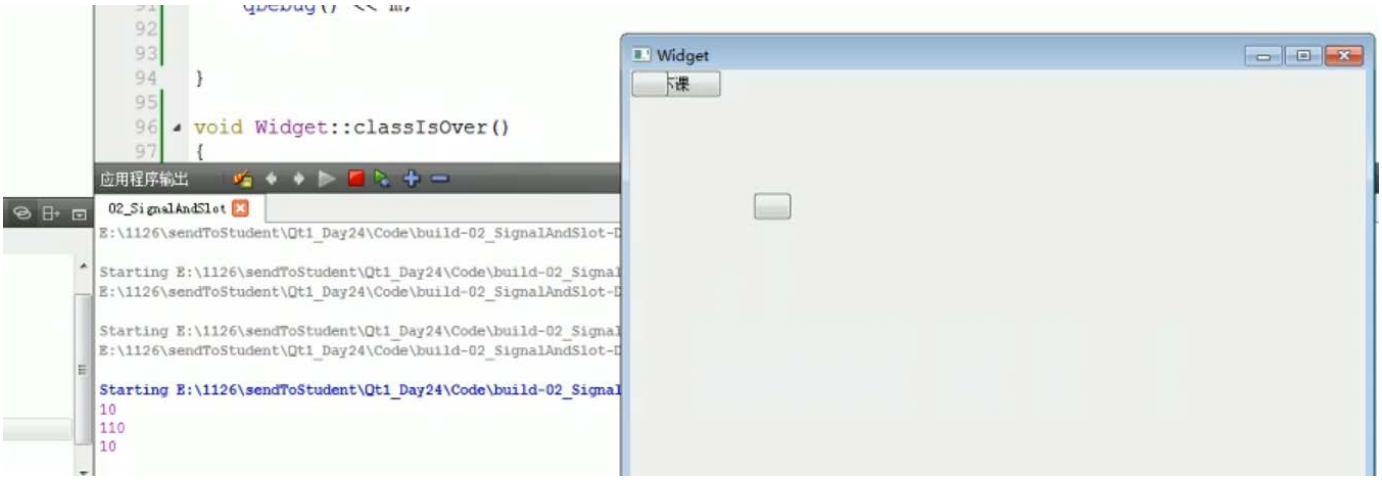
() 表示重载的操作符

```
QPushButton * myBtn = new QPushButton (this);
QPushButton * myBtn2 = new QPushButton (this);
myBtn2->move(100,100);
int m = 10;

connect(myBtn,&QPushButton::clicked,this,[m] ()mutable { m = 100 + 10; qDebug() << m; });

connect(myBtn2,&QPushButton::clicked,this,[=] ()  { qDebug() << m; });

qDebug() << m;
```

可修改标识符 mutable

可以修改按值传递进来的拷贝

需注意，修改的m的是拷贝的值，并非本体。



函数返回值

```
int ret = []()->int{return 1000;}();
```

举例：使用int变量接收返回的int类型数据，int需要加上->放在图示位置

Lambda表达式的使用

```
//利用lambda表达式 实现点击按钮 关闭窗口
QPushButton * btn2 = new QPushButton ;
btn2->setText("关闭");
btn2->move(100,0);
btn2->setParent(this);
connect(btn2,&QPushButton::clicked , this , [=](){
    // this->close();
     emit zt->hungry("宫保鸡丁");
});
```

好处在于：原本clicked在连接时无法连接有参数的，但，使用Lambda时，可以调用任何数据