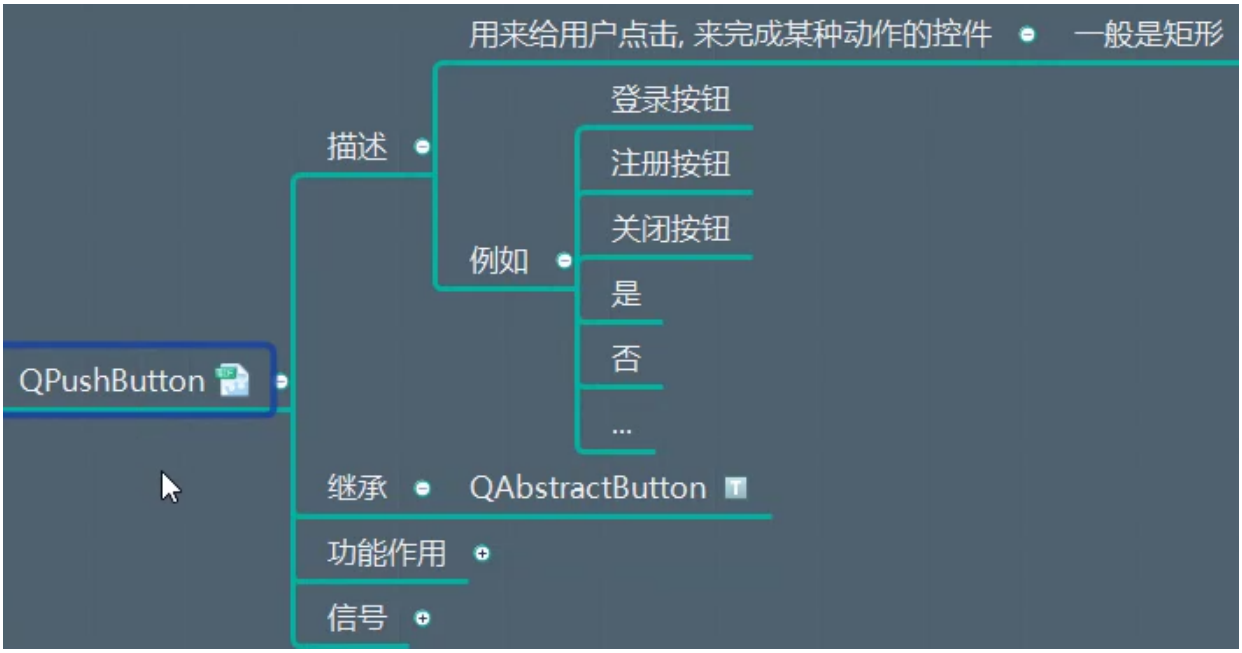
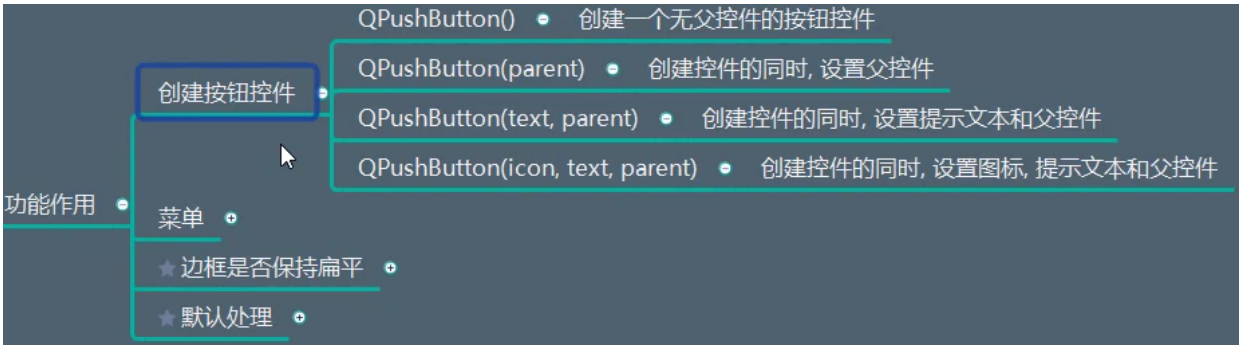
# [笔记][LIKE-Python-GUI编程-PyQt5][10]

PyQt5



# 089. Python-GUI编程-PyQt5-QPushButton-构造函数






`QPushButton` 的构造函数，有多种写法
可以去翻手册：
http://pyqt.sourceforge.net/Docs/PyQt5/class_reference.html

`Ctrl - F` 查找 `QPushButton`
https://doc.qt.io/qt-5/qpushbutton.html

得到三种 `QPushButton` 的构造方法：

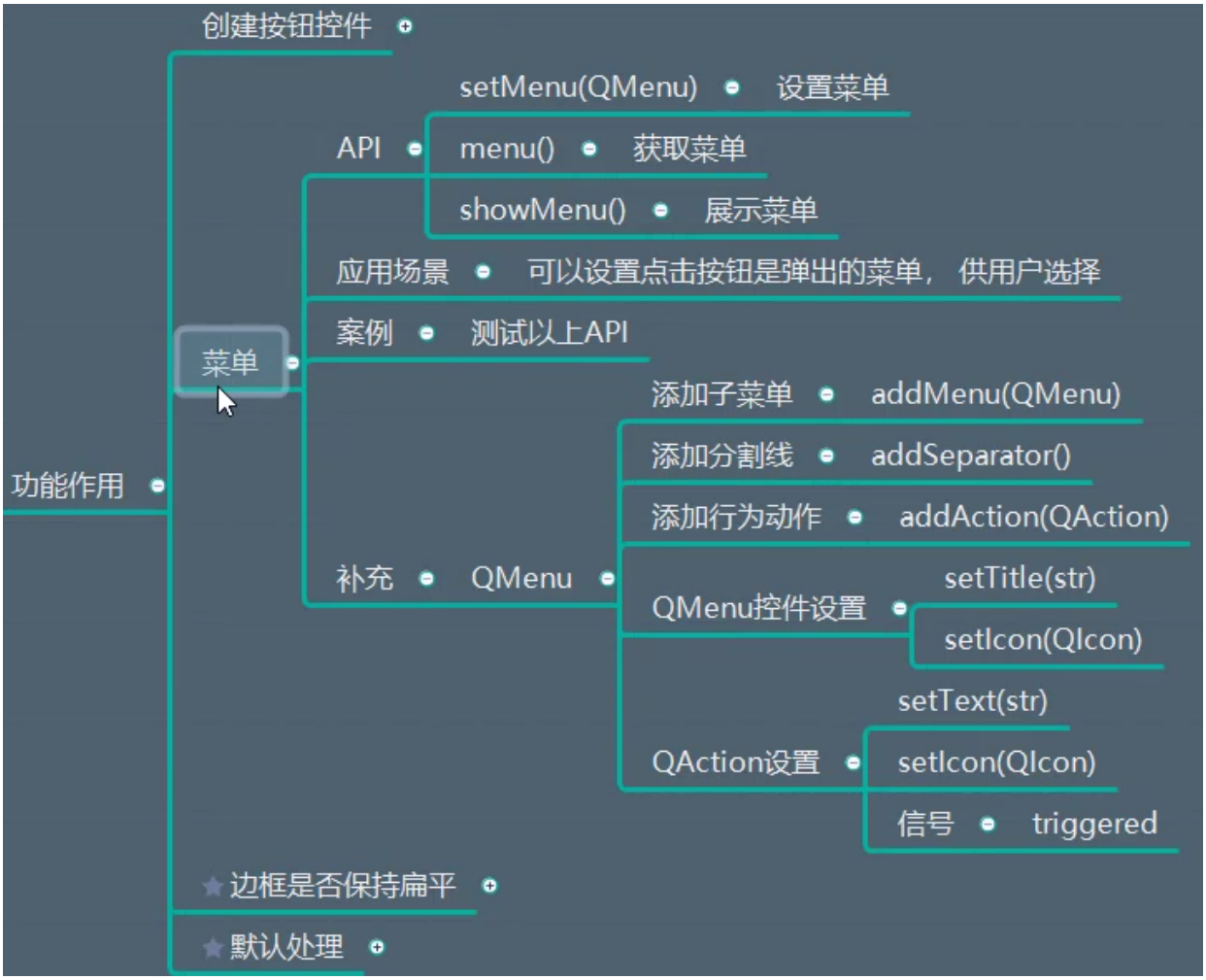
- `QPushButton(parent)` 传入父对象，也可以不写（相当于又多了一种）
- `QPushButton(text, parent)` 传入文本，父对象
- `QPushButton(icon, text, parent)` 传入图标、文本、父对象

**注意**
以后自定义控件，也要多提供构造函数，让它可以快捷地设置一些内容

```
# 0. 导入需要的包和模块
import sys
from PyQt5.Qt import *

# 1. 创建一个应用程序对象
app = QApplication(sys.argv)

# 2. 控件的操作
# 2.1 创建控件
window = QWidget()
# 2.2 设置控件
window.setWindowTitle('构造函数')
window.resize(500, 500)

# 不设置父控件的时候，就是顶层窗口，默认不显示
# 需要使用 show 或者 setVisible(True) 才行
# btn = QPushButton()
# btn.show()
# btn.setVisible(True)

# 设置父控件
# btn = QPushButton(window)
# 或者:
# btn = QPushButton()
# btn.setParent(window)
# 设置文本
```

```
    # btn.setText('按钮')

    # 同时设置文本和父控件
    # btn = QPushButton('按钮', window)
    # 设置图标
    # btn.setIcon(QIcon('img/Python.png'))

    # 同时设置图标，文本和父控件
    btn = QPushButton(QIcon('img/Python.png'), '按钮', window)

    # 2.3 展示控件
    window.show()

    # 3. 应用程序的执行，进入到消息循环
    sys.exit(app.exec_())
```

# 090. Python-GUI编程-PyQt5-QPushButton-菜单设置



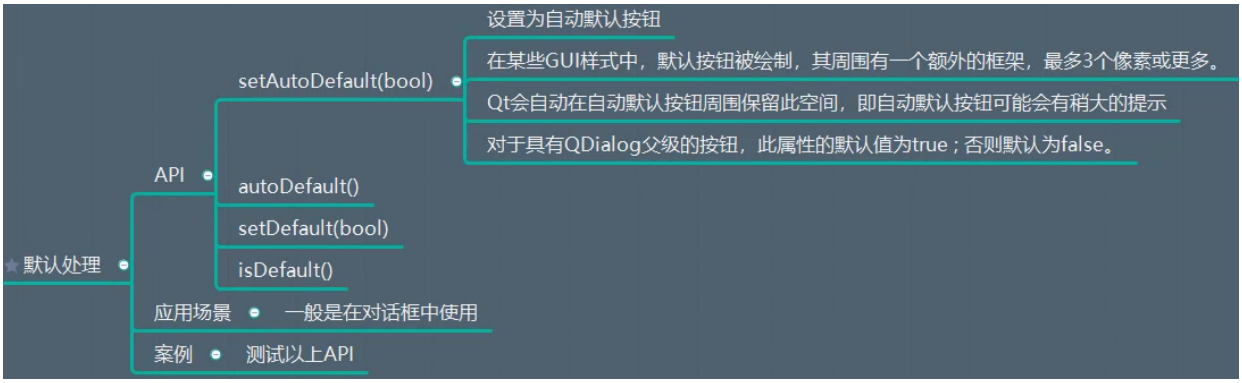
```python
# 0. 导入需要的包和模块
import sys
from PyQt5.Qt import *

# 1. 创建一个应用程序对象
app = QApplication(sys.argv)

# 2. 控件的操作
# 2.1 创建控件
window = QWidget()
# 2.2 设置控件
window.setWindowTitle('菜单的设置')
window.resize(500, 500)

# 创建按钮
btn = QPushButton(QIcon('img/Python.png'), '菜单', window)

# 创建菜单
menu = QMenu()

# 子菜单：最近打开
open_recent_menu = QMenu()
open_recent_menu.setTitle('最近打开')
# 也可以设置图标：setIcon
# 一次性设置：QMenu('标题', 父控件)
# 添加到父菜单，它不会自动添加
# open_recent_menu = QMenu(父控件) 这样不可以

# 子菜单中的行为
file_action = QAction('Python-GUI编程-PyQt5')
# 添加行为
open_recent_menu.addAction(file_action)

# 行为动作（命令）：新建，打开，分割线，退出
# 新建
new_action = QAction()
new_action.setText('新建')
new_action.setIcon(QIcon('img/新建.png'))
new_action.triggered.connect(lambda: print('新建文件'))

# 打开
# QAction的四种构造函数
# QAction()
# QAction(父对象)
# QAction(文本, 父对象)
# QAction(图标, 文本, 父对象)
# 注意：父对象写了，还是要使用 addAction!
open_action = QAction(QIcon('img/打开.png'), '打开', menu)
open_action.triggered.connect(lambda: print('打开文件'))

# 退出
```

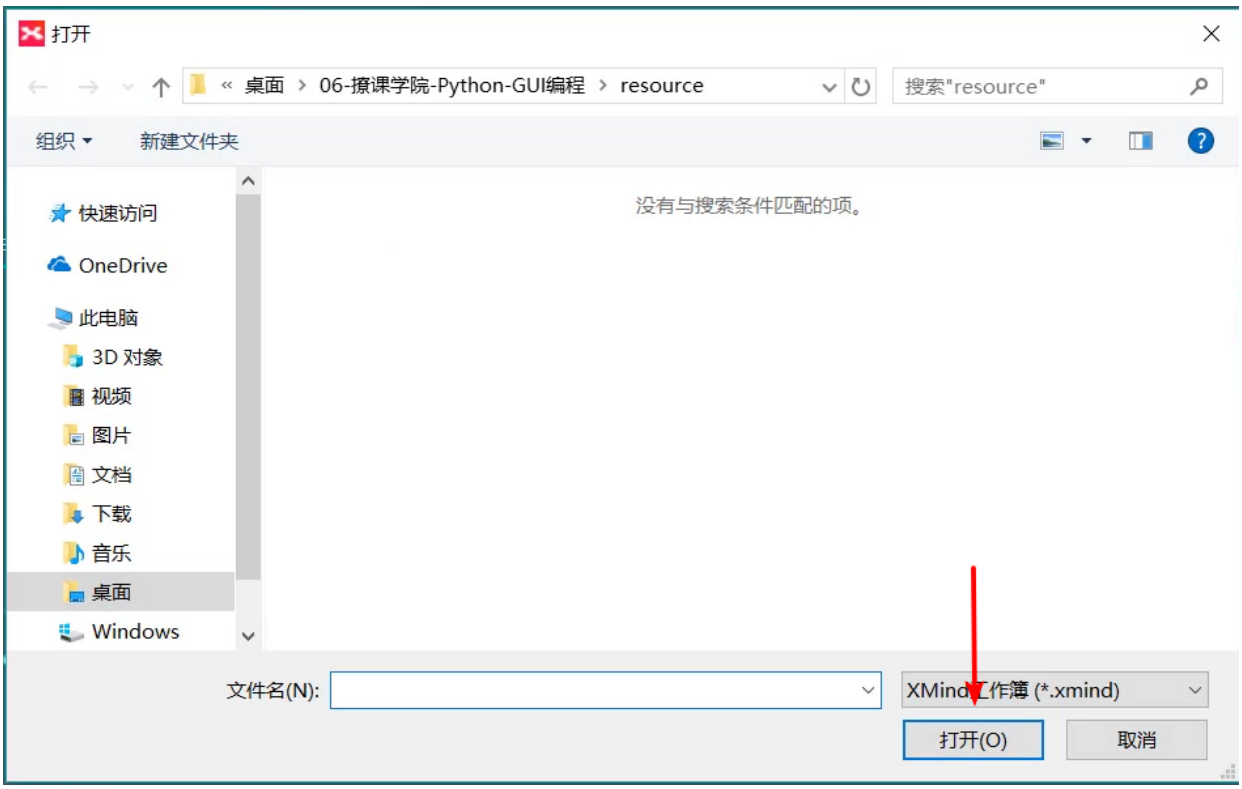
```
    exit_action = QAction(QIcon('img/退出.png'), '退出', menu)
    exit_action.triggered.connect(lambda: print('退出程序'))

    # 添加动作
    menu.addAction(new_action)
    menu.addAction(open_action)
    # 添加子菜单
    menu.addMenu(open_recent_menu)
    # 添加分割线
    menu.addSeparator()
    menu.addAction(exit_action)

    # 给按钮设置菜单
    # 设置菜单之后，按钮右边会出现下拉菜单的箭头
    btn.setMenu(menu)

    # 在这里展示菜单的话会跑到左上角
    # 因为整个窗口还没有展示，都不知道按钮被放在那里
    # 而且 QMenu 继承于 QWidget，自己是可以独立存在的
    # 所以展示菜单需要放到 window.show() 的后面
    # btn.showMenu()

    # 获取菜单，是一个 QMenu 对象
    print(btn.menu())

    # 2.3 展示控件
    window.show()
    # 展示菜单
    btn.showMenu()

    # 3. 应用程序的执行，进入到消息循环
    sys.exit(app.exec_())
```

# 091. Python-GUI编程-PyQt5-QPushButton-扁平化

```python
# 0. 导入需要的包和模块
import sys
from PyQt5.Qt import *

# 1. 创建一个应用程序对象
app = QApplication(sys.argv)

# 2. 控件的操作
# 2.1 创建控件
window = QWidget()
# 2.2 设置控件
window.setWindowTitle('扁平化')
window.resize(500, 500)

button = QPushButton(window)
button.setText('按钮')
button.move(200, 200)
print('当前按钮是否扁平：', button.isFlat())
# 设置为扁平化
button.setFlat(True)
print('当前按钮是否扁平：', button.isFlat())
# 不点击按钮的话，设置背景颜色会没有
button.setStyleSheet('background-color: red')

# 2.3 展示控件
window.show()

# 3. 应用程序的执行，进入到消息循环
sys.exit(app.exec_())
```

# 092. Python-GUI编程-PyQt5-QPushButton-默认处理



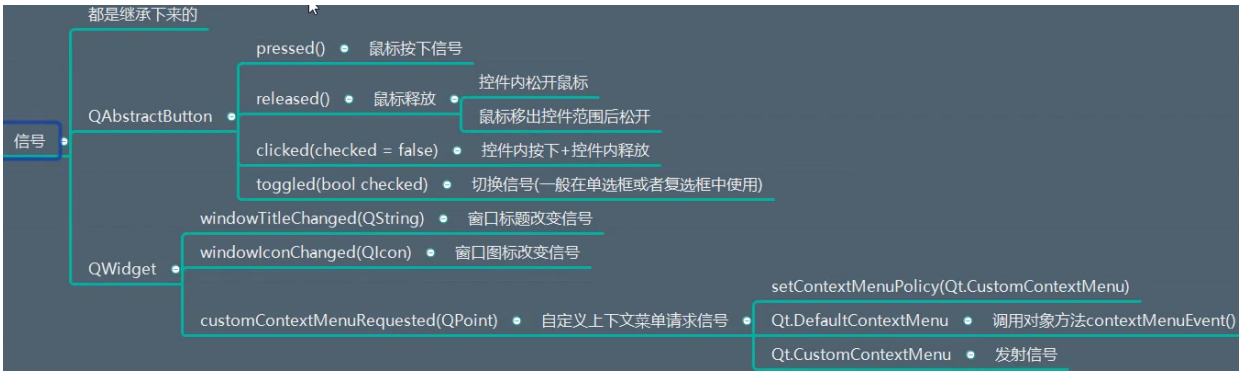



打开按钮就是默认选项，输入回车就会自动点击默认按钮。

```python
# 0. 导入需要的包和模块
import sys
from PyQt5.Qt import *

# 1. 创建一个应用程序对象
app = QApplication(sys.argv)

# 2. 控件的操作
# 2.1 创建控件
window = QWidget()
# 2.2 设置控件
window.setWindowTitle('默认按钮')
window.resize(500, 500)

btn1 = QPushButton(window)
btn1.setText('按钮1')
btn1.move(200, 200)
btn2 = QPushButton(window)
btn2.setText('按钮2')
btn2.move(200, 250)

# 设置为自动默认，点击之后就变成了默认状态
btn2.setAutoDefault(True)
# 查看自动默认状态
print('按钮1是否为自动默认：', btn1.autoDefault())
print('按钮2是否为自动默认：', btn2.autoDefault())
# 注意：自动默认必须要用户点击才会变成默认状态

# 设置为默认状态
btn1.setDefault(True)
```

```
# 2.3 展示控件
window.show()

# 3. 应用程序的执行，进入到消息循环
sys.exit(app.exec_())
```

# 093. Python-GUI编程-PyQt5-QPushButton-右键菜单




### 默认菜单

```
# 0. 导入需要的包和模块
import sys
from PyQt5.Qt import *


class Window(QWidget):
    def contextMenuEvent(self, evt):
        print('默认上下文菜单调用此方法')

        ##################################
        # lk_078_QPushButton_菜单 复制开始
        # 创建菜单
        menu = QMenu(self)

        # 子菜单：最近打开
        open_recent_menu = QMenu()
        open_recent_menu.setTitle('最近打开')
        # 也可以设置图标：setIcon
        # 一次性设置：QMenu('标题', 父控件)
        # 添加到父菜单，它不会自动添加
        # open_recent_menu = QMenu(父控件) 这样不可以

        # 子菜单中的行为
```

```
            file_action = QAction('Python-GUI编程-PyQt5')
            # 添加行为
            open_recent_menu.addAction(file_action)

            # 行为动作（命令）：新建，打开，分割线，退出
            # 新建
            new_action = QAction()
            new_action.setText('新建')
            new_action.setIcon(QIcon('img/新建.png'))
            new_action.triggered.connect(lambda: print('新建文件'))

            # 打开
            # QAction的四种构造函数
            # QAction()
            # QAction(父对象)
            # QAction(文本, 父对象)
            # QAction(图标, 文本, 父对象)
            # 注意：父对象写了，还是要使用 addAction!
            open_action = QAction(QIcon('img/打开.png'), '打开', menu)
            open_action.triggered.connect(lambda: print('打开文件'))

            # 退出
            exit_action = QAction(QIcon('img/退出.png'), '退出', menu)
            exit_action.triggered.connect(lambda: print('退出程序'))

            # 添加动作
            menu.addAction(new_action)
            menu.addAction(open_action)
            # 添加子菜单
            menu.addMenu(open_recent_menu)
            # 添加分割线
            menu.addSeparator()
            menu.addAction(exit_action)
            # lk_078_QPushButton_菜单.py 复制结束
            ##################################

            # exec_ 方法可以展示菜单
            # 要传入一个 QPoint 参数（是相对于整个桌面全局位置）
            menu.exec_(evt.globalPos())
            # 注意没有 localPos() 这个方法！只有 pos()
            # pos() 就是局部位置


# 1. 创建一个应用程序对象
app = QApplication(sys.argv)

# 2. 控件的操作
# 2.1 创建控件
window = Window()
# 2.2 设置控件
window.setWindowTitle('右键菜单1')
window.resize(500, 500)
```

```
# 控件默认的上下文菜单策略是： Qt.DefaultContextMenu
# 会调用对象的 contextMenuEvent()

# 2.3 展示控件
window.show()

# 3. 应用程序的执行，进入到消息循环
sys.exit(app.exec_())
```

### 自定义菜单

```
# 0. 导入需要的包和模块
import sys
from PyQt5.Qt import *

# 1. 创建一个应用程序对象
app = QApplication(sys.argv)

# 2. 控件的操作
# 2.1 创建控件
window = QWidget()
# 2.2 设置控件
window.setWindowTitle('右键菜单2')
window.resize(500, 500)


def show_menu(point):
    # 注意这个点为局部坐标!
    print('自定义上下文菜单，QPoint为：', point)

    ##################################
    # lk_078_QPushButton_菜单 复制开始
    # 创建菜单
    menu = QMenu(window)

    # 子菜单：最近打开
    open_recent_menu = QMenu()
    open_recent_menu.setTitle('最近打开')
    # 也可以设置图标：setIcon
    # 一次性设置：QMenu('标题', 父控件)
    # 添加到父菜单，它不会自动添加
    # open_recent_menu = QMenu(父控件) 这样不可以

    # 子菜单中的行为
    file_action = QAction('Python-GUI编程-PyQt5')
    # 添加行为
```

```
    open_recent_menu.addAction(file_action)

    # 行为动作（命令）：新建，打开，分割线，退出
    # 新建
    new_action = QAction()
    new_action.setText('新建')
    new_action.setIcon(QIcon('img/新建.png'))
    new_action.triggered.connect(lambda: print('新建文件'))

    # 打开
    # QAction的四种构造函数
    # QAction()
    # QAction(父对象)
    # QAction(文本, 父对象)
    # QAction(图标, 文本, 父对象)
    # 注意：父对象写了，还是要使用 addAction!
    open_action = QAction(QIcon('img/打开.png'), '打开', menu)
    open_action.triggered.connect(lambda: print('打开文件'))

    # 退出
    exit_action = QAction(QIcon('img/退出.png'), '退出', menu)
    exit_action.triggered.connect(lambda: print('退出程序'))

    # 添加动作
    menu.addAction(new_action)
    menu.addAction(open_action)
    # 添加子菜单
    menu.addMenu(open_recent_menu)
    # 添加分割线
    menu.addSeparator()
    menu.addAction(exit_action)
    # lk_078_QPushButton_菜单.py 复制结束
    ##################################

    # 用 exec_(全局点坐标) 展示菜单
    # 但是 customContextMenuRequested 给的是局部点坐标
    # 所以需要把局部点映射到全局 mapToGlobal()
    dest_point = window.mapToGlobal(point)
    menu.exec_(dest_point)


# 设置自定义上下文菜单
window.setContextMenuPolicy(Qt.CustomContextMenu)
# 这样会发射信号：自定义上下文菜单请求信号，customContexMenuRequested
# 这个信号发射的时候会携带一个参数：QPoint
window.customContextMenuRequested.connect(show_menu)

# 2.3 展示控件
window.show()

# 3. 应用程序的执行，进入到消息循环
sys.exit(app.exec_())
```

完成于 `201810241013`