《Python程序设计》

Python编程实践

串口通讯

刘潇
机械科学与工程学院

2023年11月6日

2023秋季

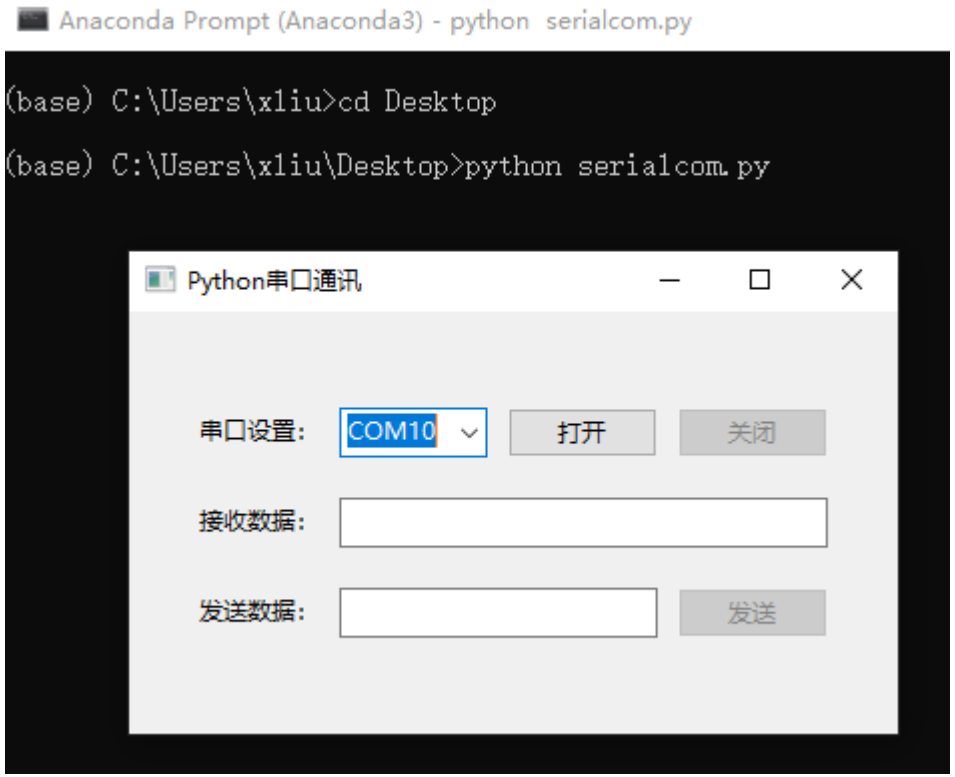
要实现什么功能?

serialcom.py

打开/关闭串口

显示串口接收的数据

向串口发送数据



串口通讯窗口

Python代码170行

本节要点

- 了解Python编程的基本思路
- 了解Python串口通讯模块（pySerial）
- 灵活运用python的内置对象（数据、函数、类等）

主要内容

1. 程序设计思路

2. 代码实现过程

3. 代码演示

程序设计思路

1个窗口 → 显示串口设置、接收和发送数据

4个函数
- 开始串口通讯函数
- 结束串口通讯函数
- 接收数据函数
- 发送数据函数

串口面板类
- (面板初始化)
- (方法绑定按钮事件)

窗口面板（wxPython）

基于wxWidgets的跨平台GUI工具包，提供丰富的标准控件，如按钮、文本框、下拉列表，对话框等

Home | Gallery | API Docs » wxPython API Documentation

next | modules | index

wxPython

Cross-Platform GUI Library

Search

Go

wxPython API Documentation

Welcome! This is the API reference documentation for the **4.2.1 release** of wxPython Phoenix, built on 06 June 2023.

If you are porting your code from Classic wx

You can download a local copy of this docur
default browser, downloading it first if neede

Note: If you wish to help in the docun

Sections

wx Overview Documents
A collection of overview and how-to docume

wx functions
The index of top-level functions available in

Modules

Modules

wx
The classes which appear in the main wx namespace
核心模块

wx.adv
Less commonly used or more advanced
高级控件包

wx.grid
Widget and supporting classes for disp
表格模块
ular data

wx.dataview
Classes for viewing tabular or hierarchical data

wx.richtext
A generic, ground-up implementation o
富文本处理模块
ple text styles and images.

wx.ribbon
A set of classes for writing a ribbon-bas
Ribbon风格控件包
toolbar, similar to the UI in MS Office and Windows 10.

wx.html
Widget and supporting classes for a generic html renderer

wx.html2
Widget and supporting classes for a native html renderer, with CSS and javascript support
html渲染器模块

wx.aui
Docking/floating window panes, draggable
Aui风格控件包

wx.svg
Classes to parse and render Scalable Vector Graphics files.

wx.lib
Our pure-Python library of widgets
纯Python控件库

wx.glcanvas
Classes for embedding OpenGL views in a window

wx.stc
Classes for Styled Text Control, a.k.a Scintilla

wx.msw
A few classes available only on Windows

wx.media
MediaCtrl and related classes
多媒体模块

wx.propgrid
PropertyGrid and related classes for editing a grid of name/value pairs.

wx.xrc
Classes for loading widgets
xml界面布局

wx.xml
Some simple XML classes for use with XRC

wx.py
The py package, containing
Py包（交互式shell）

wx.tools
Some useful tools and utilities for wxPython.

functions
Top-level functions in the wx package.

https://docs.wxpython.org/

窗口面板（wxPython）

Search

From here you can search these documents. Enter your search words into the box below and click "search". Note that the search function will automatically search for all of the words. Pages containing fewer words won't appear in the result list.

wx.statictext | search

Search Results

Search finished, found 54 page(s) matching the search query.

- **wx.StaticText** (Python class, in **wx.StaticText**)
- **wx.FileDialogCustomize.AddStaticText** (Python method, in **wx.FileDialogCustomize**)
- **wx.FileD**
- **wx.lib.a**
- **wx.lib.s**
- **wx.FileD**
- **wx.FileD**
- **wx.FileD**
- **wx.FileD**
- **wx.lib.a**

wx.StaticText

A static text control displays one or more lines of read-only text.

wx.StaticText supports the three classic text alignments, label ellipsization i.e. replacing parts of the text with the ellipsis ("...") if the label doesn't fit into the provided space and also formatting markup with wx.Control.SetLabelMarkup .

^^

Window Styles

This class supports the following styles:

- wx.ALIGN_LEFT: Align the text to the left.
- wx.ALIGN_RIGHT: Align the text to the right.
- wx.ALIGN_CENTRE_HORIZONTAL: Center the text (horizontally).
- wx.ST_NO_AUTORESIZE: By default, the control will adjust its size to exactly fit to the size of the text when SetLabel is called. If this style flag is given, the control will not change its size (this style is especially useful with controls which also have the ALIGN_RIGHT or the ALIGN_CENTRE_HORIZONTAL style because otherwise they won't make sense any longer after a call to SetLabel).
- wx.ST_ELLIPSIZE_START: If the labeltext width exceeds the control width, replace the beginning of the label with an ellipsis; uses wx.Control.Ellipsize .
- wx.ST_ELLIPSIZE_MIDDLE: If the label text width exceeds the control width, replace the middle of the label with an ellipsis; uses wx.Control.Ellipsize .
- wx.ST_ELLIPSIZE_END: If the label text width exceeds the control width, replace the end of the label with an ellipsis; uses wx.Control.Ellipsize . ^^

See also: wx.StaticBitmap, wx.StaticBox

Class Hierarchy

Inheritance diagram for class **StaticText**:

wx.StaticText 控件

窗口面板（wxPython）

D:\anaconda3\Scripts\wxdemo.exe

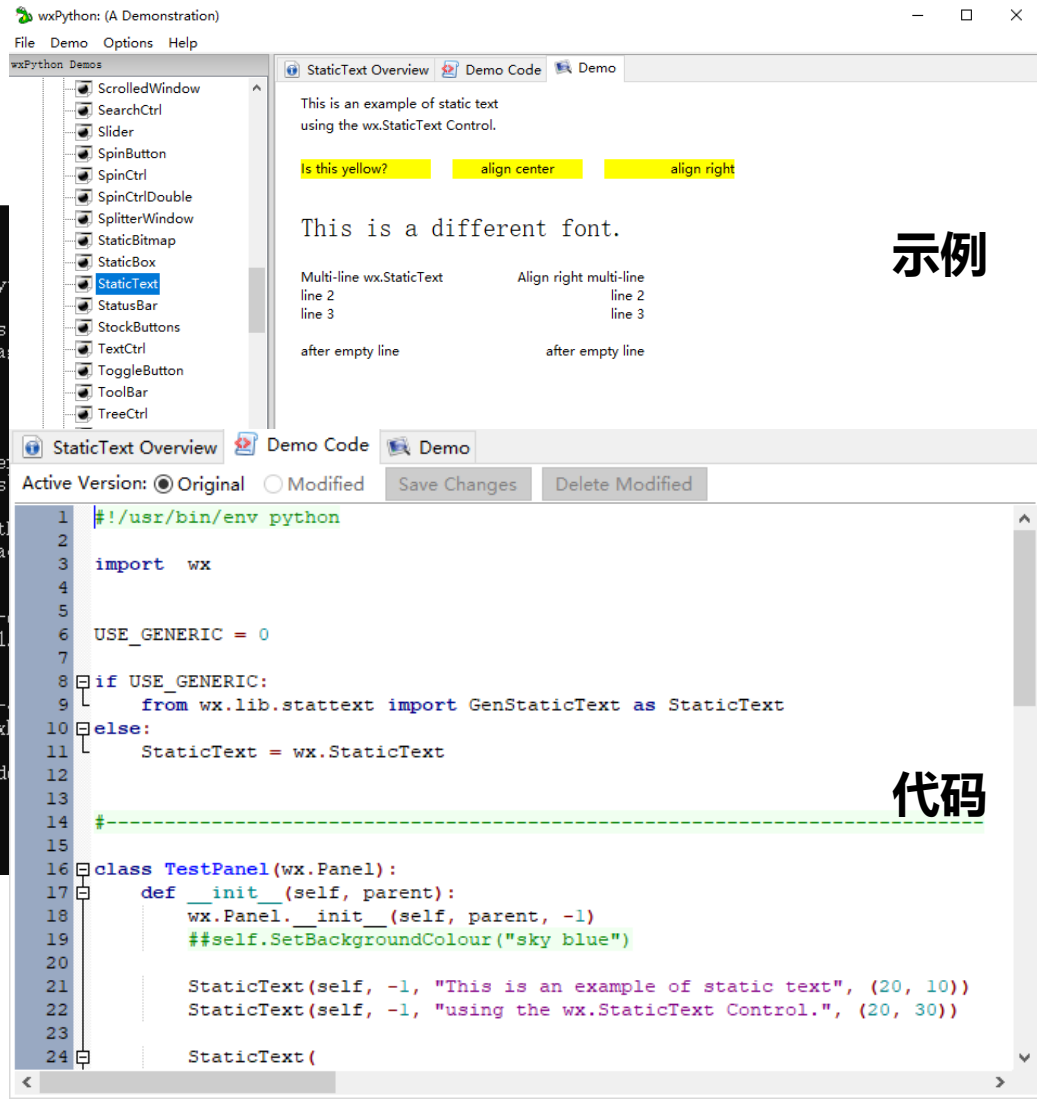
终端运行wxdemo

Anaconda Prompt (Anaconda3)

```
Collecting wxpython
  Downloading wxPython-4.2.1-cp38-cp38-win_amd64.whl (18.1 MB)
     |                                                18.1 MB 2.2 MB/s
Requirement already satisfied: numpy; python_version >= "3.0" and py
conda3\lib\site-packages (from wxpython) (1.18.5)
Requirement already satisfied: six in d:\anaconda3\lib\site-packages
Requirement already satisfied: pillow in d:\anaconda3\lib\site-packa
Installing collected packages: wxpython
Successfully installed wxpython-4.2.1

(base) C:\Users\xliu>wxdemo
sys.version_info(major=3, minor=8, micro=3, releaselevel='final', se
2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] ['D:\\Anaconda3\\Scripts
Launch Demo for wxPython V4.2.1
Looking for wxPython-demo-4.2.1 at C:\Users\xliu\AppData\Local\wxPyt
Looking for cached C:\Users\xliu\AppData\Local\wxPython\wxDocsDemoCa
ar.gz
Trying:
   wget https://extras.wxpython.org/wxPython4/extras/4.2.1/wxPython-
liu\AppData\Local\wxPython\wxDocsDemoCache\4.2.1\wxPython-demo-4.2.1
wget did not work or not installed - trying urllib
Trying to Download via urllib from:
   https://extras.wxpython.org/wxPython4/extras/4.2.1/wxPython-demo-
Unpack C:\Users\xliu\AppData\Local\wxPython\wxDocsDemoCache\4.2.1\wx
Users\xliu\AppData\Local\wxPython
Launching C:\Users\xliu\AppData\Local\wxPython\wxPython-demo-4.2.1\d
Demo starting as PID 7028 - may take a few seconds!
Closing Launcher App!

(base) C:\Users\xliu>
```



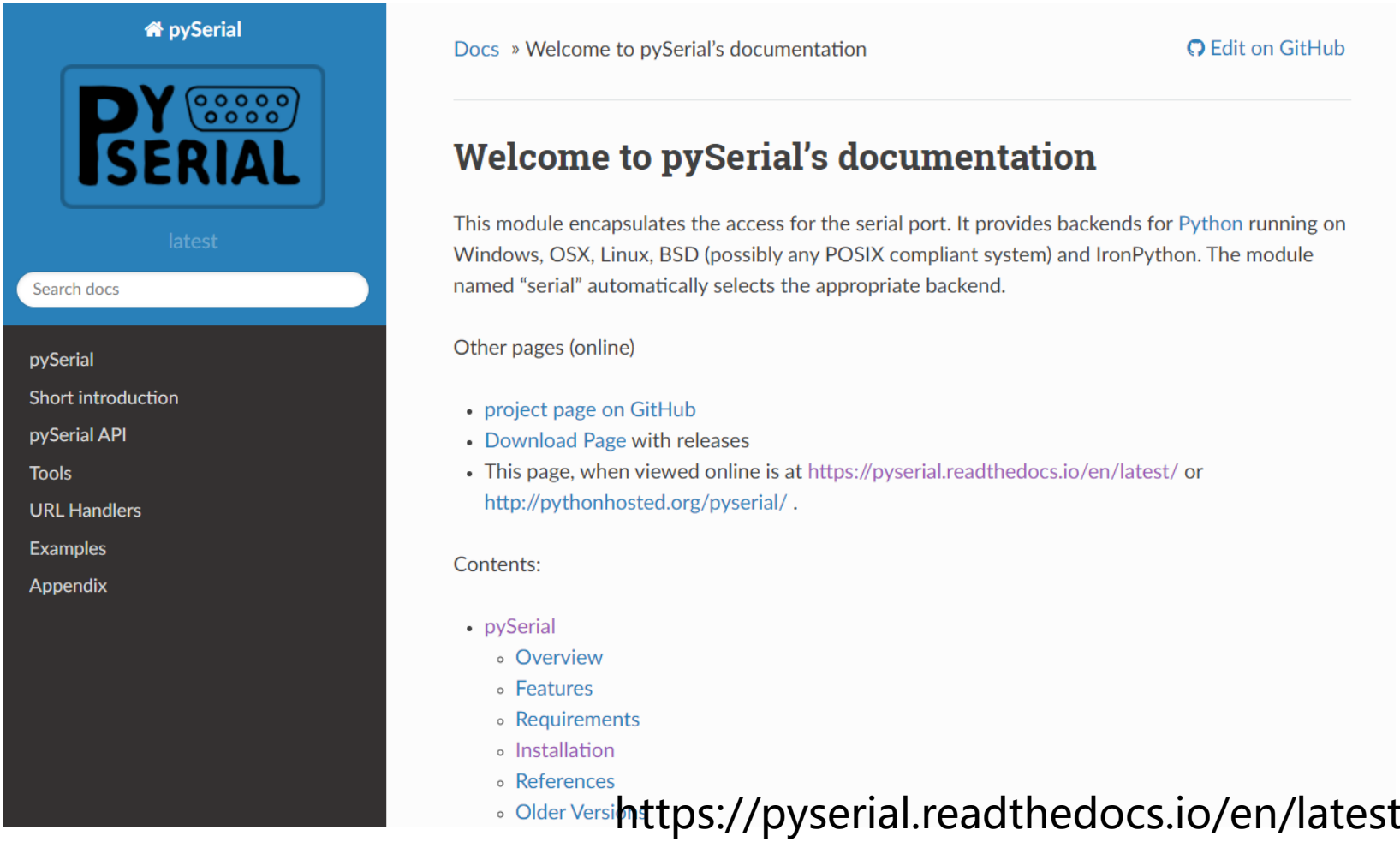
示例

代码

wx.StaticText 控件

串口函数实现（pySerial）

Python的第三方串口通讯库，支持不同平台下对串口设备进行读写操作



https://pyserial.readthedocs.io/en/latest/

串口函数实现（pySerial）

安装

pip install pyserial

Anaconda Prompt (Anaconda3)

```
(base) C:\Users\xliu>pip install pyserial
Collecting pyserial
  Downloading pyserial-3.5-py2.py3-none-any.whl (90 kB)
     |[redacted] 90 kB 980 kB/s
Installing collected packages: pyserial
Successfully installed pyserial-3.5

(base) C:\Users\xliu>_
```

Installed | Channels | Update index... | Search Packages

Name	T	Description	Version
pyrsistent			0.16.0
pyserial		Python serial port access library	3.5
pysocks		A python socks client module. see https://github.com/anorov/pysocks for more information.	1.7.1
pytables		Brings together python, hdf5 and numpy to easily handle large amounts of data.	3.6.1
pytest		Simple and powerful testing with python.	5.4.3
python		General purpose programming language	3.8.3

串口函数实现（pySerial）

pySerial API

https://pyserial.readthedocs.io/en/latest/pyserial_api.html

Classes

Native ports

class `serial.Serial`

`__init__`(*port=None, baudrate=9600, bytesize=EIGHTBITS, parity=PARITY_NONE, stopbits=STOPBITS_ONE, timeout=None, xonxoff=False, rtscts=False, write_timeout=None, dsrdtr=False, inter_byte_timeout=None, exclusive=None*)

Parameters:

- **port** – Device name or `None`. 设备名称或None
- **baudrate** (*int*) – Baud rate such as 9600 or 115200 etc. 波特率（整型数据）
- **bytesize** – Number of data bits. Possible values: `FIVEBITS`, `SIXBITS`, `SEVENBITS`, `EIGHTBITS` 数据位数
- **parity** – Enable parity checking. Possible values: `PARITY_NONE`, `PARITY_EVEN`, `PARITY_ODD`, `PARITY_MARK`, `PARITY_SPACE` 校验位
- **stopbits** – Number of stop bits. Possible values: `STOPBITS_ONE`, `STOPBITS_ONE_POINT_FIVE`, `STOPBITS_TWO` 停止位
- **timeout** (*float*) – Set a read timeout value in seconds. 设置读取超时值，用于控制read()行为
- **xonxoff** (*bool*) – Enable software flow control.

timeout=None	永远等待，直到收到请求的字节数
timeout=0	非阻塞模式，任何情况下，立即返回
timeout=x	在请求的字节数可用时立即返回，否则等到超时，返回之前收到的所有字节

串口函数实现（pySerial）

pySerial基本运用

```
# 导入pyserial模块
import serial

# 创建serial对象
serial_obj = serial.Serial("COM10", 19200)
```

COM10为串口号，19200为波特率

```
# 发送数据
serial_obj.write(b"Hello world")
```

向串口发送数据（字节字符串，以二进制字节序列的形式存储字符串）

```
# 接收数据
data = serial_obj.read(5)
print(data)
```

从串口读取5个字节数据

```
#关闭串口连接
serial_obj.close()
```

串口函数实现（pySerial）

pySerial基本运用

```
# 导入pyserial模块
import serial

# 创建serial对象
serial_obj = serial.Serial("COM10", 19200)
```

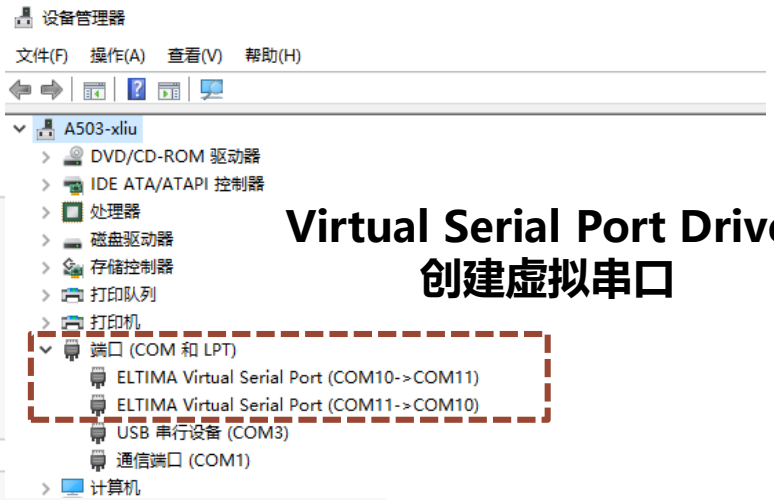
```
# 发送数据
serial_obj.write(b"Hello world")
```

11　**11个字节**

```
# 接收数据
data = serial_obj.read(5)
print(data)
```

b'12345'

```
#关闭串口连接
serial_obj.close()
```



Virtual Serial Port Driver
创建虚拟串口

COM11

[12:43:47.738]收←◆48 65 6C 6C 6F 20 77 6F 72 6C 64

Bin (二进制)	Oct (八进制)	Dec (十进制)	Hex (十六进制)	缩写/字符	解释
0100 1000	0110	72	0x48	H	大写字母H
0110 0101	0145	101	0x65	e	小写字母e
0110 1100	0154	108	0x6C	l	小写字母l
0110 1100	0154	108	0x6C	l	小写字母l
0110 1111	0157	111	0x6F	o	小写字母o
0010 0000	040	32	0x20	(space)	空格
0111 0111	0167	119	0x77	w	小写字母w
0110 1111	0157	111	0x6F	o	小写字母o
0111 0010	0162	114	0x72	r	小写字母r
0110 1100	0154	108	0x6C	l	小写字母l
0110 0100	0144	100	0x64	d	小写字母d

https://baike.baidu.com/item/ASCII/309296?fr=ge_ala

代码实现过程

1个窗口

```
import wx # 加载wxPython模块
import serial, serial.tools.list_ports # 加载pySerial模块

serialcomconfig = ["COM10", "19200", "8", "None", "1"] # 串口参数初始化

# 创建串口通讯面板类
class SerialcomPanel(wx.Panel):
    # 面板界面初始化
    def __init__(self, parent=None):
        # 面板类初始化
        wx.Panel.__init__(self, parent)

# 创建一个应用程序对象
app = wx.App()

# 创建一个顶层框架对象（全局坐标系）
frame = wx.Frame(None, title="Python串口通讯")

# 创建一个局部面板对象（局部坐标系）
panel = SerialcomPanel(frame)

# 显示框架
frame.Center()
frame.Show()

# 运行程序
app.MainLoop()
```

Panel布局

```
# 面板水平布局器
sizer = wx.BoxSizer(wx.HORIZONTAL)

# 面板水平布局器左边空字符串填充
text_left = wx.StaticText(self, wx.ID_ANY, "")
sizer.Add(text_left, proportion=0, flag=wx.ALIGN_CENTRE|wx.ALL, border=10)

# 中间垂直布局器
sizer_center = wx.BoxSizer(wx.VERTICAL)

# 串口设置（第一行水平布局器(静态文字-下拉选择框-打开串口按钮-关闭串口按钮)）
settings_box = wx.BoxSizer(wx.HORIZONTAL)
self.statictext1 = wx.StaticText(self, wx.ID_ANY, u"串口设置:", # 静态文字
settings_box.Add(self.statictext1, 0, wx.ALIGN_CENTRE | wx.ALL, 5)

comsetchoices = []
port_list = list(serial.tools.list_ports.comports()) # 获取系统串口列表
if len(port_list) == 0:
    wx.MessageBox("找不到串口!")
else:
    for i in range(0, len(port_list)):
        comsetchoices.append(str(port_list[i])[:5])
self.comset = wx.ComboBox(self, wx.ID_ANY, serialcomconfig[0], wx.DefaultPosi
settings_box.Add(self.comset, 1, wx.ALIGN_CENTRE | wx.ALL, 5)

self.btn_open = wx.Button(self, label="打开") # 打开串口按钮
# self.Bind(wx.EVT_BUTTON, self.Openserialcom, self.
settings_box.Add(self.btn_open, 1, wx.ALIGN_CENTRE | wx.ALL, 5)

self.btn_close = wx.Button(self, label="关闭") # 关闭串口按钮
# self.Bind(wx.EVT_BUTTON, self.Closeserialcom, self.
self.btn_close.Disable()
settings_box.Add(self.btn_close, 1, wx.ALIGN_CENTRE | wx.ALL, 5)

sizer_center.Add(settings_box, 0, wx.EXPAND | wx.ALL, 5)
```

水平布局器

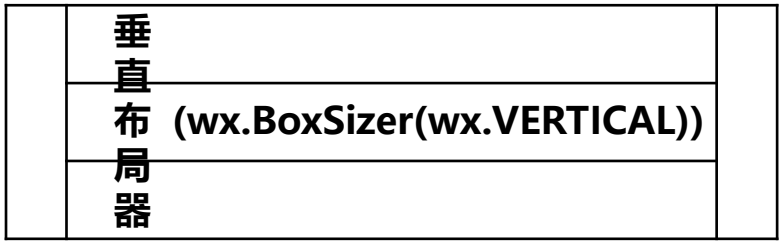
垂直布局器

静态文本

下拉选择框

打开串口按钮

关闭串口按钮



水平布局器
(wx.BoxSizer(wx.HORIZONTAL))

控件定义为Panel类的属性

4个函数-开始串口通讯函数

self.Openserialcom函数绑定点击self.btn_open按钮

```
self.btn_open = wx.Button(self, label="打开") # 打开串口按钮
self.Bind(wx.EVT_BUTTON, self.Openserialcom, self.btn_open)
settings_box.Add(self.btn_open, 1, wx.ALIGN_CENTRE | wx.ALL, 5)
```

```
# 打开串口通讯端口
def Openserialcom(self, event):
    try:
        global serialcomconfig
        port = self.comset.GetValue()
        baudrate = int(serialcomconfig[1])
        bytesize = int(serialcomconfig[2])
        parity = serialcomconfig[3]
        if parity == u"None":
            parity = serial.PARITY_NONE
        elif parity == u"Odd":
            parity = serial.PARITY_ODD
        elif parity == u"Even":
            parity = serial.PARITY_EVEN
        elif parity == u"Mark":
            parity = serial.PARITY_MARK
        elif parity == u"Space":
            parity = serial.PARITY_SPACE
        else:
            wx.MessageBox("校验位设置错误！")
        stopbit = serialcomconfig[4]
        if stopbit == u"1":
            stopbit = serial.STOPBITS_ONE
        elif stopbit == u"1.5":
            stopbit = serial.STOPBITS_ONE_POINT_FIVE
        elif stopbit == u"2":
            stopbit = serial.STOPBITS_TWO
        else:
            wx.MessageBox("停止位设置错误！")
```

导入串口参数

```
        # 创建串口类对象
        self.serialport = serial.Serial()
        self.serialport.port = port
        self.serialport.baudrate = baudrate
        self.serialport.bytesize = bytesize
        self.serialport.parity = parity
        self.serialport.stopbits = stopbit

        # 打开串口通讯
        self.serialport.open()
        self.btn_open.Disable()
        self.btn_close.Enable()
        self.btn_send.Enable()

        # 启动更新接收数据文本框线程
        #self.stopthread_flag = False
        #self.thread_read = Thread(target=self.Receivedata)
        #self.thread_read.setDaemon(True)
        #self.thread_read.start()

    except serial.SerialException as e:
        wx.MessageBox("检查串口设置！")
```

开启串口

异常处理

4个函数-接收数据函数

```
import wx # 加载wxPython模块
import serial, serial.tools.list_ports # 加载pySerial模块
from threading import Thread # 加载超线程模块
import time # 超线程内函数定时运行
from binascii import unhexlify, b2a_hex # 数据转换
```

超线程运行接收数据函数

```
        # 启动更新接收数据文本框线程
        self.stopthread_flag = False
        self.thread_read = Thread(target=self.Receivedata)
        self.thread_read.setDaemon(True)
        self.thread_read.start()

    except serial.SerialException as e:
        wx.MessageBox("检查串口设置！")

# 接收数据函数
def Receivedata(self):
    while self.serialport.isOpen() and not self.stopthread_flag:
        time.sleep(1)
        n = self.serialport.inWaiting()
        if n:
            receive_data = b2a_hex(self.serialport.read(n))
            self.receive_textctrl.SetLabelText(receive_data)
```

启动超线程

二进制转换为十六进制，输出对应的字符串

4个函数-结束串口通讯函数

self.Closeserialcom函数绑定点击self.btn_close按钮

```
self.btn_close = wx.Button(self, label="关闭") # 关闭串口按钮
self.Bind(wx.EVT_BUTTON, self.Closeserialcom, self.btn_close)
self.btn_close.Disable()
settings_box.Add(self.btn_close, 1, wx.ALIGN_CENTRE | wx.ALL, 5)
```

```
# 关闭串口通讯端口
def Closeserialcom(self, event):
    # 结束更新接收数据文本框线程
    self.stopthread_flag = True
    self.thread_read.join()
    # 关闭串口通讯
    self.serialport.close()
    self.btn_open.Enable()
    self.btn_close.Disable()
    self.btn_send.Disable()
```

结束接收数据超线程

关闭串口，恢复按钮状态

4个函数-发送数据函数

self.Senddata函数绑定点击self.btn_send按钮

```
# 发送数据（第三行水平布局器(静态文字-显示文本框-发送按钮)）
senddata_box = wx.BoxSizer(wx.HORIZONTAL)
self.statictext3 = wx.StaticText(self, wx.ID_ANY, u"发送数据：") # 静态文字
senddata_box.Add(self.statictext3, 0, wx.ALIGN_CENTRE | wx.ALL, 5)
self.send_textctrl = wx.TextCtrl(self, wx.ID_ANY) # 显示文本框
senddata_box.Add(self.send_textctrl, 2, wx.ALIGN_CENTRE | wx.ALL, 5)
self.btn_send = wx.Button(self, label="发送") # 发送按钮
self.Bind(wx.EVT_BUTTON, self.Senddata, self.btn_send)
self.btn_send.Disable()
senddata_box.Add(self.btn_send, 1, wx.ALIGN_CENTRE | wx.ALL, 5)
```

```
# 发送数据函数
def Senddata(self, event):
    if self.serialport.isOpen():
        send_data = unhexlify(self.send_textctrl.GetValue())
        self.serialport.write(send_data)
    else:
        wx.MessageBox("串口未连接！")
```

十六进制字符串转换为二进制字节字符串

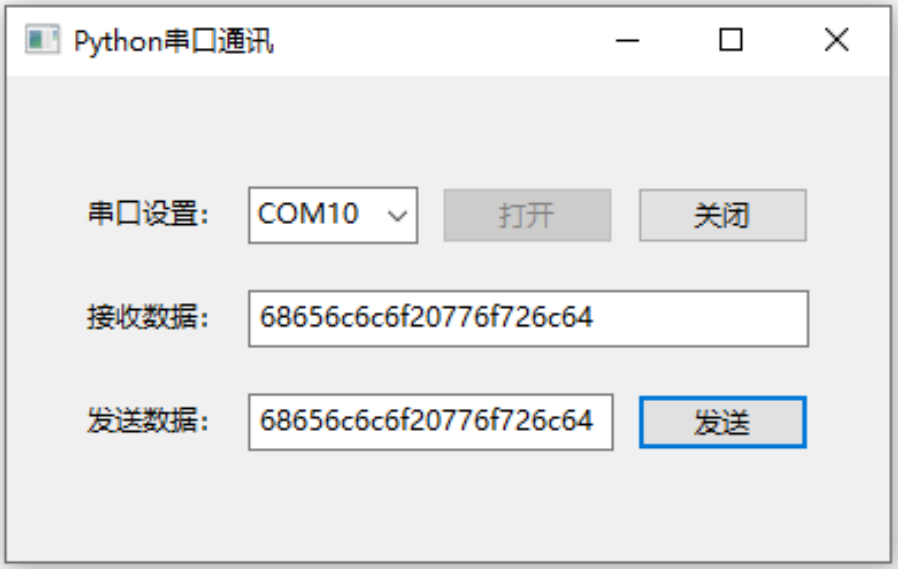
代码演示

通讯端口 串口设置 显示 发送 多字符串 小工具 帮助 联系作者 大虾论坛

```
[15:38:19.961]发→◇hello world□
[15:38:28.876]收←◆hello world
```

b2a_hex() 返回十六进制对应的字符串

unhexlify() 返回二进制字节字符串



清除窗口 打开文件 发送文件 停止 清发送区 最前 English 保存

端口号 COM11 ELTIMA Virtual Seria HEX显示 保存数据 接收数据到文件 HEX发送 定时发送: 1000 ms/次

关闭串口 更多串口设置 加时间戳和分包显示, 超时时间: 20 ms 第 1 字节 至 末尾 加校验 None

RTS DTR 波特率: 19200

hello world

小结

- Python的面向对象编程（属性和方法）
- GUI和串口通讯模块
- 命名空间和函数参数传递
- 程序异常处理

下一节：Python科学计算