day04

综合案例-机器人管理系统

系统需求

- 程序启动，显示名片管理系统欢迎界面，并显示功能菜单

```
**************************************************
欢迎使用[机器人任务管理系统]V1.0

1.新建任务
2.显示所有任务
3.查询任务

0.退出系统
**************************************************
```

系统需求

- 用户用数字选择不同的功能
- 根据功能选择，执行不同的功能
- 任务系统需要记录任务id,任务名,任务类型
- 如果查询到指定的任务，用户可以选择 修改 或者 删除 任务

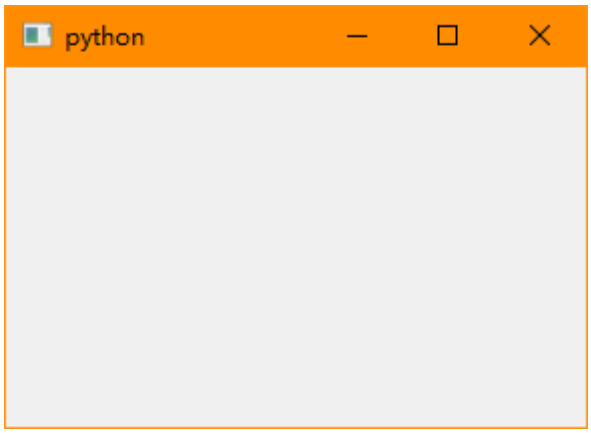
```
"""---------------------- 综合管理系统 ----------------------"""
Task = [["任务id", "任务名", "任务类型"]]

# 程序启动，显示名片管理系统欢迎界面，并显示功能菜单
def show():
    print("*" * 50)
    print("欢迎使用机器人管理系统V1.0")
    print()
    print("1.新建任务")
    print("2.显示所有任务")
    print("3.查询任务")
    print()
    print("0.退出系统")
    print("*" * 50)

# 新建任务
def add_text():
    task_id = input("任务id:")
    task_name = input("任务名:")
    task_types = input("任务类型:")
    # 存入列表中
    Task.append([task_id, task_name, task_types])

# 显示所有任务
def show_text():
    for ele in Task:
        for x in ele:
            print(x, end="  ")
        print()
    input()

# 查询任务
def index_text():
    task_index = input("请用户输入查询的内容:")
    for i in range(0, len(Task)):
        for j in range(0, len(Task[i])):
            if Task[i][j] == task_index:
                for x in Task[i]:
                    print(x, end="  ")
                print()
                flag_index = input("是否修改或者删除任务1表示修改，0表示删除:")
```

```
                if flag_index == '1':
                    Task[i][0] = input("任务id:")
                    Task[i][1] = input("任务id:")
                    Task[i][2] = input("任务id:")
                elif flag_index == '0':
                    del Task[i]
                    return
                else:
                    return
    else:
        print("没有找到")

def Task_function():
    # 显示功能
    show()
    # 选择功能
    a = int(input("请输入要选择的功能:"))
    if a == 1:
        # 新建用户
        add_text()
    elif a == 2:
        # 显示所有任务
        show_text()
    elif a == 3:
        # 查询任务
        index_text()
        pass
    elif a == 0:
        # 退出系统
        return a

while True:
    flag = Task_function()
    if flag == 0:
        break
```

二、PyQt

1.窗口图标

```python
from PyQt5.QtWidgets import QWidget,QApplication
import sys

# 创建PyQt的程序，sys.argv固定写法
app = QApplication(sys.argv)
# 创建窗口
window = QWidget()
# 显示窗口
window.show()
# 暂停
sys.exit(app.exec())
# 结果：
```

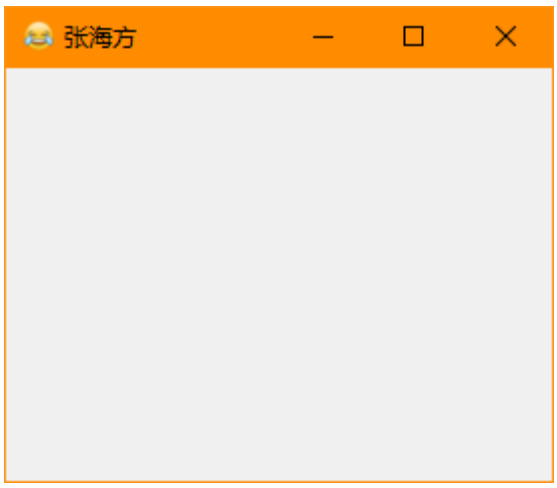


1.2.PyQt5 主要模块

```
QtCore 包含了和兴的非GUI功能。主要和时间、文件与文件夹、各种数据、流、进程与线程一起使用
QtGui:包含窗口系统、事件处理...
QtWidgets:包含创建桌面应用的UI元素
```

1.3.接口函数

```python
# 修改标题
window.setWindowTitle("字符串")
# 修改图标
from PyQt5.QtGui import QIcon
# 创建图标
icon = QIcon("图片名")
# 修改图标
window.setWindowIcon(icon)
# 修改窗口大小
window.resize(400, 300)
```



2.文本控件QLabel

```
1  可以显示不可编辑的文本或者图片，也可以放置GIF动画，首先要先导入QLabel控件在
   PyQt5.QtWidgets中
```

2.1.接口函数

1. 文本的显示

```
1  from PyQt5.QtWidgets import QLabel
2  # 创建文本
3  label = QLabel("想要展示的文字直接写进来")
4  # 设置字体和大小
5  font = QFont("宋体",50)
6  label.setFont(font)
7  # label显示在窗口上,设置父窗体
8  label.setParent(widget)
9  # 结果:
```

2. 图片的显示

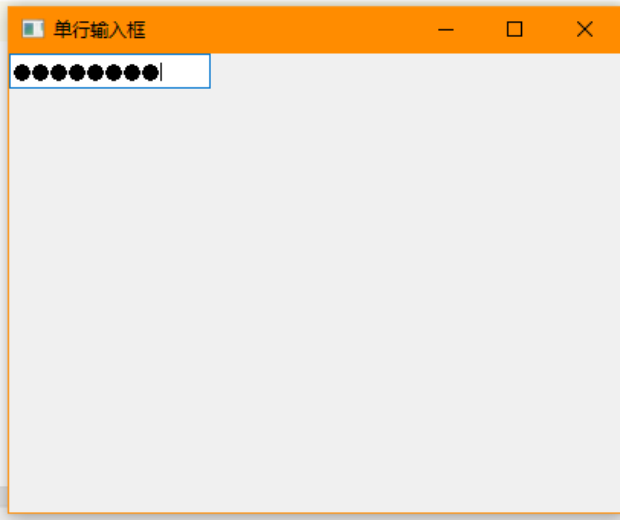
```
# 创建QLabel
label = QLabel()
pixMap = QPixmap("img_0529.jpg")
label.setPixmap(pixMap)
label.setParent(widget)
# 结果：
```

3.单行输入框和多行输入框

3.1.单行输入

```
# 导入QLineEdit
# 创建控件
edit = QLineEdit()
# 设置模式
edit.setEchoMode(QLineEdit.Password)
# 创建父窗体
edit.setParent(widget)
```

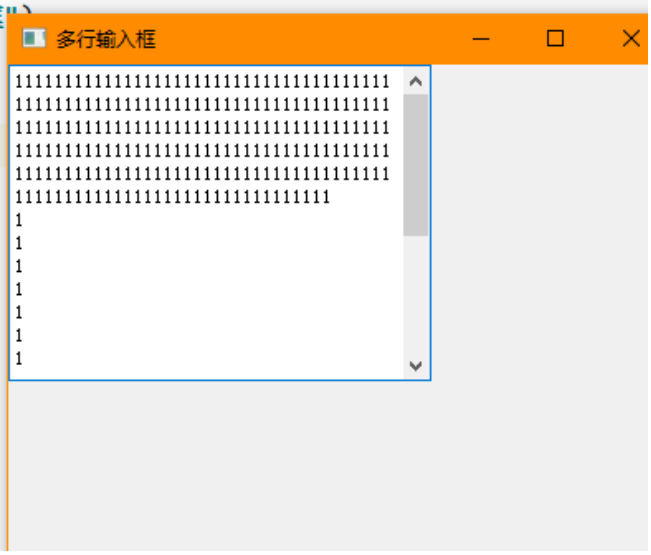
```
# 设置窗口标题
widget.setWindowTitle("单行输入框")
edit = QLineEdit()
# 设置模式
edit.setEchoMode(QLineEdit.Password)
edit.setParent(widget)
# 显示这个窗口
widget.show()

# 系统安全退出 固定写法
sys.exit(app.exec())
```



3.2.多行输入框

```
edit = QTextEdit()
edit.setParent(widget)
# 结果：
```

```
dget.setWindowTitle("多行输入框")

it = QTextEdit()
it.setParent(widget)

显示这个窗口
dget.show()

系统安全退出 固定写法
s.exit(app.exec())
```

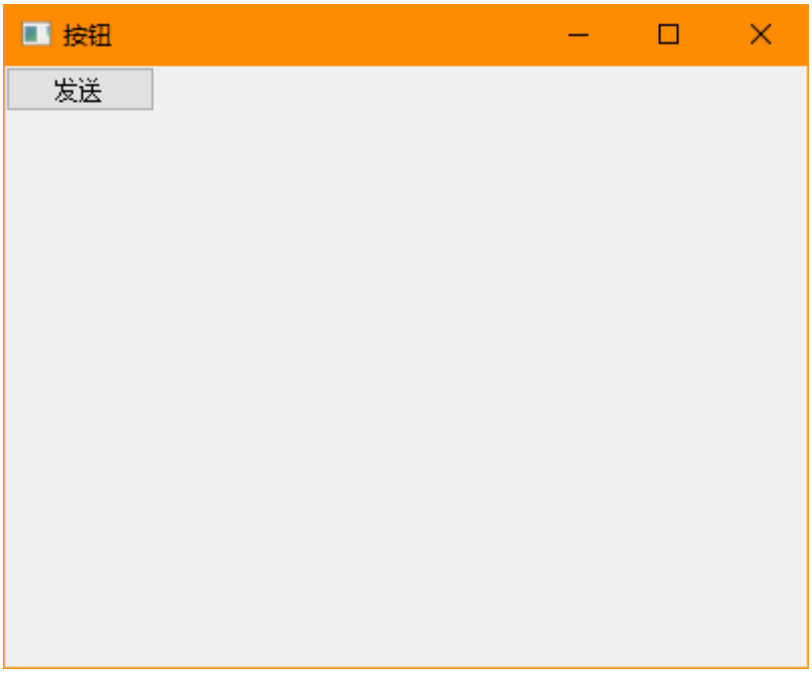


4.按钮

4.1.QPushButton

接口函数

```
# 导入QPushButton即可
# 设置窗口标题，文字直接在后面加
widget.setWindowTitle("按钮")
# 创建按钮
btn = QPushButton("发送")
btn.setParent(widget)
```

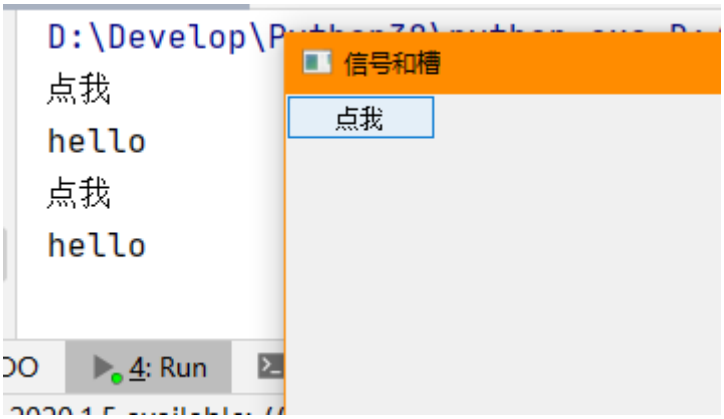


5.信号和槽

```
# 定义槽函数
def func():
    print("hello")
# 定义按键
btn = QPushButton("点我")
btn.setParent(widget)
# 连接信号和槽,func是槽函数
btn.clicked.connect(func)
```

5.1.获取发布者

```
# 通过sender()获取
def func():
    # 获取发布者
    sender = widget.sender()
    print(sender.text())
    print("hello")
# 定义按键
btn = QPushButton("点我")

btn.text()
btn.setParent(widget)
# 连接信号和槽,func是槽函数
btn.clicked.connect(func)
```



6.布局方式：设置排序规则

6.1.布局方式

```
1  水平布局、竖直布局、绝对布局、网格布局、表单布局
```

6.2.框布局

```
1  水平布局（QHBoxLayout）和竖直布局（QVBoxLayout）
```

```
1  # 创建布局
2  layout = QHBoxLayout()
3  widget.setLayout(layout)
4  # 创建按钮
5  btn1 = QPushButton("1")
6  btn2 = QPushButton("2")
7  btn3 = QPushButton("3")
8  btn4 = QPushButton("4")
9  # 将按键添加到布局中
10 layout.addWidget(btn1)
11 layout.addWidget(btn2)
12 layout.addWidget(btn3)
13 layout.addWidget(btn4)
14 # 展示布局
15 layout.setParent(widget)
16 # 水平和竖直一样的代码
```

```
# 创建布局
layout = QHBoxLayout()
widget.setLayout(layout)
# 创建按钮
btn1 = QPushButton("1")
btn2 = QPushButton("2")
btn3 = QPushButton("3")
btn4 = QPushButton("4")
# 将按键添加到布局中
layout.addWidget(btn1)
layout.addWidget(btn2)
layout.addWidget(btn3)
layout.addWidget(btn4)
# 展示布局
layout.setParent(widget)
```