《Python程序设计》

Python模块

刘潇
机械科学与工程学院

2023年11月2日

2023秋季

我们学了什么?

数字、字符串、列表 函数、类	~~import~~	第一梯队，直接集成在python中的库和模块
math、 random、 time、re、os …	import	第二梯队，Python自带的标准库和模块 *lib文件夹*
wxpython、 numpy、 scipy …	安装后 import	第三梯队，第三方公司提供的库和模块 *lib\site-packages文件夹*
module1、module2…	编写后 import	第四梯队，自己编写的的库和模块 *PATH环境变量*

什么是模块?

模块是在函数和类的基础上，将一系列相关的代码组织在一起的集合体，在Python中以“.py”结尾的文件就是一个模块

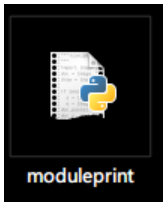
函数 → 代码封装 → **类** → 代码封装 → **模块**

```python
def print1():
    print("hello world!")

print1()
print1()
print1()
```

```
hello world!
hello world!
hello world!
```

```python
class classprint():
    def print1(self):
        print("hello world1!")

    def print2(self):
        print("hello world2!")

obj = classprint()
obj.print1()
obj.print2()
```

```
hello world1!
hello world2!
```



moduleprint.py

```python
import moduleprint

# print(dir(moduleprint))
moduleprint.print1()
obj = moduleprint.classprint()
obj.print1()
```

```
hello world!
hello world1!
```

调用模块实现更高级的代码重复利用

本节要点

- 了解程序的模块封装和文件管理
- 掌握Python中模块的创建和导入规则
- 了解Python中的GUI模块(wxpython)

主要内容

1. 内置模块

2. 第三方模块

3. 创建模块

内置模块

- 通过import调用实现更高级的代码重复利用
- 文件中的一段代码、文件、放文件的文件夹、多个关联的文件夹

/Anaconda3/Lib

queue.py	2021/8/31 3:02	PY 文件
quopri.py	2021/8/31 3:02	PY 文件
random.py	2021/8/31 3:02	PY 文件
re.py	2021/8/31 3:02	PY 文件
reprlib.py	2021/8/31 3:02	PY 文件
rlcompleter.py	2021/8/31 3:02	PY 文件
runpy.py	2021/8/31 3:02	PY 文件
sched.py	2021/8/31 3:02	PY 文件
secrets.py	2021/8/31 3:02	PY 文件
selectors.py	2021/8/31 3:02	PY 文件
shelve.py	2021/8/31 3:02	PY 文件

I:) › anaconda3 › Lib › site-packages › numpy ›

名称	修改日期	类型	大小
__pycache__	2022/4/25 13:04	文件夹	
compat	2022/4/25 12:13	文件夹	
core	2022/4/25 12:13	文件夹	
distutils	2022/4/25 12:13	文件夹	
doc	2022/4/25 12:13	文件夹	
f2py	2022/4/25 12:13	文件夹	
fft	2022/4/25 12:13	文件夹	
lib	2022/4/25 12:13	文件夹	
linalg	2022/4/25 12:13	文件夹	
ma	2022/4/25 12:13	文件夹	
matrixlib	2022/4/25 12:13	文件夹	
polynomial	2022/4/25 12:13	文件夹	
random	2022/4/25 12:13	文件夹	
testing	2022/4/25 12:13	文件夹	
tests	2022/4/25 12:13	文件夹	
typing	2022/4/25 12:13	文件夹	
__config__.py	2022/4/25 12:13	PY 文件	3 KB
__init__.cython-30.pxd	2021/7/14 22:16	PXD 文件	36 KB
__init__.pxd	2021/7/14 22:16	PXD 文件	34 KB
__init__.py	2021/7/14 22:16	PY 文件	16 KB
__init__.pyi	2021/7/14 22:16	PYI 文件	59 KB
_distributor_init.py	2021/7/14 22:16	PY 文件	1 KB
_globals.py	2021/7/14 22:16	PY 文件	3 KB
_pytesttester.py	2021/7/14 22:16	PY 文件	7 KB
char.pyi	2021/7/14 22:16	PYI 文件	1 KB
conftest.py	2021/7/14 22:16	PY 文件	4 KB
ctypeslib.py	2021/7/14 22:16	PY 文件	17 KB
ctypeslib.pyi	2021/7/14 22:16	PYI 文件	1 KB
dual.py	2021/7/14 22:16	PY 文件	3 KB
emath.pyi	2021/7/14 22:16	PYI 文件	1 KB
LICENSE	2021/7/14 22:16	文本文档	2 KB
matlib.py	2021/7/14 22:16	PY 文件	11 KB
py.typed	2021/7/14 22:16	TYPED 文件	0 KB
rec.pyi	2021/7/14 22:16	PYI 文件	1 KB
setup.py	2021/7/14 22:16	PY 文件	1 KB
version.py	2021/7/14 22:16	PY 文件	1 KB

模块、包、库

如何导入模块

使用import语句导入模块，创建模块对象

import module1, [module2[, ... module N]]

导入多个模块

```
import math, random
print(dir())
```

一个文件中执行多次import，一个模块只会导入一次

```
['In', 'Out', '_', '__', '___', '__builtin__', '__builtins__', '__doc__', '__loader__', '
i', '_i1', '_ih', '_ii', '_iii', '_oh', 'exit', 'get_ipython', 'math', 'quit', 'random']
```

from语句从模块中导入一个指定的部分到当前命名空间

from ... import导入

```
from math import pi
print(dir())
print(pi)
```

```
['In', 'Out', '_', '__', '___', '__builtin__', '__builtins__', '__doc__', '__loader__', '__name__', '
i', '_i1', '_i2', '_ih', '_ii', '_iii', '_oh', 'exit', 'get_ipython', 'math', 'pi', 'quit', 'random']
3.141592653589793
```

```
from random import random, choice
print(random())
print(choice("hello world!"))
```

random和choice函数在当前命名空间中可直接调用，**不创建模块对象**

```
0.5425813896827737
!
```

from语句不会把整个模块导入当前命名空间，只会将指定的代码（数据、函数、类等对象）导入到执行这个声明模块的全局命名空间

```
print(dir())
```

```
['In', 'Out', '_', '__', '___', '__builtin__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__',
i', '_i1', '_i2', '_i3', '_i4', '_ih', '_ii', '_iii', '_oh', 'choice', 'exit', 'get_ipython', 'math', 'pi', 'quit', 'random']
```

```
print(random.random())
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_4648/4208411749.py in <module>
----> 1 print(random.random())

AttributeError: 'builtin_function_or_method' object has no attribute 'random'
```

from ... import *会把模块中的所有内容导入到当前命名空间，不建议使用，会消耗内存空间

from ... import ... as ... 导入并重命名

从哪导入（模块路径）

import语句执行后，python解释器对模块位置的搜索顺序是：

当前目录 → 环境变量PYTHONPATH下的目录 → python模块路径目录

存储在sys模块的sys.path变量中

```
import sys, pprint
pprint.pprint(sys.path)
```

```
['C:\\Users\\xliu\\Desktop',
 'D:\\anaconda3\\python39.zip',
 'D:\\anaconda3\\DLLs',
 'D:\\anaconda3\\lib',
 'D:\\anaconda3',
 '',
 'D:\\anaconda3\\lib\\site-packages',
 'D:\\anaconda3\\lib\\site-packages\\locket-0.2.1-py3.9.egg',
 'D:\\anaconda3\\lib\\site-packages\\win32',
 'D:\\anaconda3\\lib\\site-packages\\win32\\lib',
 'D:\\anaconda3\\lib\\site-packages\\Pythonwin',
 'D:\\anaconda3\\lib\\site-packages\\IPython\\extensions',
 'C:\\Users\\xliu\\.ipython']
```

空字符串表示当前工作目录

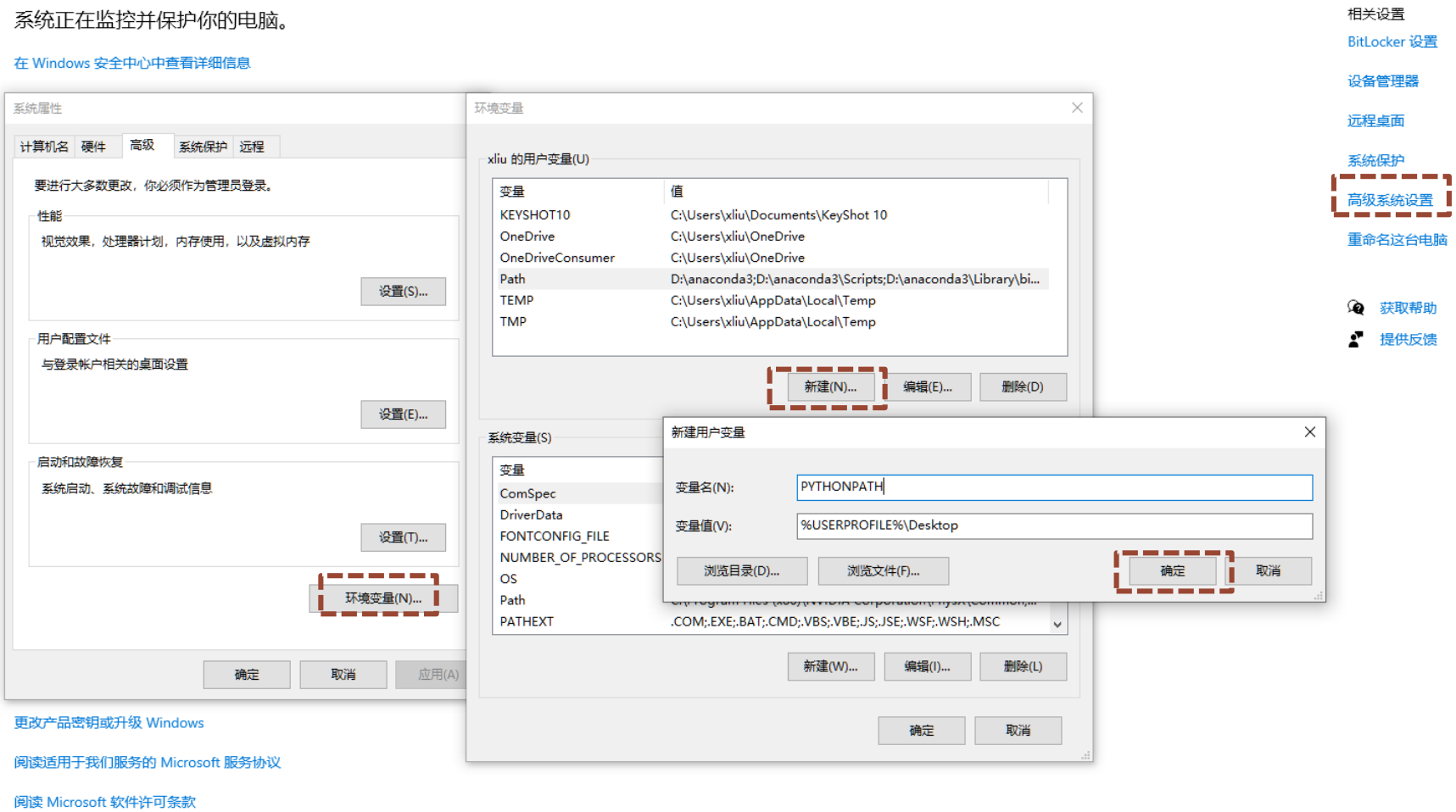
当前目录

命令提示符 - python

```
Microsoft Windows [版本 10.0.19045.2251]
(c) Microsoft Corporation。保留所有权利。

C:\Users\xliu>python
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated.  Libraries may fail to load.  To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> import a
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'a'
>>> exit()
```

```
C:\Users\xliu\Desktop>python
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated.  Libraries may fail to load.  To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> import a
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__', 'a']
>>>
```

环境变量PYTHONPATH（推荐）



PYTHONPATH **%USERPROFILE%\Desktop**

命令提示符 - python

```
Microsoft Windows [版本 10.0.19045.2251]
(c) Microsoft Corporation。保留所有权利。

C:\Users\xliu>python
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated.  Libraries may fail to load.  To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> import a
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__', 'a']
>>> import sys, pprint
>>> pprint.pprint(sys.path)
['',
 'C:\\Users\\xliu\\Desktop',
 'D:\\anaconda3\\python39.zip',
 'D:\\anaconda3\\DLLs',
 'D:\\anaconda3\\lib',
 'D:\\anaconda3',
 'D:\\anaconda3\\lib\\site-packages',
 'D:\\anaconda3\\lib\\site-packages\\locket-0.2.1-py3.9.egg',
 'D:\\anaconda3\\lib\\site-packages\\win32',
 'D:\\anaconda3\\lib\\site-packages\\win32\\lib',
 'D:\\anaconda3\\lib\\site-packages\\Pythonwin']
>>>
```

需要重启cmd更新环境变量

sys.path变量

命令提示符 - python

```
Microsoft Windows [版本 10.0.19045.2251]
(c) Microsoft Corporation。保留所有权利。

C:\Users\xliu>python
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated.  Libraries may fail to load.  To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> import a
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'a'
>>> import sys
>>> print(sys.path)
['', 'D:\\anaconda3\\python39.zip', 'D:\\anaconda3\\DLLs', 'D:\\anaconda3\\lib', 'D:\\anaconda3', 'D:\\anaconda3\\lib\\s
ite-packages', 'D:\\anaconda3\\lib\\site-packages\\locket-0.2.1-py3.9.egg', 'D:\\anaconda3\\lib\\site-packages\\win32',
'D:\\anaconda3\\lib\\site-packages\\win32\\lib', 'D:\\anaconda3\\lib\\site-packages\\Pythonwin']
>>> sys.path.append(r"C:\Users\xliu\Desktop")
>>> print(sys.path)
['', 'D:\\anaconda3\\python39.zip', 'D:\\anaconda3\\DLLs', 'D:\\anaconda3\\lib', 'D:\\anaconda3', 'D:\\anaconda3\\lib\\s
ite-packages', 'D:\\anaconda3\\lib\\site-packages\\locket-0.2.1-py3.9.egg', 'D:\\anaconda3\\lib\\site-packages\\win32',
'D:\\anaconda3\\lib\\site-packages\\win32\\lib', 'D:\\anaconda3\\lib\\site-packages\\Pythonwin', 'C:\\Users\\xliu\\Deskt
op']
>>> import a
>>>
```

sys.path.append()

第三方模块

lib/site-packages目录存放了用户安装的第三方模块（库）

科学计算：numpy, scipy, pandas, ...

绘图可视化：matplotlib, seaborn, plotly, ...

机器学习：scikit-learn, XGBoost, LightGBM, ...

网络模块：socket, urllib, urllib2, scrapy, ...

GUI模块：tkinter, pyqt, wxpython, ...

Anaconda环境和包管理

环境管理

包管理（模块、拓展）

安装、卸载、更新

安装包更新

numpy（Numerical Python）是Python的一种开源的数值计算扩展，可用来存储和处理大型矩阵。

wxpython安装

C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

```
(base) PS C:\Users\xliu> pip list
Package                           Version
--------------------------------- -------------------
alabaster                         0.7.12
anaconda-client                   1.7.2
anaconda-navigator                1.9.12
anaconda-project                  0.8.3
argh
asn1crypto
astroid
astropy
atomicwrites
attrs                             19.3.0
autopep8                          1.5.3
Babel                             2.8.0
backcall                          0.2.0
backports.functools-lru-cache     1.6.1
backports.shutil-get-terminal-size 1.0.0
backports.tempfile                1.0
backports.weakref                 1.0.post1
bcrypt                            3.1.7
beautifulsoup4                    4.9.1
bitarray                          1.4.0
bkcharts                          0.2
bleach                            3.1.5
```

pip list：查询安装包

C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

```
wincertstore                      0.2
wrapt                             1.11.2
xlrd                              1.2.0
XlsxWriter
xlwings
xlwt
xmltodict
yapf
zict
zipp
zope.event
zope.interface                    4.7.1
(base) PS C:\Users\xliu>
(base) PS C:\Users\xliu> pip install wxpython
Collecting wxpython
  Using cached wxPython-4.1.0-cp38-cp38-win_amd64.whl (17.9 MB)
Requirement already satisfied: numpy; python_version >= "3.0" in d:\anaconda3\lib\site-packages (from wxpython) (1.18.5)
Requirement already satisfied: six in d:\anaconda3\lib\site-packages (from wxpython) (1.15.0)
Requirement already satisfied: pillow in d:\anaconda3\lib\site-packages (from wxpython) (7.2.0)
Installing collected packages: wxpython
Successfully installed wxpython-4.1.0
(base) PS C:\Users\xliu>
```

pip install 安装包
pip uninstall 安装包
pip install –upgrade 安装包

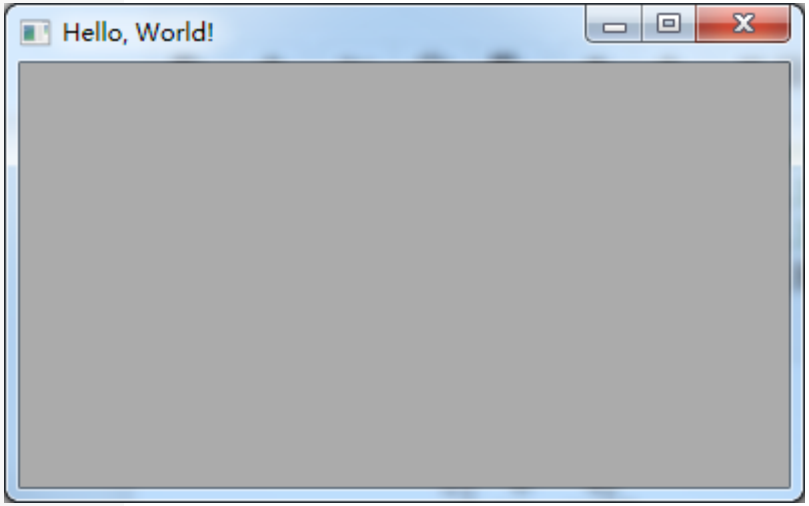
Installed | Channels | Update index... | Search Packages

Name	T	Description	Version
winpty		Winpty provides an interface similar to a unix pty-master for communicating with windows console programs.	0.4.3
wrapt		Module for decorators, wrappers and monkey patching.	1.11.2
wxpython		Cross platform gui toolkit for python, "phoenix" version	4.1.0
xlrd		Library for developers to extract data from microsoft excel (tm) spreadsheet files	1.2.0
xlsxwriter		A python module for creating excel xlsx files	1.2.9
xlwings		Interact with excel from python and vice versa	0.19.5

基于wxWidgets，跨平台，控件多，案例丰富 (http://wxpython.org)

第一个Python GUI窗口

```python
# 加载wx模块
import wx

# 创建一个应用程序对象
app1 = wx.App(False)

# 创建一个顶层窗口对象
frame1 = wx.Frame(None, wx.ID_ANY, "Hello, World!")

# 显示窗口
frame1.Show(True)

# 运行程序
app1.MainLoop()
```



app = wx.App(False)：创建一个应用程序对象（与操作系统的接口）

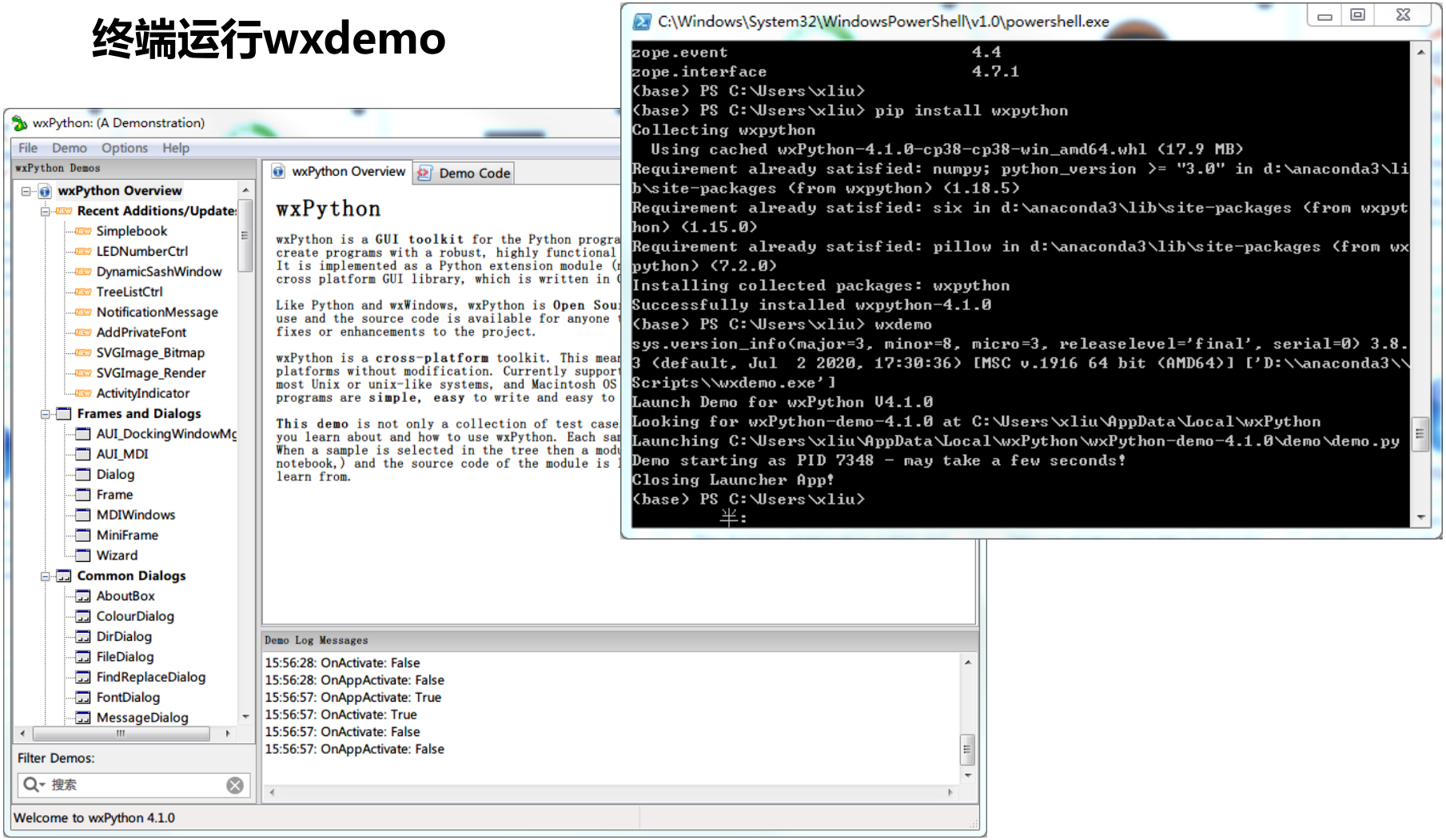
frame = wx.Frame(None, wx.ID_ANY, “Hello, World!”)：使用 “None” 来表示这个frame是顶层的窗口，没有父窗口；使用 “wx.ID_ANY” 让 wxWidgets 来给我们挑选一个ID（窗口对象，全局坐标系）

frame.Show(True)：窗口默认隐藏，显示这个Frame

app.MainLoop()：运行这个应用程序

wxpython demo

终端运行wxdemo

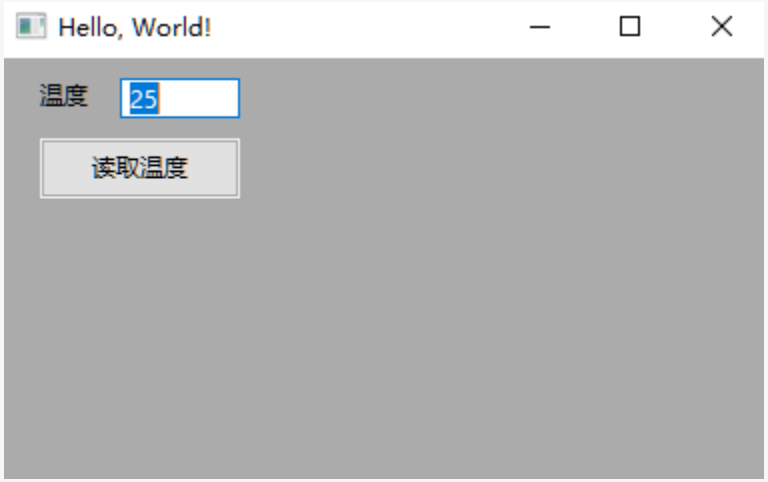


D:\anaconda3\Scripts\wxdemo.exe

自定义窗口类

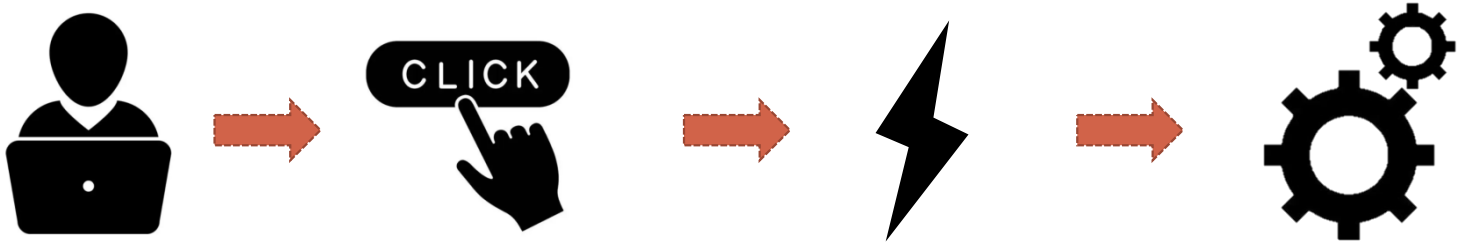
```
# 加载wx模块
import wx

# 创建自定义窗口类
class myframe(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, wx.ID_ANY, "Hello, World!")
        # 添加控件
        self.text1 = wx.StaticText(self, -1, "温度", size=(40, 20), pos=(20, 10)) # 静态文本框
        self.text2 = wx.TextCtrl(self, -1, "25", size=(60, 20), pos=(60, 10)) # 单行动态文本框
        self.button1 = wx.Button(self, -1, "确定", size=(100, 30), pos=(20, 40)) # 按钮

# 创建一个应用程序对象
app2 = wx.App(False)

# 创建一个顶层窗口对象
frame2 = myframe()

# 显示窗口
frame2.Show(True)

# 运行程序
app2.MainLoop()
```

参考wxdemo



事件处理

- **事件源：**事件发生的场所，就是各个控件，例如按钮事件的事件源是按钮
- **事件：**wxpython中的事件被封装为事件类wx.Event及其之类，例如按钮事件的事件类是wx.CommandEvent，鼠标事件类是wx.MoveEvent
- **事件处理程序：**一个响应用户事件的方法



用户 **单击按钮** **单击事件** **处理事件**

```python
# 加载wx模块
import wx, random

# 创建自定义窗口类
class myframe(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, wx.ID_ANY, "Hello, World!")
        # 添加控件
        self.text1 = wx.StaticText(self, -1, "温度", size=(40, 20), pos=(20, 10)) # 静态文本框
        self.text2 = wx.TextCtrl(self, -1, "25", size=(60, 20), pos=(60, 10)) # 单行动态文本框
        self.button1 = wx.Button(self, -1, "确定", size=(100, 30), pos=(20, 40)) # 按钮

        # 添加事件
        self.Bind(wx.EVT_BUTTON, self.On_click, self.button1) #按钮控件关联函数

    def On_click(self, event):
        self.text2.SetValue(str(random.randint(25,100)))

# 创建一个应用程序对象
app3 = wx.App(False)

# 创建一个顶层窗口对象
frame3 = myframe()

# 显示窗口
frame3.Show(True)

# 运行程序
app3.MainLoop()
```

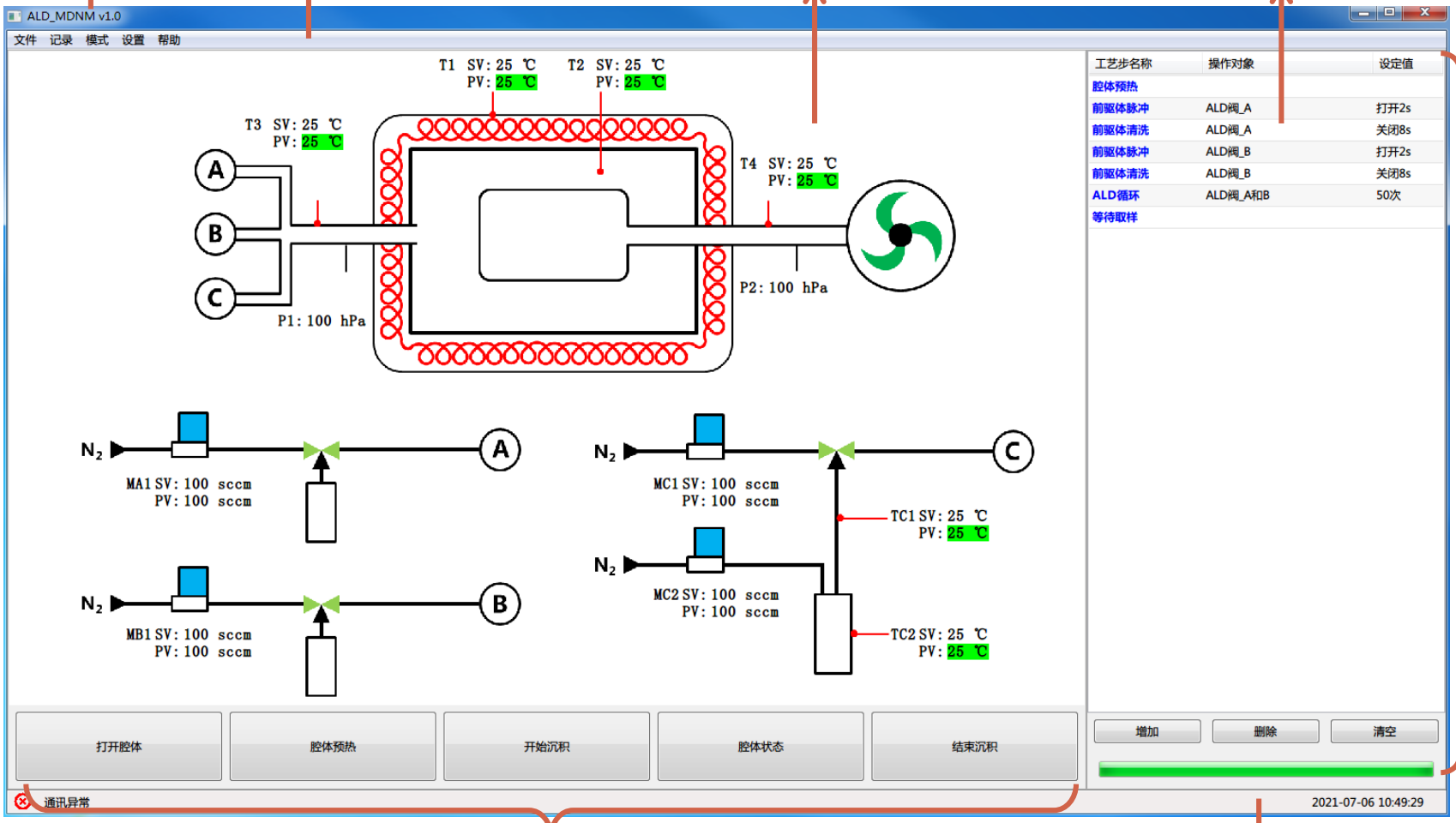
按钮事件源

绑定事件，wx.EVT_BUTTON是事件类型，即按钮单击事件

事件处理程序

原子层沉积系统控制软件主窗口

窗口　　菜单栏　　腔体面板　　工艺面板



垂直布局

水平布局　　状态栏

面板(wx.Panel)和盒子(wx.BoxSizer)布局器

父窗口

```
class TestPanel(wx.Panel):
    def __init__(self, parent):
        wx.Panel.__init__(self, parent)
        # 添加控件
        self.text1 = wx.StaticText(self, -1, "温度", size=(40, 20), pos=(20, 10)) # 静态文本框
        self.text2 = wx.TextCtrl(self, -1, "25", size=(60, 20), pos=(60, 10)) # 单行动态文本框
        self.button1 = wx.Button(self, -1, "确定", size=(100, 30), pos=(20, 40)) # 按钮

        # 添加事件
        self.Bind(wx.EVT_BUTTON, self.On_click, self.button1) #按钮控件关联函数

        # 利用wx.BoxSizer布局控件
        self.text_box = wx.BoxSizer(wx.HORIZONTAL)  # 水平布局器
        self.text_box.Add(self.text1, proportion=1, flag=wx.EXPAND | wx.ALL, border=0)
        self.text_box.Add(self.text2, proportion=1, flag=wx.EXPAND | wx.ALL, border=0)

        self.panel_box = wx.BoxSizer(wx.VERTICAL)  # 垂直布局器
        self.panel_box.Add(self.text_box, proportion=2, flag=wx.EXPAND | wx.ALL, border=0)
        self.panel_box.Add(self.button1, proportion=1, flag=wx.EXPAND | wx.ALL, border=0)

        self.SetSizer(self.panel_box)  # 设置腔体面板布局器

    def On_click(self, event):
        self.text2.SetValue(str(random.randint(25,100)))
```

创建模块

模块是在函数和类的基础上，将一系列相关的代码组织在一起的集合体，在Python中以“.py”结尾的文件就是一个模块

a.py

```
"module a “

# define global a
a = 1

# define function func1
def func1():
    print(a)

# define class newclass1
class newclass1():
    var1 = 1
    def method1(self):
        print("hello world!")
```

导入模块a

```
print(dir())
import a
print(dir())
a.func1()
```

```
['In', 'Out', '_', '__', '___', '__builtin__',
i', '_i1', '_ih', '_ii', '_iii', '_oh', 'exit'
['In', 'Out', '_', '__', '___', '__builtin__',
i', '_i1', '_ih', '_ii', '_iii', '_oh', 'a',
1
```

a在全局命名空间中

通过a.func1()调用模块中的变量、函数和类

func1在a的命名空间中

模块内置属性

```
pprint.pprint(dir(a))
```

内置属性

```
['__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'a',
 'func1',
 'newclass1']
```

```
print(a.__doc__)
```

文档字符串

```
module a
```

```
print(a.__file__)
```

模块位置

```
C:\Users\xliu\Desktop\a.py
```

```
print(a.__name__)
```

模块文件名

```
a
```

```
print(a.__builtins__)
```

内置命名空间字典

a: 全局变量

func1: 函数

newclass1: 类

注意：__name__为“__main__”表明该模块自身在运行，否则模块是被导入执行（__name__为模块文件名）

__name__内置属性

命令提示符 - python

```
Microsoft Windows [版本 10.0.19045.2251]
(c) Microsoft Corporation。保留所有权利。

C:\Users\xliu>cd Desktop

C:\Users\xliu\Desktop>python b.py
__main__

C:\Users\xliu\Desktop>python
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated.  Libraries may fail to load.  To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> import b
b
>>> print(b.__name__)
b
>>>
```

直接执行

import导入

import导入后，模块的顶层代码被执行了

利用if __name__ == “__main__”调试模块

panelGUI.py - C:\Users\xliu\Desktop\模块\panelGUI.py (2.7.11)

File Edit Format Run Options Window Help

```python
# -*- coding: utf-8 -*-
# 加载wx模块
import wx, random

class TestPanel(wx.Panel):
    def __init__(self, parent):
        wx.Panel.__init__(self, parent)
        # 添加控件
        self.text1 = wx.StaticText(self, -1, "温度", size=(40, 20), pos=(20, 10)) # 静态文本框
        self.text2 = wx.TextCtrl(self, -1, "25", size=(60, 20), pos=(60, 10)) # 单行动态文本框
        self.button1 = wx.Button(self, -1, "确定", size=(100, 30), pos=(20, 40)) # 按钮

        # 添加事件
        self.Bind(wx.EVT_BUTTON, self.On_click, self.button1) #按钮控件关联函数

        # 利用wx.BoxSizer布局控件
        self.text_box = wx.BoxSizer(wx.HORIZONTAL)  # 水平布局器
        self.text_box.Add(self.text1, proportion=1, flag=wx.EXPAND | w
        self.text_box.Add(self.text2, proportion=1, flag=wx.EXPAND | w

        self.panel_box = wx.BoxSizer(wx.VERTICAL)  # 垂直布局器
        self.panel_box.Add(self.text_box, proportion=2, flag=wx.EXPAND
        self.panel_box.Add(self.button1, proportion=1, flag=wx.EXPAND

        self.SetSizer(self.panel_box)  # 设置腔体面板布局器

    def On_click(self, event):
        self.text2.SetValue(str(random.randint(25,100)))

if __name__ == "__main__":
    # 创建一个应用程序对象
    app4 = wx.App(False)

    # 创建一个顶层框架对象（全局坐标系）
    frame4 = wx.Frame(None, wx.ID_ANY, "Hello, World!")

    # 创建一个局部面板对象（局部坐标系）
    panel4 = TestPanel(frame4)

    # 显示框架
    frame4.Show(True)

    # 运行程序
    app4.MainLoop()
```

模块做为主模块单独执行

命令提示符 - D:\Anaconda3\python.exe panelGUI.py

```
Microsoft Windows [版本 10.0.19045.3324]
(c) Microsoft Corporation。保留所有权利。

C:\Users\xliu>cd Desktop/模块

C:\Users\xliu\Desktop\模块>D:\Anaconda3\python.exe panelGUI.py
```

利用if __name__ == “__main__”调试模块

mainGUI.py - C:\Users\xliu\Desktop\模块\mainGUI.py (2.7.11)

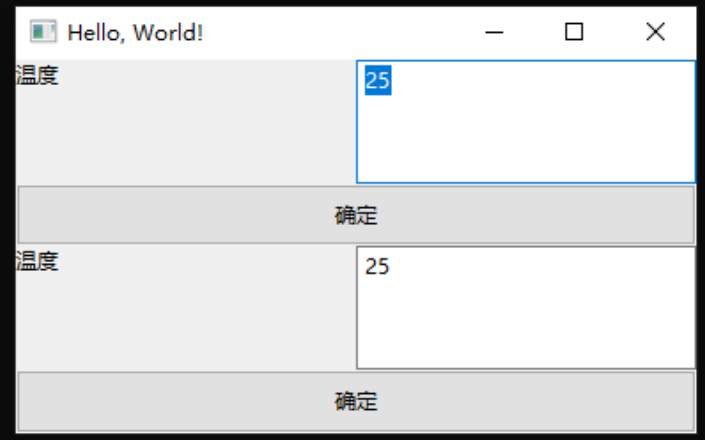
File Edit Format Run Options Window Help

```python
# -*- coding: utf-8 -*-
# 加载wx模块
import wx, panelGUI

# 创建一个应用程序对象
app = wx.App(False)

# 创建一个顶层框架对象
frame = wx.Frame(None, wx.ID_ANY, "Hello, World!")

# 创建第一个局部面板对象（局部坐标系）
panel1 = panelGUI.TestPanel(frame)

# 创建第二个局部面板对象（局部坐标系）
panel2 = panelGUI.TestPanel(frame)

# 框架内布局
panel_box = wx.BoxSizer(wx.VERTICAL)
panel_box.Add(panel1, proportion=1, flag = wx.EXPAND | wx.ALL
panel_box.Add(panel2, proportion=1, flag = wx.EXPAND | wx.ALL
frame.SetSizer(panel_box)

# 显示框架
frame.Show(True)

# 运行程序
app.MainLoop()
```

导入模块

命令提示符 - D:\Anaconda3\python.exe mainGUI.py

```
Microsoft Windows [版本 10.0.19045.3324]
(c) Microsoft Corporation。保留所有权利。

C:\Users\xliu>cd Desktop/模块

C:\Users\xliu\Desktop\模块>D:\Anaconda3\python.exe panelGUI.py

C:\Users\xliu\Desktop\模块>D:\Anaconda3\python.exe mainGUI.py
```



每个模块即可做主模块单独执行，也可作为构件

模块导入import

__builtins__

len, map, filter, list, tuple ...

c.py

```
"module c"
import a

# define global c
c = 2

# define function func1
def outer():
    print(a.a)
    def inner():
        print("hello world!")

outer()
```

global

enclosing

local

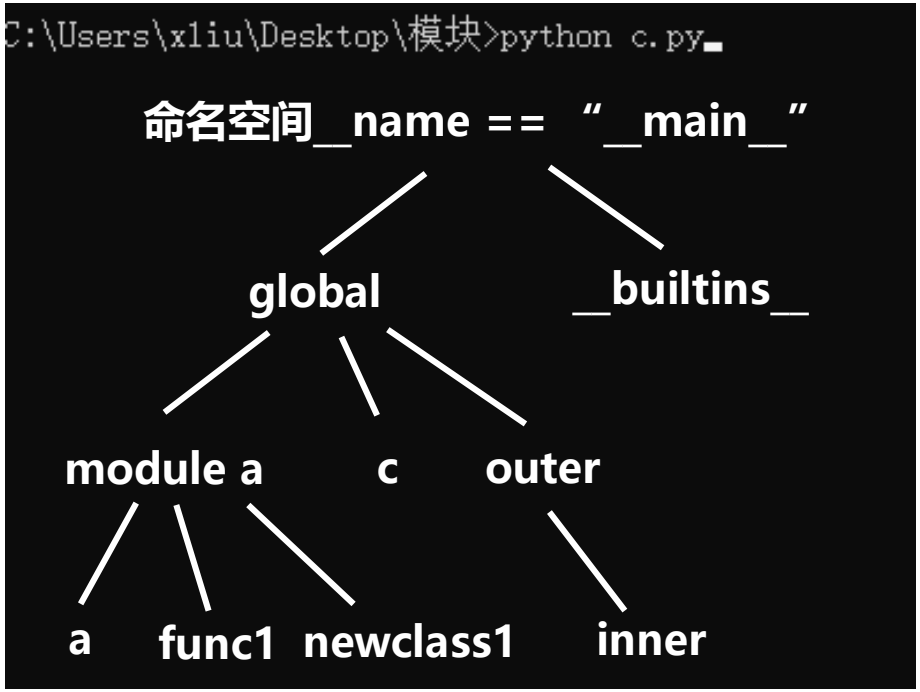
a.py

```
"module a"

# define global a
a = 1

# define function func1
def func1():
    print(a)

# define class newclass1
class newclass1():
    var1 = 1
    def method1(self):
        print("hello world!")
```

global

enclosing

local



- 隐性赋值（去掉py即为模块名字）
- 顶层赋值代码执行生成模块属性
- 使用限定符按照命名空间访问属性
- 函数不划分命名空间，只在调用outer()时inner才一闪而过

模块导入reload

当一个模块被导入后，模块顶层部分的代码只会被执行一次，可以使用reload()函数重新执行模块里顶层部分的代码

```
"module c"
import a

# define global c
c = 2

# define function func1
def outer():
    print(a.a)
    def inner():
        print("hello world!")

outer()
```

1

```
"module c"
import a

# define global c
c = 2

a.a = 3
import a

# define function func1
def outer():
    print(a.a)
    def inner():
        print("hello world!")

outer()
```

3

```
"module c"
import a

# define global c
c = 2

a.a = 3
reload(a)

# define function func1
def outer():
    print(a.a)
    def inner():
        print("hello world!")

outer()
```

1

重新导入之前导入过的模块

不重启Python使配置就生效必须reload

模块导入from

__builtins__

len, map, filter, list, tuple ...

d.py

```
"module d"
from a import *

# define global c
c = 2

# define function func1
def outer():
    print(a)
    def inner():
        print("hello world!")

outer()
```

global

enclosing

local

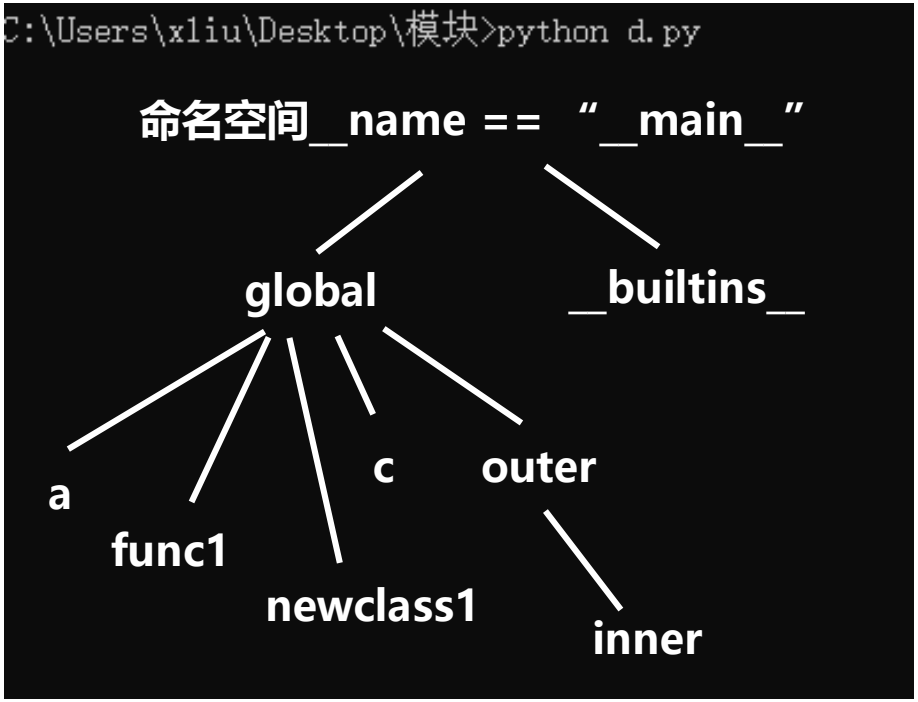
a.py

```
"module a"

# define global a
a = 1

# define function func1
def func1():
    print(a)

# define class newclass1
class newclass1():
    var1 = 1
    def method1(self):
        print("hello world!")
```

global

enclosing

local



可能污染全局命名空间

嵌入导入

m.py

m.py - C:\Users\xliu\Desktop\模块\m.py

File Edit Format Run Options Windo

```
val1 = "1"
import m1
print("hello", val1, m1.val1)
```

1

2

4

m1.py

m1.py - C:\Users\xliu\Desktop\模块\m1.py

File Edit Format Run Options Window

```
val1 = "m1value"
#import m2
#print("m2.val2=", m2.val2)
#print("m2.m1.val1=", m2.m1.val1)
```

3

Anaconda Prompt (Anaconda3) - python

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

(base) C:\Users\xliu\Desktop>cd 模块

(base) C:\Users\xliu\Desktop\模块>python m.py
hello 1 m1value

(base) C:\Users\xliu\Desktop\模块>python
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import m
hello 1 m1value
>>> print(m.m1)
<module 'm1' from 'C:\\Users\\xliu\\Desktop\\模块\\m1.py'>
>>> print(m.m1.val1)
m1value
>>> print(m.val1)
1
>>>
```

执行顶层的导入

嵌入导入

m1.py

m1.py - C:\Users\xliu\Desktop\模块\m1.py

File Edit Format Run Options Window

```
val1 = "m1value"                    1   5
import m2                           2   6
print("m2.val2=", m2.val2)              7   9
#print("m2.m1.val1=", m2.m1.val1)
```

m2.py

m2.py - C:\Users\xliu\Desktop\模块\m2.py

File Edit Format Run Options Window

```
val2 = "m2value"                3
import m1                       4
print("m1.val1=", m1.val1)          8
```

Anaconda Prompt (Anaconda3)

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

(base) C:\Users\xliu\Desktop>cd 模块

(base) C:\Users\xliu\Desktop\模块>python m1.py
m2.val2= m2value
m1.val1= m1value
m2.val2= m2value

(base) C:\Users\xliu\Desktop\模块>python m2.py
m1.val1= m1value
m2.val2= m2value
m1.val1= m1value

(base) C:\Users\xliu\Desktop\模块>_
```

第6步：发现递归不再进入，而且此时已经有以m2为根的命名空间

第7步：m1执行结束后，返回m2继续执行

第8步：m2执行结束后，返回m1继续执行

嵌入导入

m1.py

m1.py - C:\Users\xliu\Desktop\模块\m1.py

File Edit Format Run Options Window

```
val1 = "m1value"                    1
import m2                           2
print("m2.val2=", m2.val2)          5
print("m2.m1.val1=", m2.m1.val1)    6
                                    7
                                    8
```

m2.py

m2.py - C:\Users\xliu\Desktop\模块\m2.py

File Edit Format Run Options Window

```
val2 = "m2value"                    3
import m1                           4
print("m1.val1=", m1.val1)
```

Anaconda Prompt (Anaconda3)

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

(base) C:\Users\xliu\Desktop>cd 模块

(base) C:\Users\xliu\Desktop\模块>python m1.py
m2.val2= m2value
Traceback (most recent call last):
  File "m1.py", line 2, in <module>
    import m2
  File "C:\Users\xliu\Desktop\模块\m2.py", line 2, in <module>
    import m1
  File "C:\Users\xliu\Desktop\模块\m1.py", line 4, in <module>
    print("m2.m1.val1=", m2.m1.val1)
AttributeError: partially initialized module 'm2' has no attribute 'm1' (most likely due to a circular import)

(base) C:\Users\xliu\Desktop\模块>python m2.py
m1.val1= m1value
m2.val2= m2value
m2.m1.val1= m1value
m1.val1= m1value

(base) C:\Users\xliu\Desktop\模块>
```

第8步：m2的命名空间已经建立，但m2下的m1还未完成所以执行出错

导入后才有名字树节点

嵌入导入

m1.py

m1.py - C:\Users\xliu\Desktop\模块\m1.py

File Edit Format Run Options Window

```
val1 = "m1value"                    3
import m2                           4
print("m2.val2=", m2.val2)                  8
print("m2.m1.val1=", m2.m1.val1)            9
```

m2.py

m2.py - C:\Users\xliu\Desktop\模块\m2.py

File Edit Format Run Options Window

```
val2 = "m2value"            1       5
import m1                   2       6
print("m1.val1=", m1.val1)          7       10
```

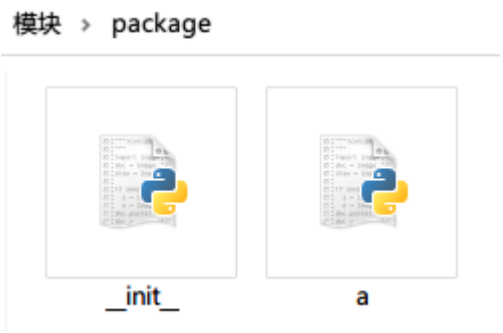
Anaconda Prompt (Anaconda3)

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

(base) C:\Users\xliu\Desktop>cd 模块

(base) C:\Users\xliu\Desktop\模块>python m1.py
m2.val2= m2value
Traceback (most recent call last):
  File "m1.py", line 2, in <module>
    import m2
  File "C:\Users\xliu\Desktop\模块\m2.py", line 2, in <module>
    import m1
  File "C:\Users\xliu\Desktop\模块\m1.py", line 4, in <module>
    print("m2.m1.val1=", m2.m1.val1)
AttributeError: partially initialized module 'm2' has no attribute 'm1' (most likely due to a circular import)

(base) C:\Users\xliu\Desktop\模块>python m2.py
m1.val1= m1value
m2.val2= m2value
m2.m1.val1= m1value
m1.val1= m1value

(base) C:\Users\xliu\Desktop\模块>
```

第9步：m1已经执行完毕，m2下的m1命名空间已经建立完毕

创建模块包

包是一个包含了多个模块的文件夹，可以有多层次的子包，形成一个包的层次结构（文件夹下默认包含__init__.py文件）



目录结构

/package

__init__.py （空文件）

a.py

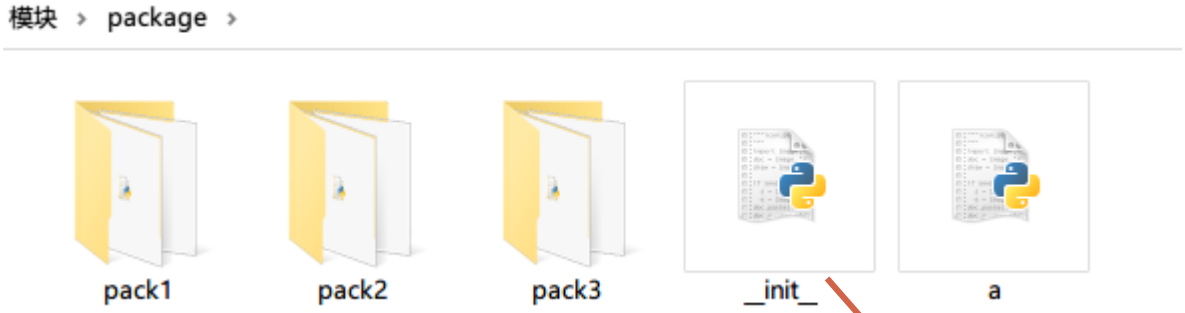
Anaconda Prompt (Anaconda3) - python

```
(base) C:\Users\xliu\Desktop>cd 模块

(base) C:\Users\xliu\Desktop\模块>python
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] :: A
, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import package
>>> print(package)
<module 'package' from 'C:\\Users\\xliu\\Desktop\\模块\\package\\__init__.py'>
>>> dir(package)
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
ckage__', '__path__', '__spec__']
>>> exit()

(base) C:\Users\xliu\Desktop\模块>python
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] :: A
, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import package.a
>>> print(package.a)
<module 'package.a' from 'C:\\Users\\xliu\\Desktop\\模块\\package\\a.py'>
>>> print(package.a.a)
1
>>>
```

模块包

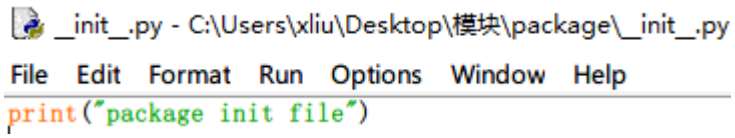
Python3.2版本后没有init文件，文件夹也可以被当成命名空间包导入

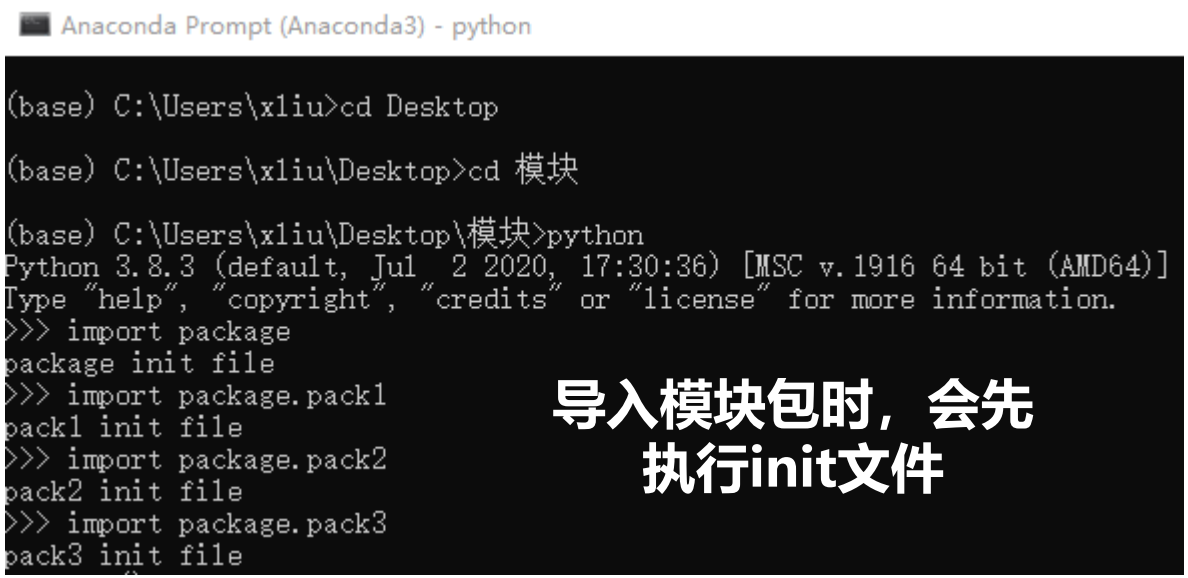
创建模块包



目录结构

/package
- /pack1
  - __init__.py
  - a.py
- /pack2
  - __init__.py
  - a.py
- /pack3
  - __init__.py
  - a.py
- __init__.py
- a.py





导入模块包时，会先执行init文件

创建模块包

Anaconda Prompt (Anaconda3) - python

```
(base) C:\Users\xliu\Desktop\模块>python
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import package
package init file
>>> import package.pack1
pack1 init file
>>> import package.pack2
pack2 init file
>>> import package.pack3
pack3 init file
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__', 'package']
>>> dir(package)
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__path__', '__s
pack1', 'pack2', 'pack3']
>>> exit()

(base) C:\Users\xliu\Desktop\模块>python
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import package.pack1
package init file
pack1 init file
>>>
```

导入子包，也会执行父包的init文件

创建模块包

init.py - C:\Users\xliu\Desktop\模块\package_init_.py

File Edit Format Run Options Window Help

```
print("package init file")
__all__ = ["pack1", "pack2"]
```

__all__列表列出需要导入的模块和子包

Anaconda Prompt (Anaconda3) - python

```
(base) C:\Users\xliu\Desktop\模块>python
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import package
package init file
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__', 'package']
>>> dir(package)
['__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__path__',
__spec__']
>>> print(package.__all__)
['pack1', 'pack2']
>>> exit()

(base) C:\Users\xliu\Desktop\模块>python
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from package import *
package init file
pack1 init file
pack2 init file
>>> dir(package)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'package' is not defined
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__', 'pack1', 'pack2']
>>>
```

from导入__all__列表中的子包

小结

- 模块的导入规则和命名空间
- 模块的内置属性，如__name__, __doc__等
- 模块的创建和使用

下一节：Python编程实践（串口通讯）