Lairx - 004

Lx4-TDR 项目手册

Total Directory Report

Flashy

2023-8-20

修改记录

版本	更新日期	描述
1.0	2023/8/20	**建立初始版本**

目 录

第1章 环境安装与部署

1.1 Python

1.1.1. Python 简介

Python 是一种解释型、面向对象、动态数据类型的高级程序设计语言

1.1.2. Python 安装

Pycharm 安装自带 python，以下不展开详细介绍

1.2 Pycharm

1.2.1. Pycharm 简介

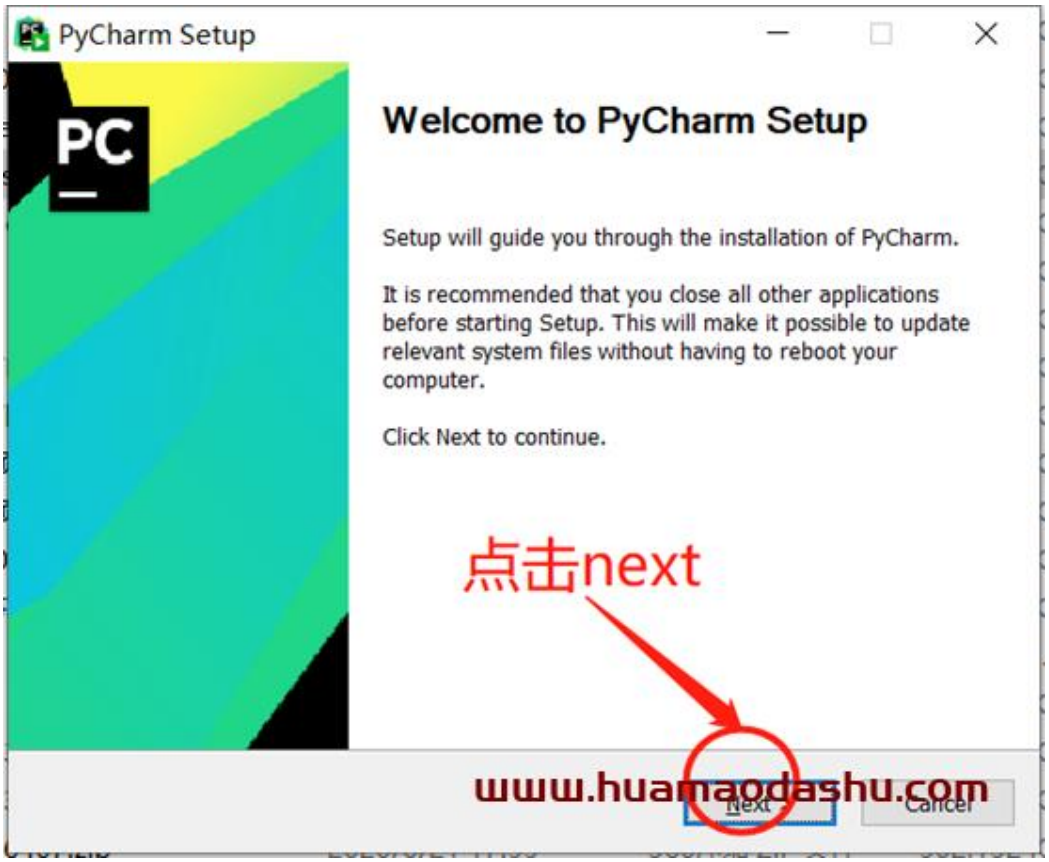
很多软件厂商针对编程语言开发了 IED 工具，Python 中非常知名 IDE 工具的便是 Pycharm.

PyCharm 可以帮助用户在使用 Python 语言开发时，提高代码的编写效率.

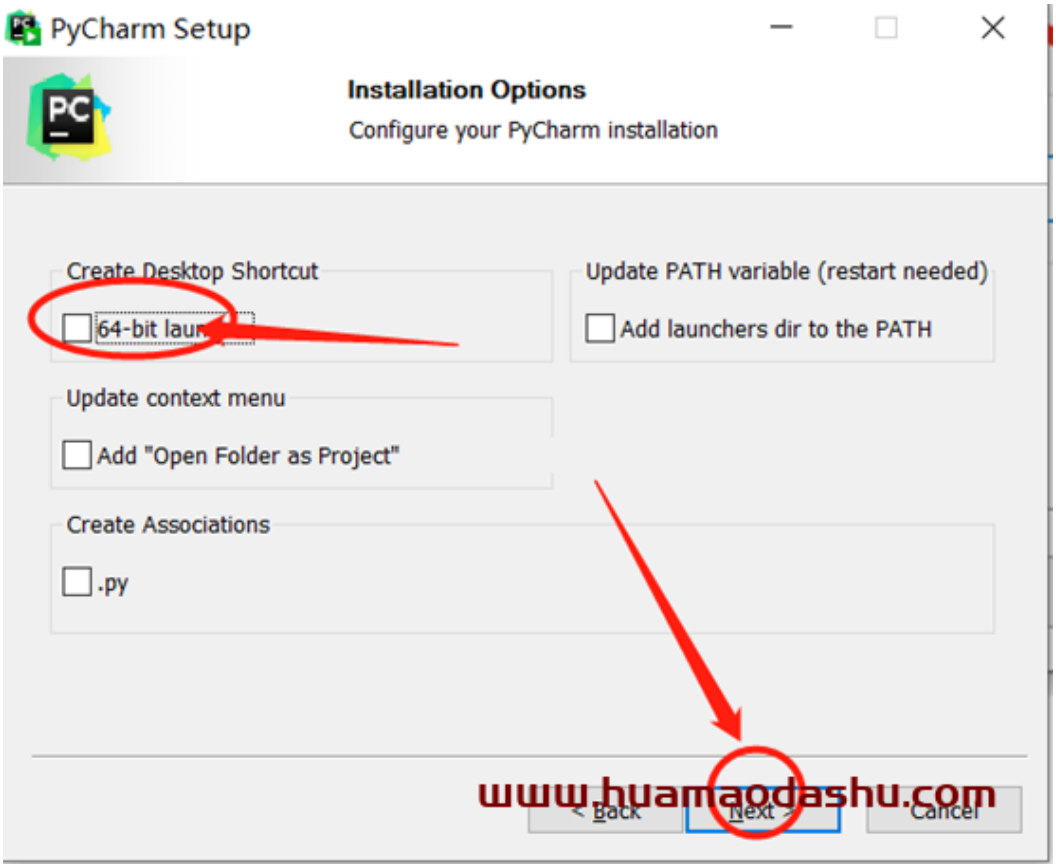
IDE(Integrated Development Environment): 集成开发环境，是用于提供程序开发环境的应用程序,一般包括代码编辑器，编译器，调试器和图形用户界面等工具.

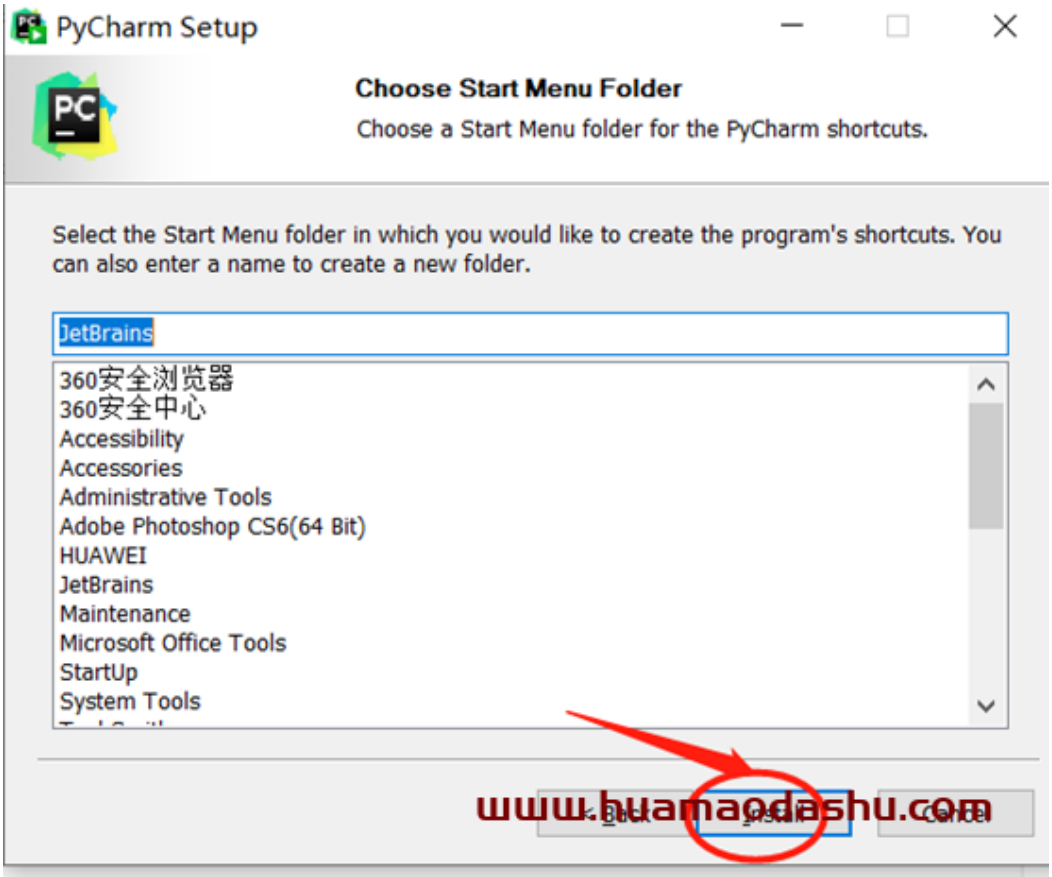
1.2.2. Pycharm 安装

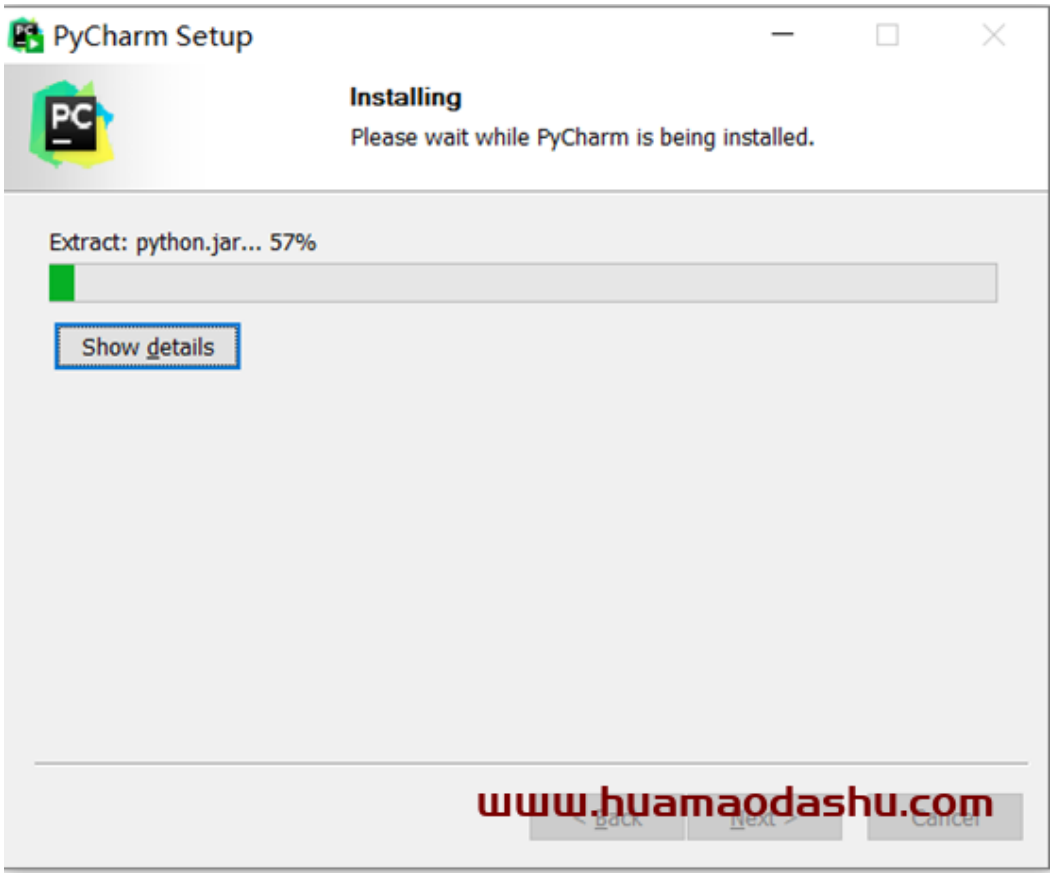
下载完软件之后，双击点击下载的软件

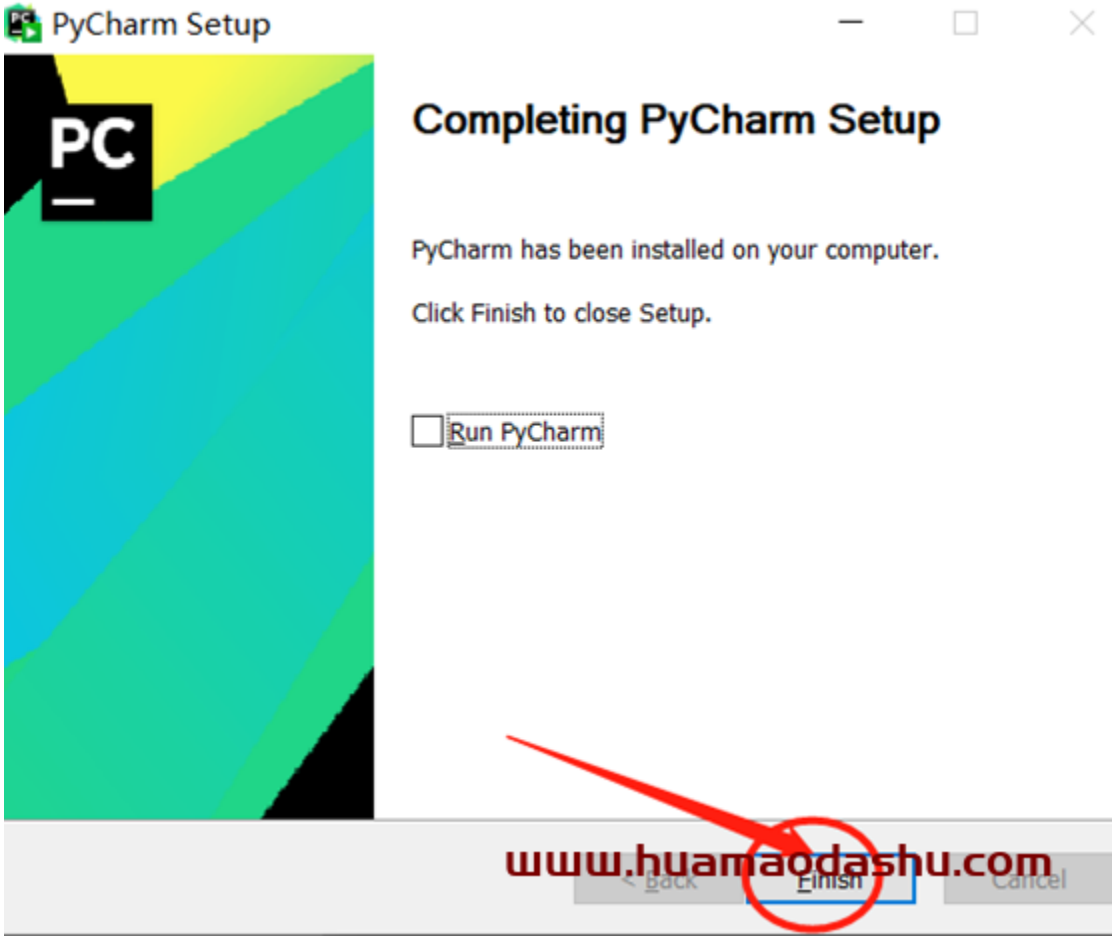

PyCharm Setup
Welcome to PyCharm Setup
Setup will guide you through the installation of PyCharm.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
点击next
Next
Cancel
www.huamaodashu.com


PyCharm Setup
Choose Install Location
Choose the folder in which to install PyCharm.
Setup will install PyCharm in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
选择安装的路径
Destination Folder
C:\Program Files\JetBrains\PyCharm 2020.3
Browse...
Space required: 1.1 GB
Space available: 19.9 GB
www.huamaodashu.com

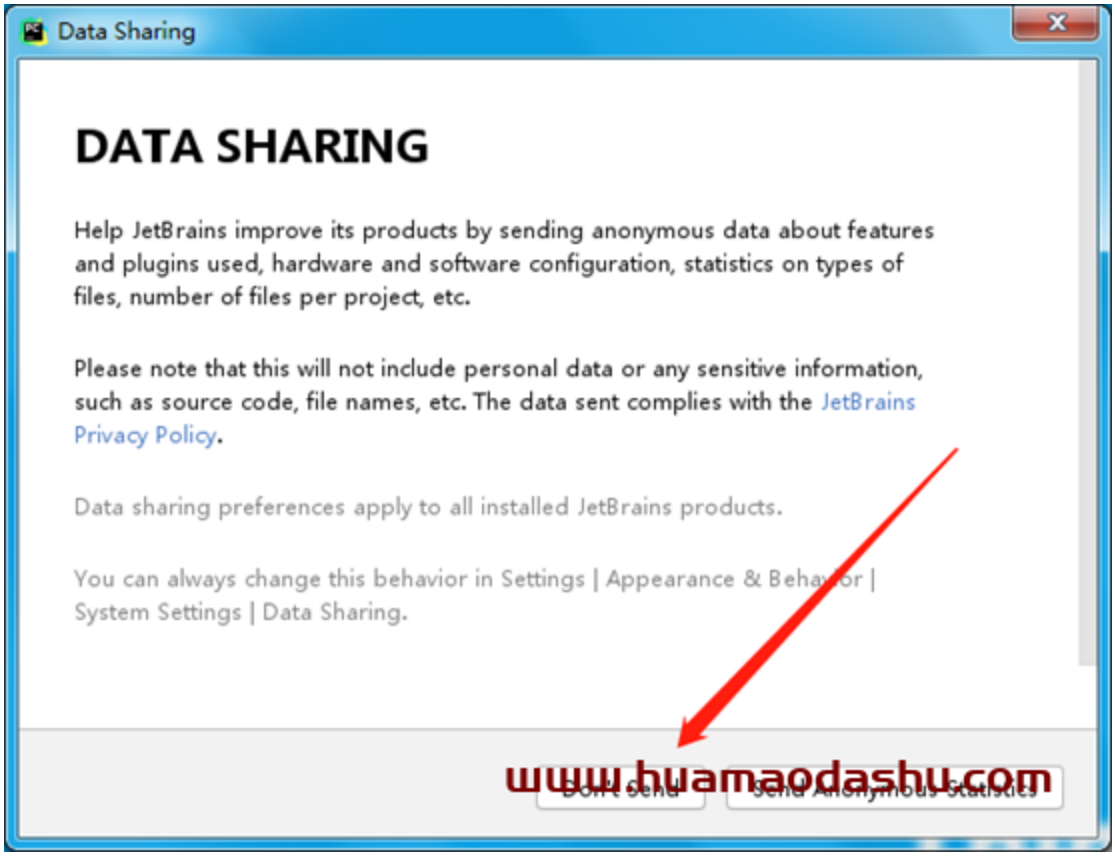

PyCharm Setup
Installation Options
Configure your PyCharm installation
Create Desktop Shortcut
64-bit laun
Update PATH variable (restart needed)
Add launchers dir to the PATH
Update context menu
Add "Open Folder as Project"
Create Associations
.py
< Back
Next >
Cancel
www.huamaodashu.com

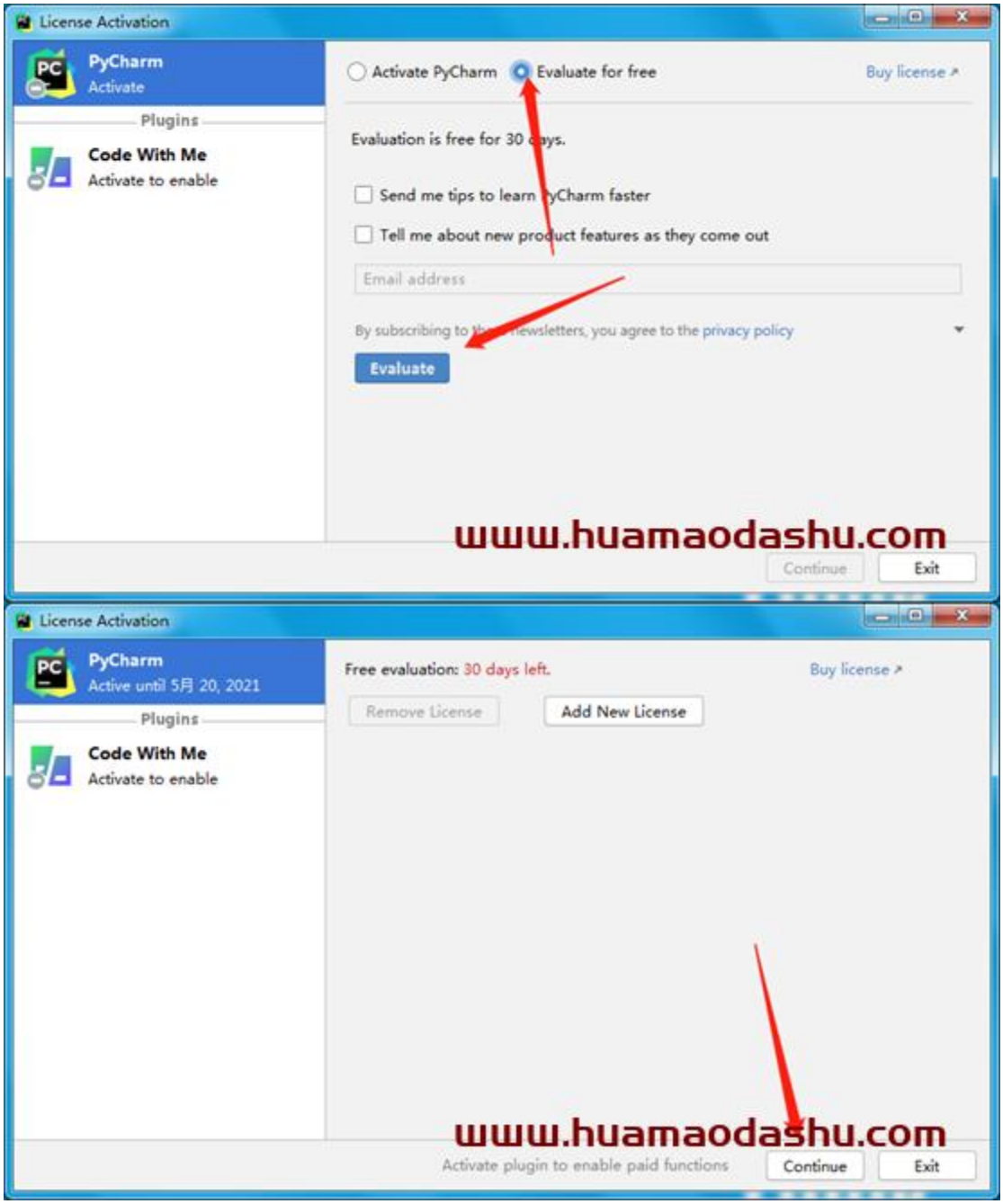

PyCharm Setup
Choose Start Menu Folder
Choose a Start Menu folder for the PyCharm shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
JetBrains
360安全浏览器
360安全中心
Accessibility
Accessories
Administrative Tools
Adobe Photoshop CS6(64 Bit)
HUAWEI
JetBrains
Maintenance
Microsoft Office Tools
StartUp
System Tools
www.huamaodashu.com

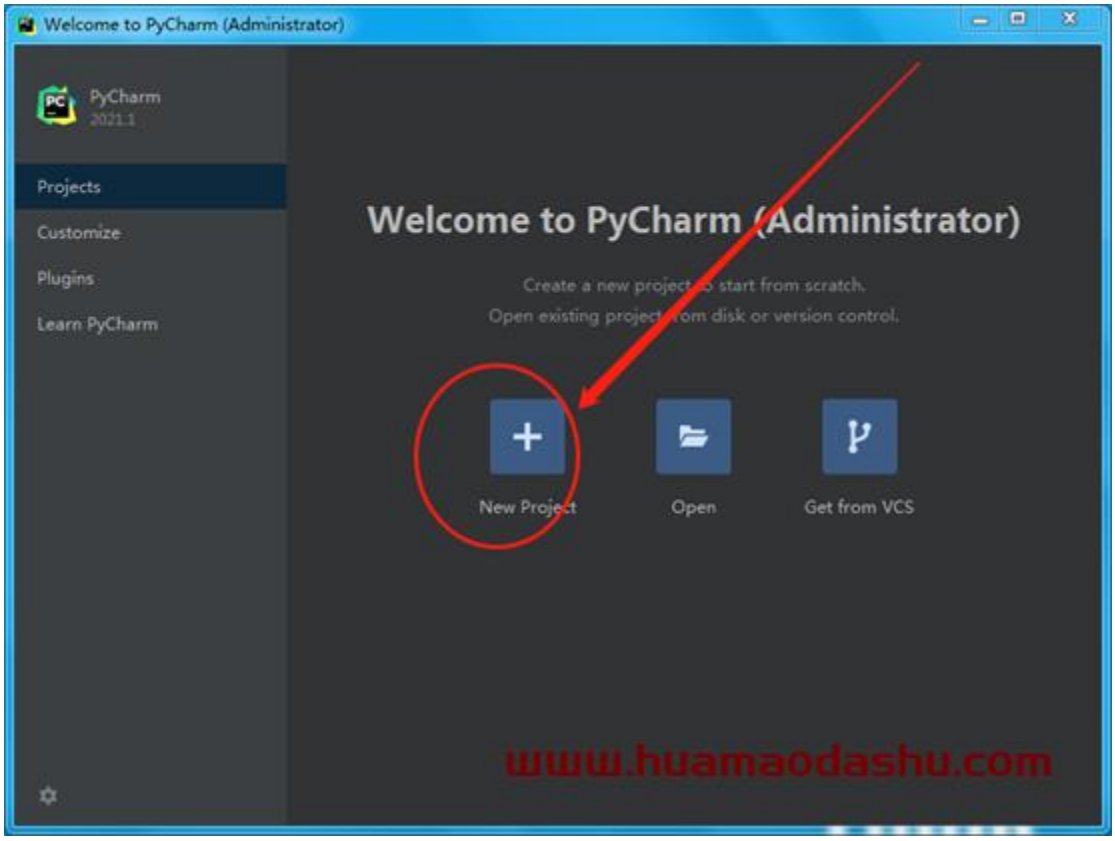

PyCharm Setup
Installing
Please wait while PyCharm is being installed.
Extract: python.jar... 57%
Show details
www.huamaodashu.com

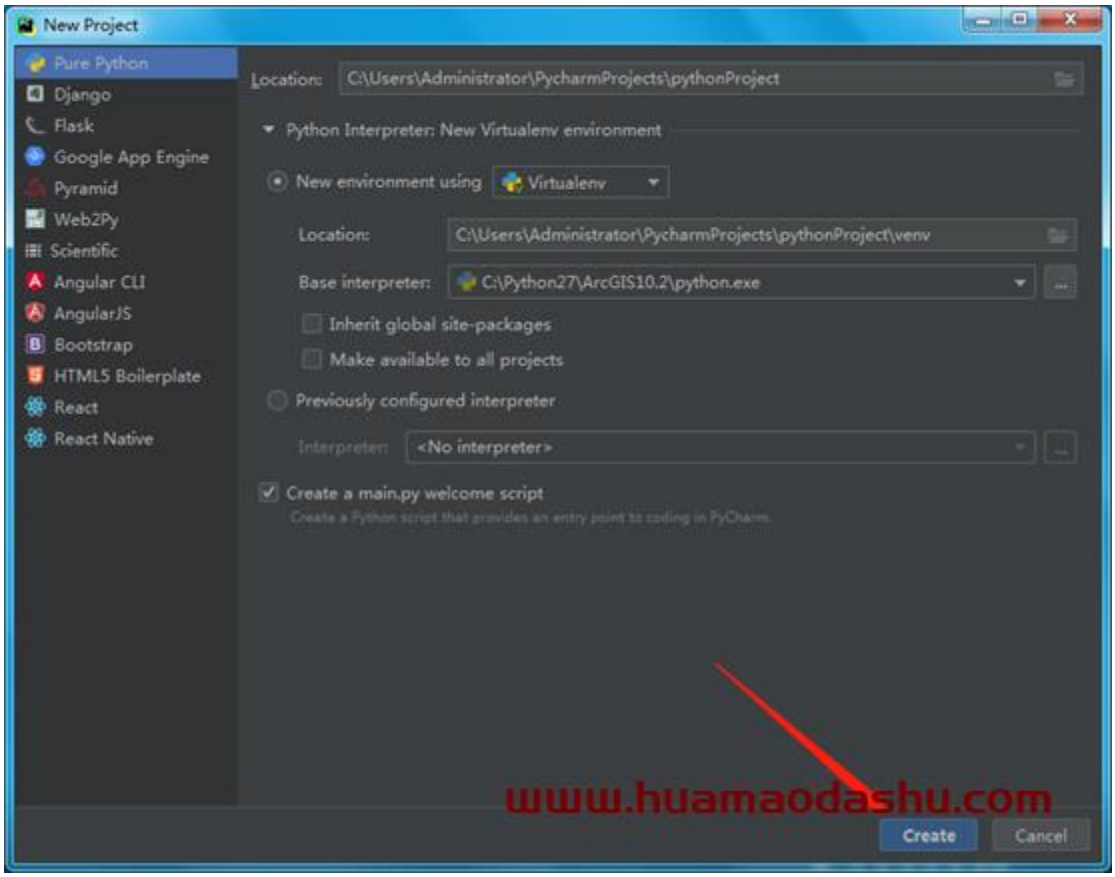

PyCharm Setup
Completing PyCharm Setup
PyCharm has been installed on your computer.
Click Finish to close Setup.
Run PyCharm
www.huamaodashu.com

双击桌面的图标，得到如下界面，如果没有继续往下看

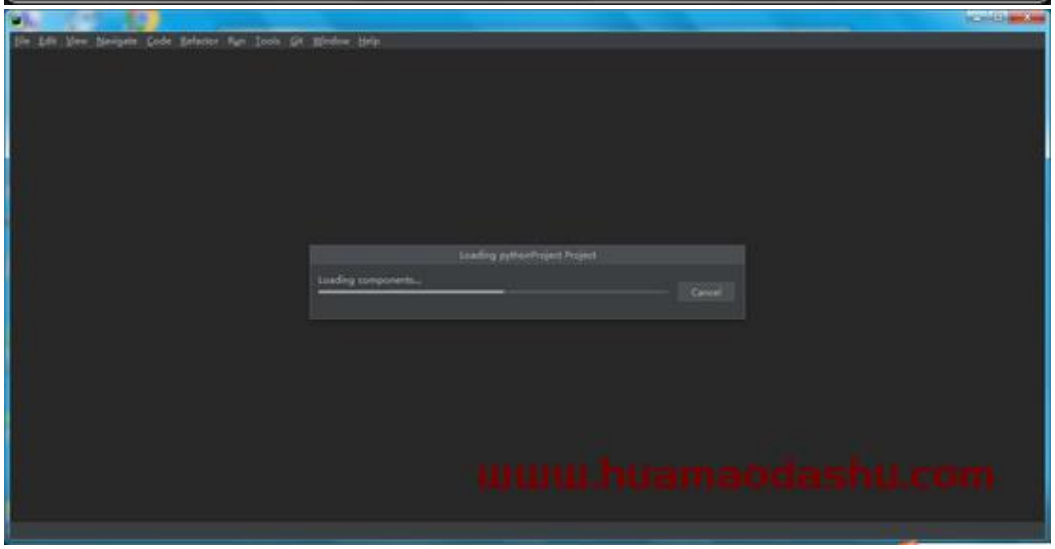


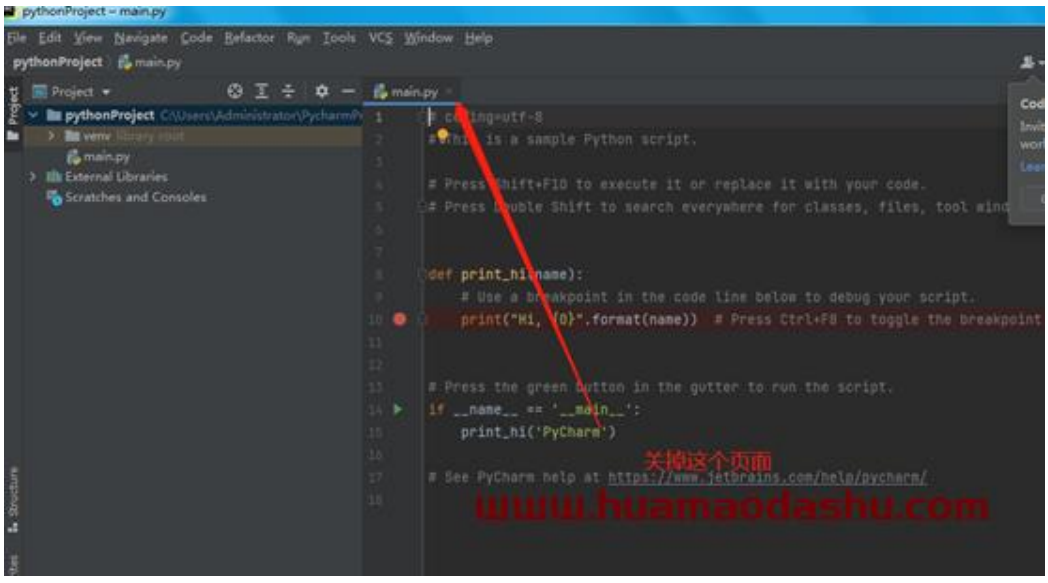

License Activation
PyCharm
Activate
Plugins
Code With Me
Activate to enable
Activate PyCharm
Evaluate for free
Buy license
Evaluation is free for 30 days.
Send me tips to learn PyCharm faster
Tell me about new product features as they come out
Email address
By subscribing to the newsletters, you agree to the privacy policy
Evaluate
www.huamaodashu.com
Continue
Exit
License Activation
PyCharm
Active until 5月 20, 2021
Plugins
Code With Me
Activate to enable
Free evaluation: 30 days left.
Buy license
Remove License
Add New License
www.huamaodashu.com
Activate plugin to enable paid functions
Continue
Exit


Welcome to PyCharm (Administrator)
PyCharm
2021.1
Projects
Customize
Plugins
Learn PyCharm
Welcome to PyCharm (Administrator)
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
www.huamaodashu.com


New Project
Pure Python
Django
Flask
Google App Engine
Pyramid
Web2Py
Scientific
Angular CLI
AngularJS
Bootstrap
HTML5 Boilerplate
React
React Native
Location: C:\Users\Administrator\PycharmProjects\pythonProject
Python Interpreter: New Virtualenv environment
New environment using Virtualenv
Location: C:\Users\Administrator\PycharmProjects\pythonProject\venv
Base interpreter: C:\Python27\ArcGIS10.2\python.exe
Inherit global site-packages
Make available to all projects
Previously configured interpreter
Interpreter: <No interpreter>
Create a main.py welcome script
Create a Python script that provides an entry point to coding in PyCharm
www.huamaodashu.com
Create
Cancel


Tip of the Day
To quickly add code cells in Jupyter notebooks, press Shift+Alt+A to insert a new cell above the currently selected cell, or Shift+Alt+B to create a new cell below the selected cell.
Don't show tips
Previous Tip
Next Tip
Close
www.huamaodashu.com


www.huamaodashu.com


pythonProject – main.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject
main.py
Project
pythonProject
venv
main.py
External Libraries
Scratches and Consoles
# This is a sample Python script.
# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.
def print_hi(name):
# Use a breakpoint in the code line below to debug your script.
print("Hi, {0}".format(name)) # Press Ctrl+F8 to toggle the breakpoint.
# Press the green button in the gutter to run the script.
if __name__ == '__main__':
print_hi('PyCharm')
# See PyCharm help at https://www.jetbrains.com/help/pycharm/
关掉这个页面
www.huamaodashu.com

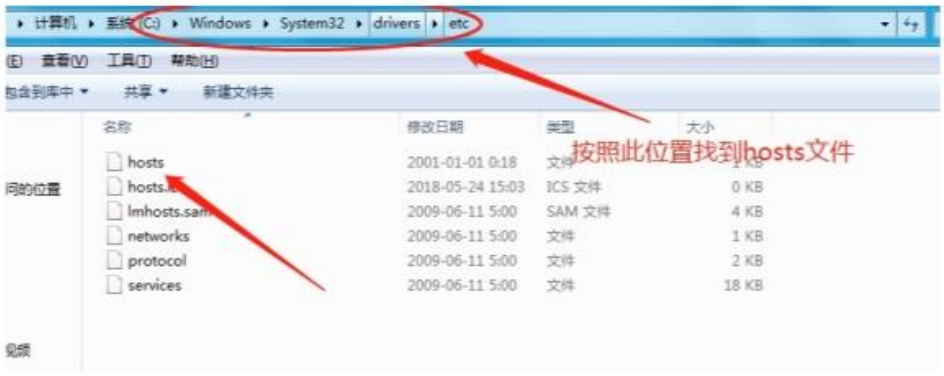
1.2.3. Pycharm 破解

激活步骤 修改 hosts 文件→拖插件（拖完插件务必重启）→复制粘贴补丁

一、 修改 hosts 文件

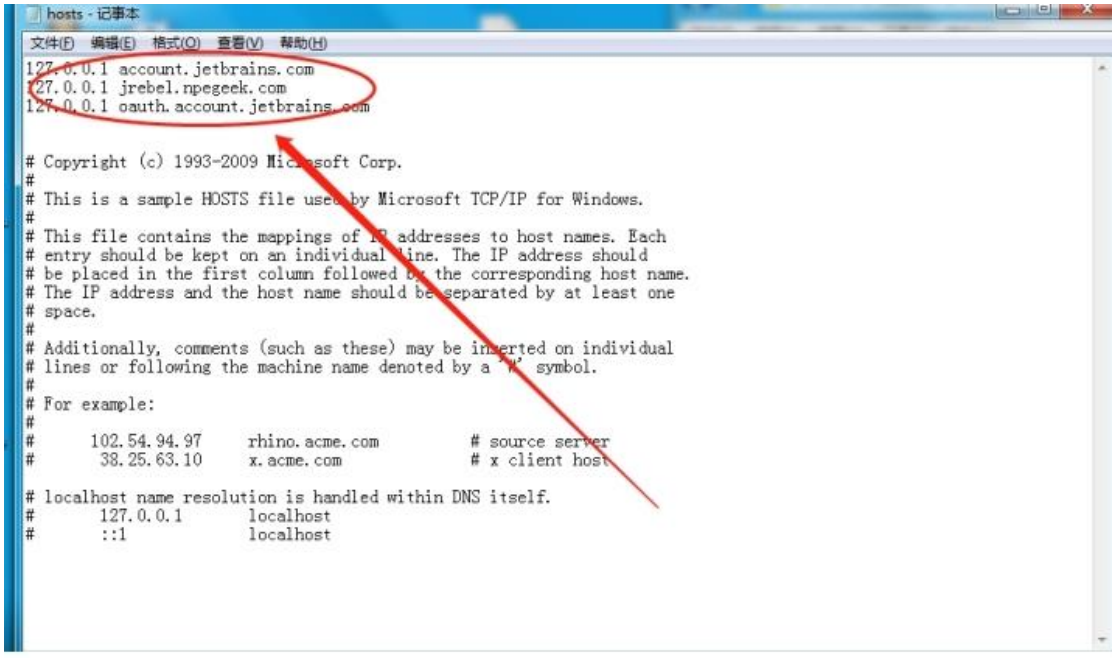
Win

按照以下位置 hosts 文件并修改，如果再此文件修改不了，请将 hosts 文件托到桌面，修改完再拖回来



hosts 文件添加以下内容并保存

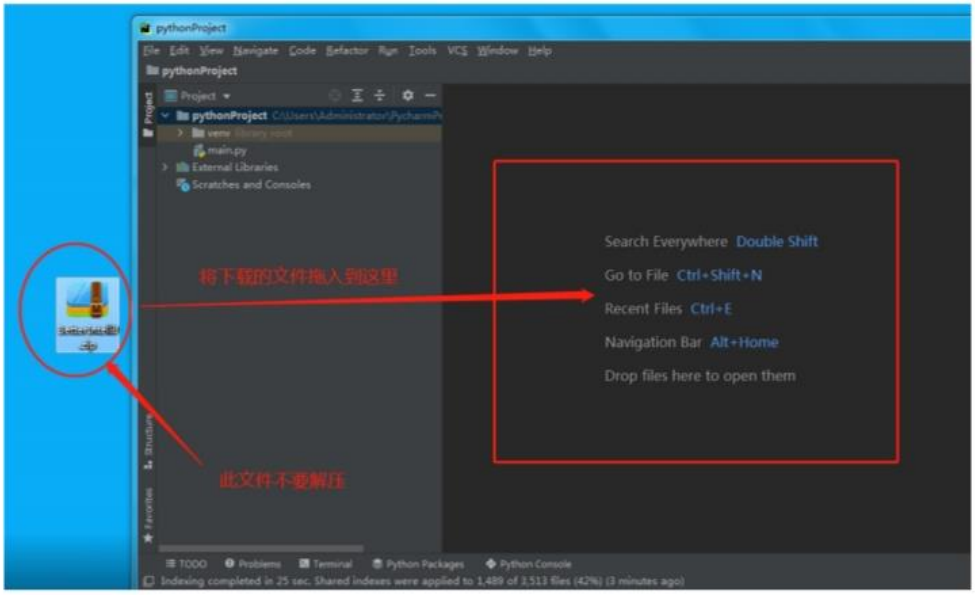
```
127.0.0.1 account.jetbrains.com

127.0.0.1 jrebel.npegeek.com

127.0.0.1 oauth.account.jetbrains.com
```



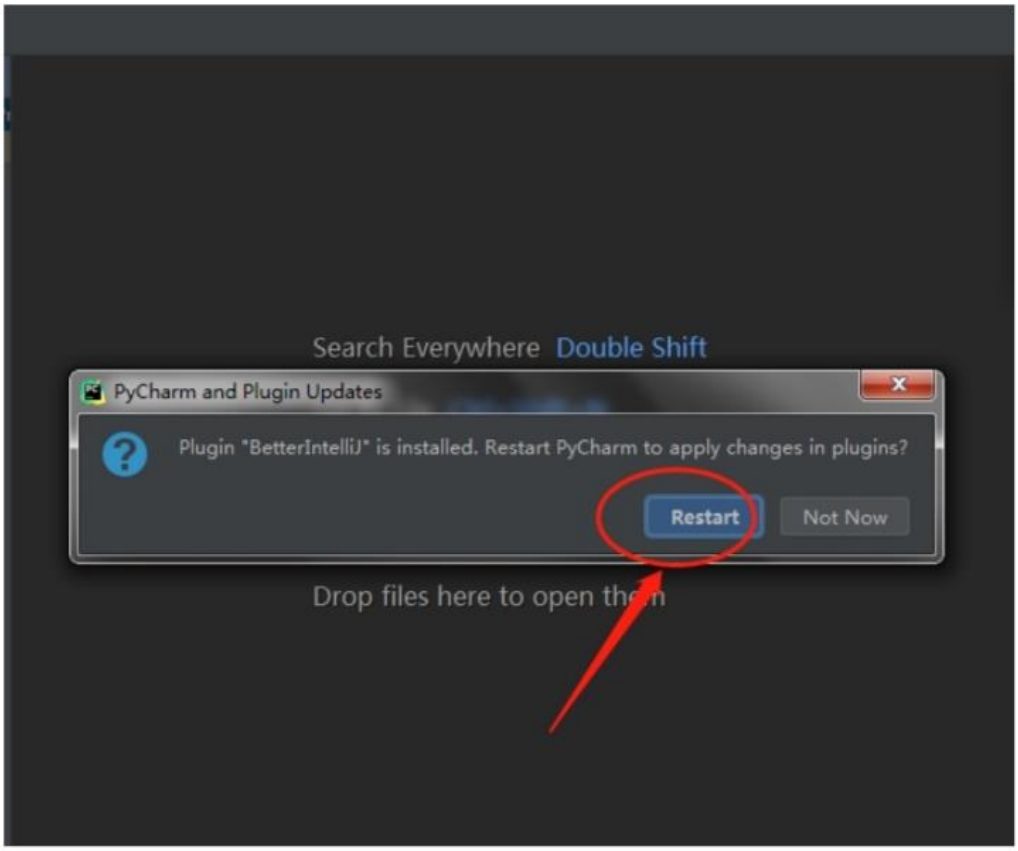
二、拖插件（拖完插件务必重启）

将下载的文件 BetterIntelliJ.zip（不要解压）拖入如下界面

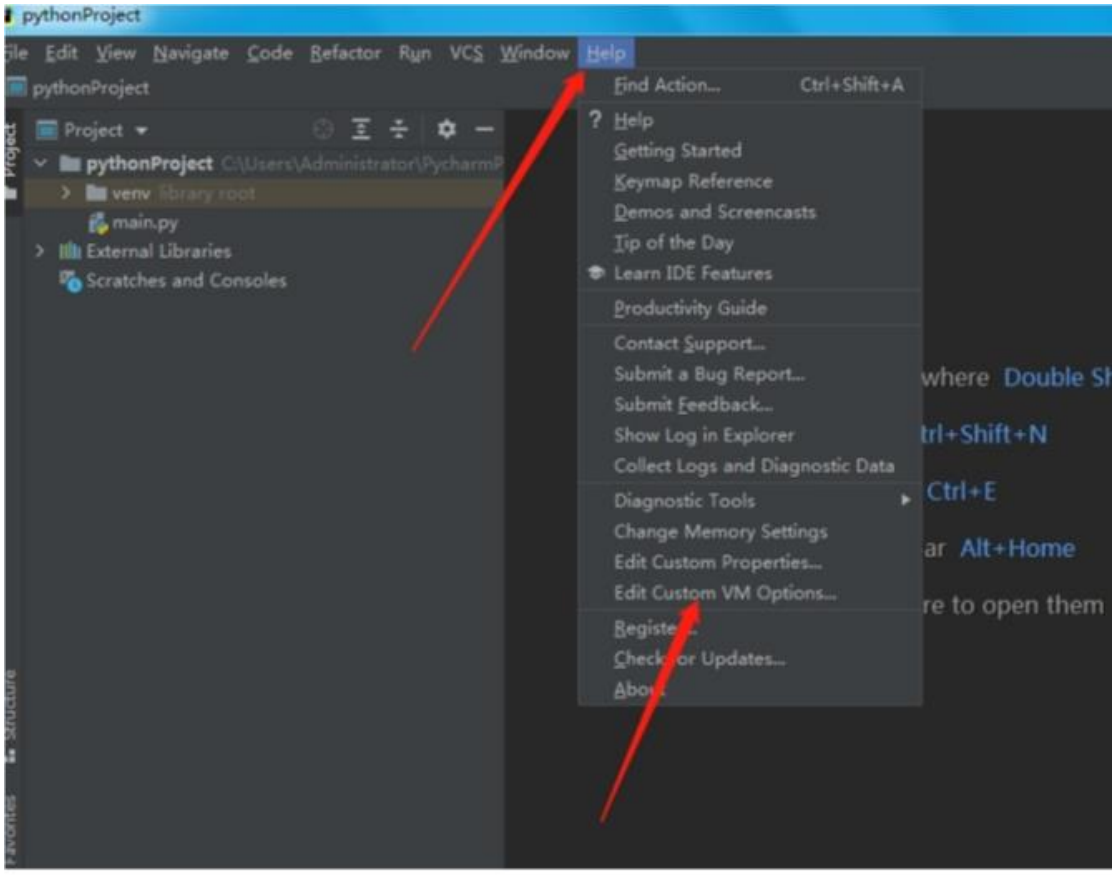
如果不是此界面请新建一个新项目



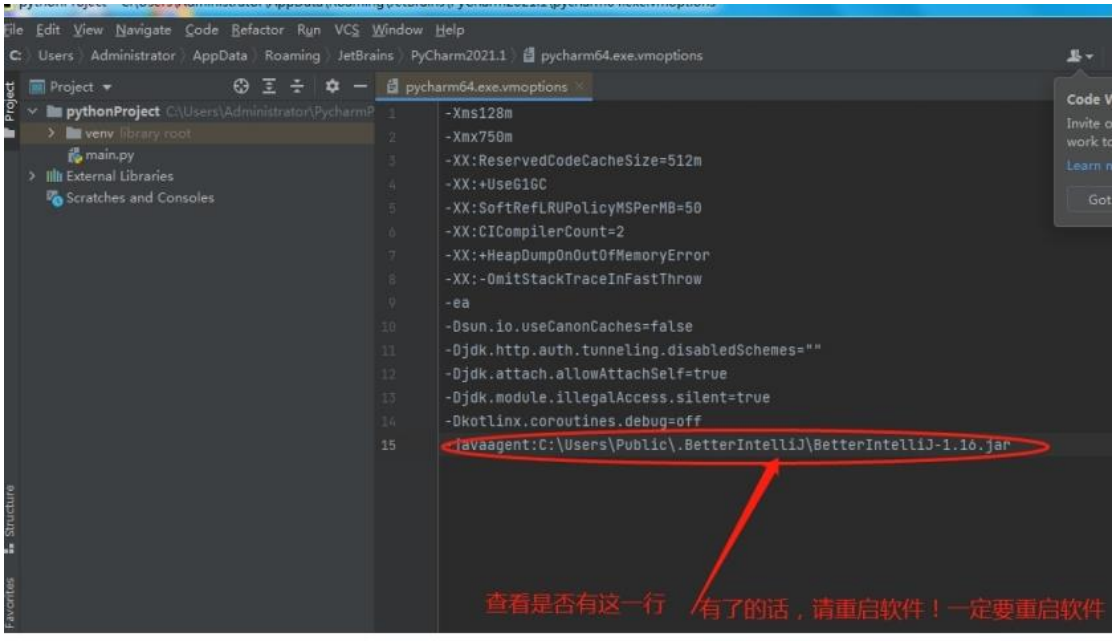
如果没有出现 restart，请重新拖



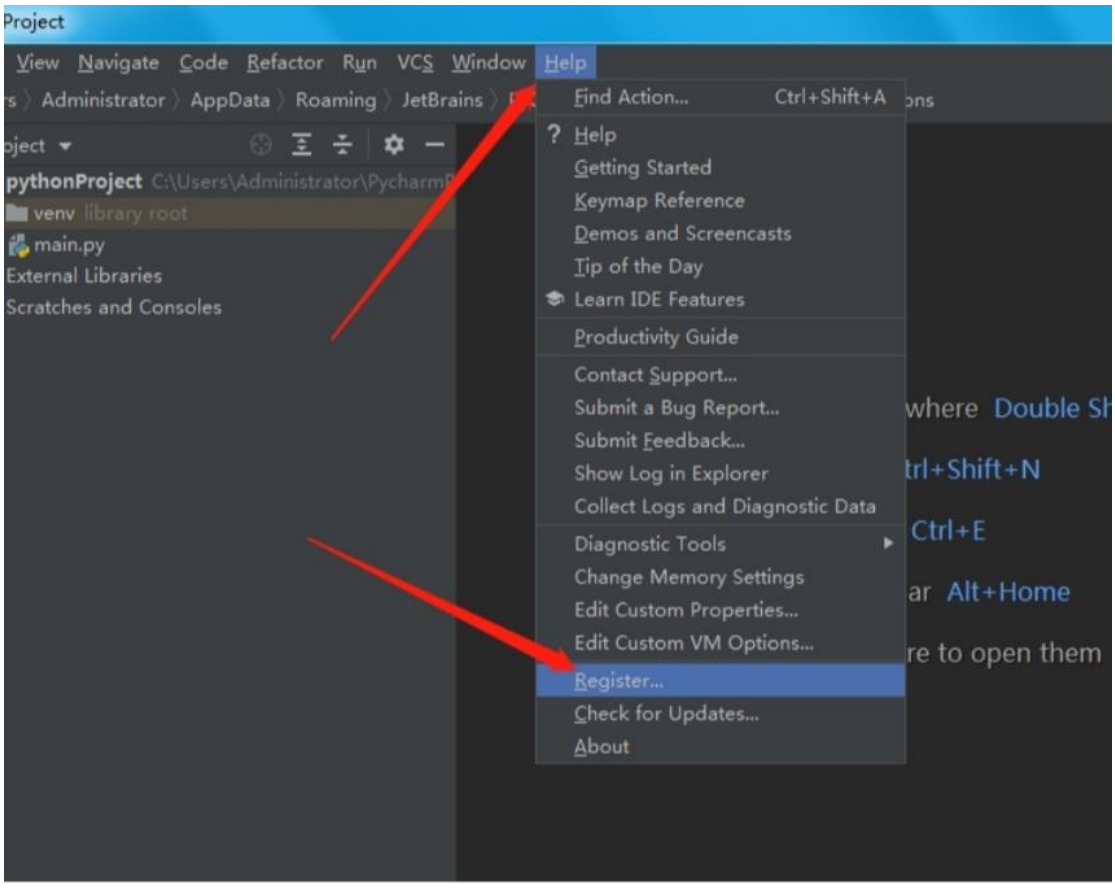
点击 help edit custom vm options 查看是否拖进来

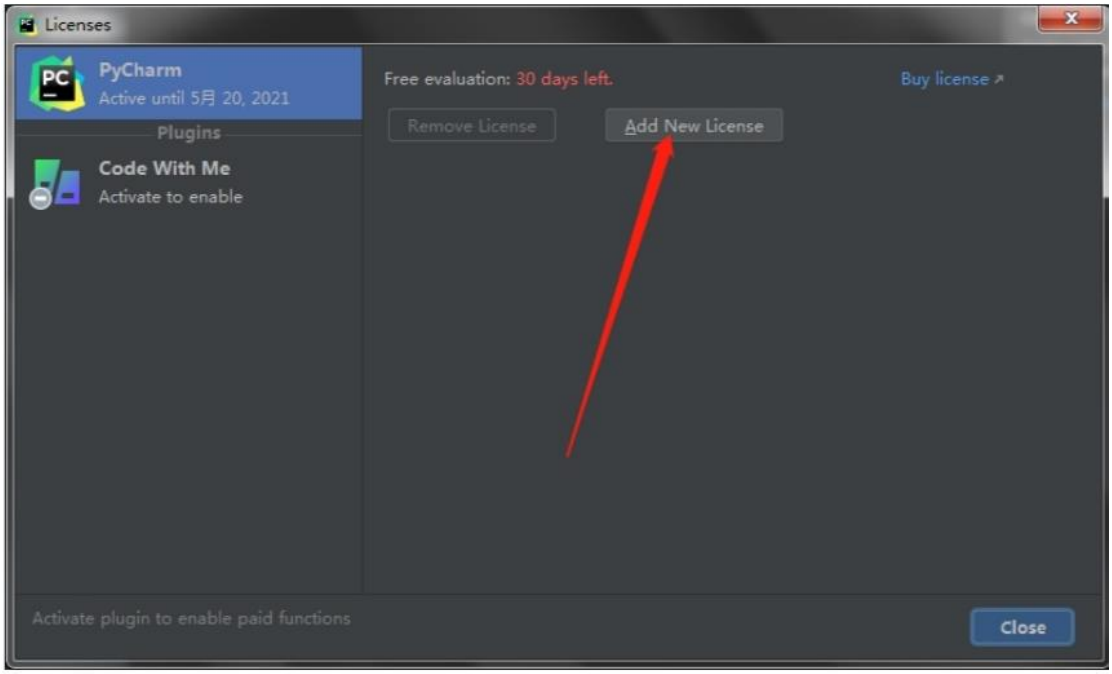


如果有，请重启软件

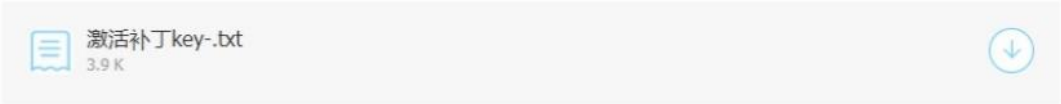


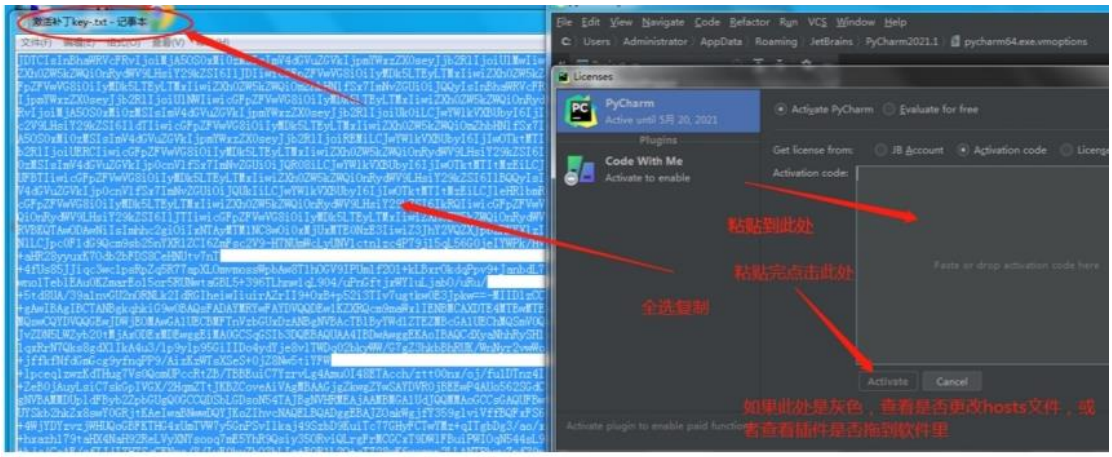
三、复制粘贴补丁



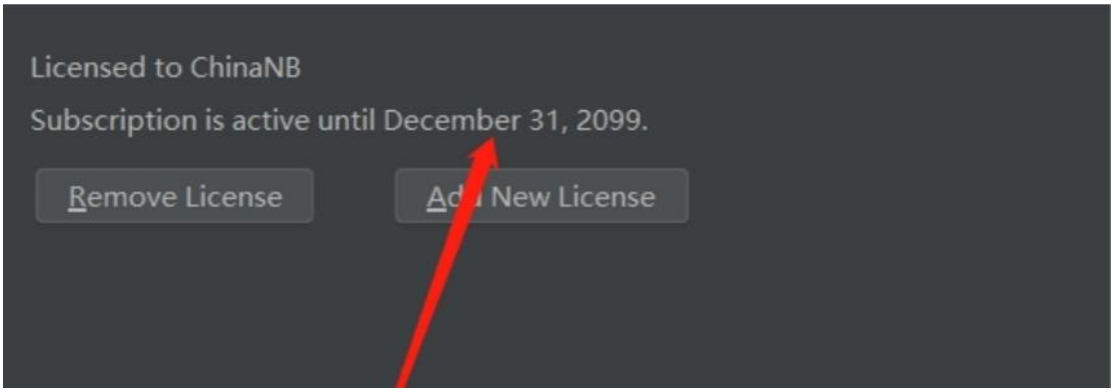


一定要打开激活补丁，复制粘贴过去

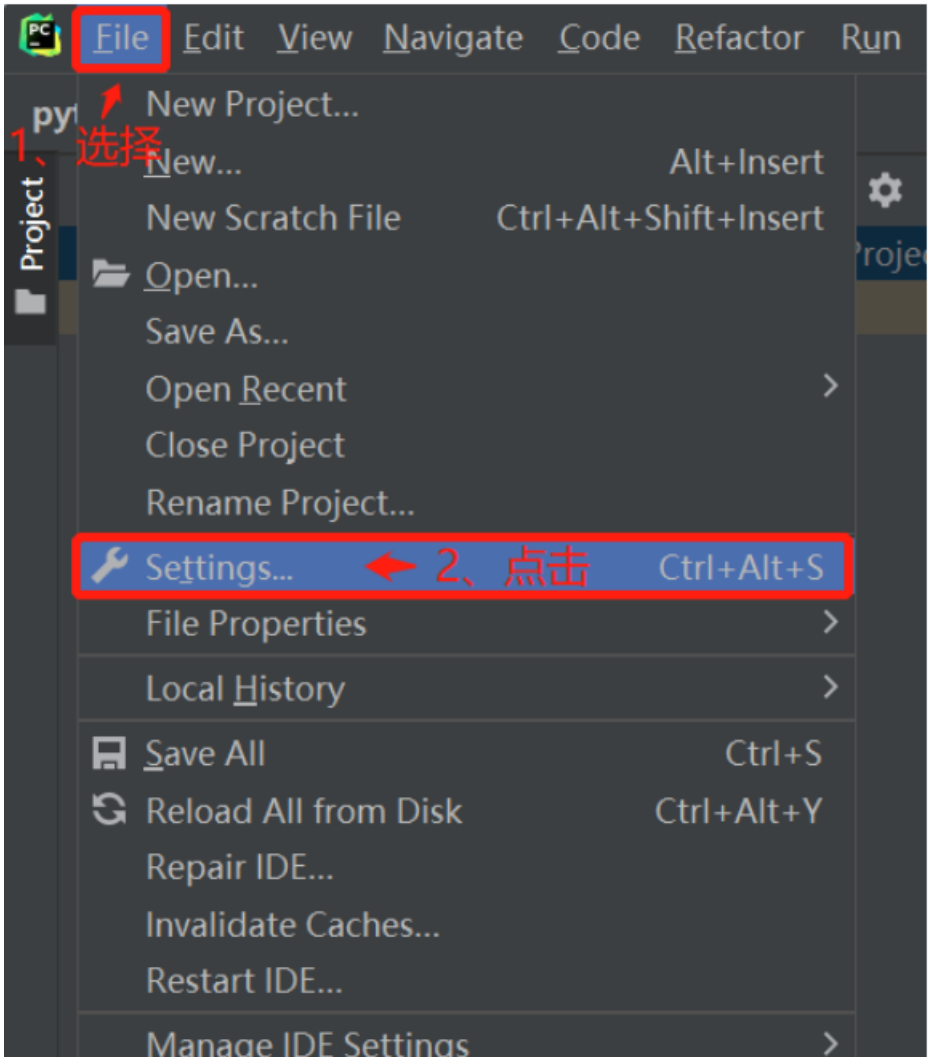


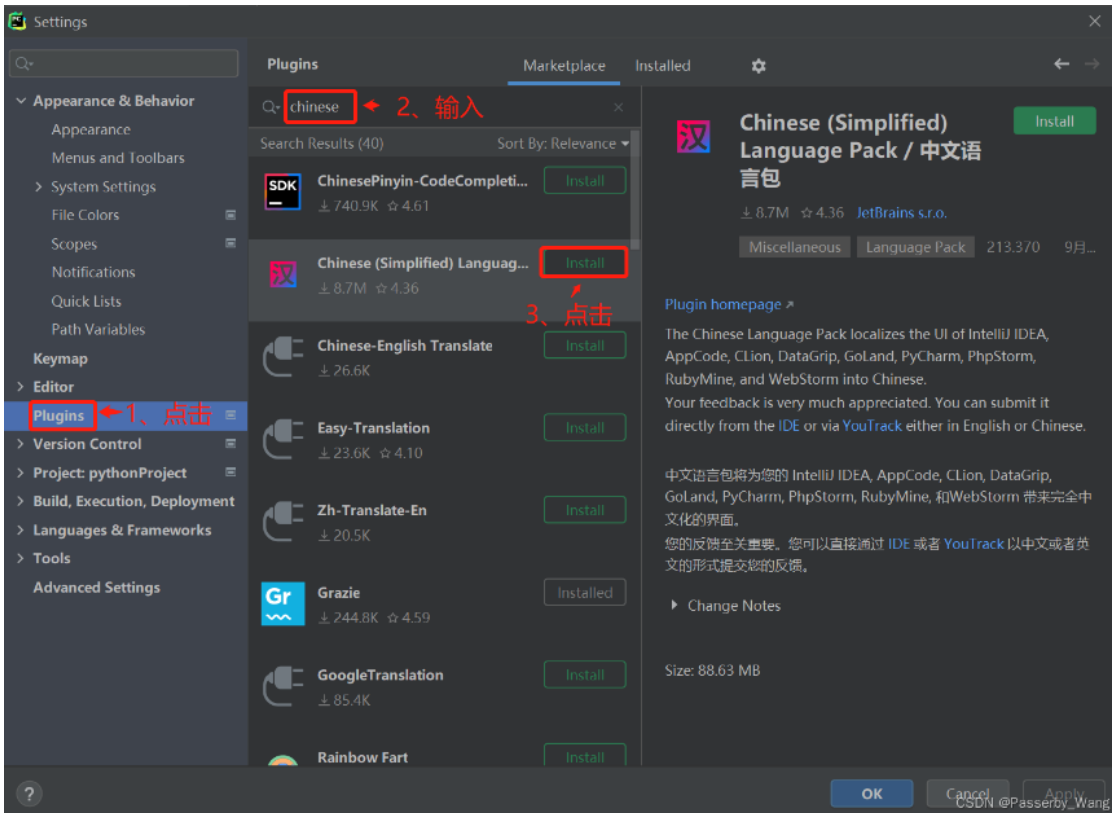

粘贴到此处
粘贴完点击此处
全选复制
如果此处是灰色，查看是否更改hosts文件，或者查看插件是否拖到软件里

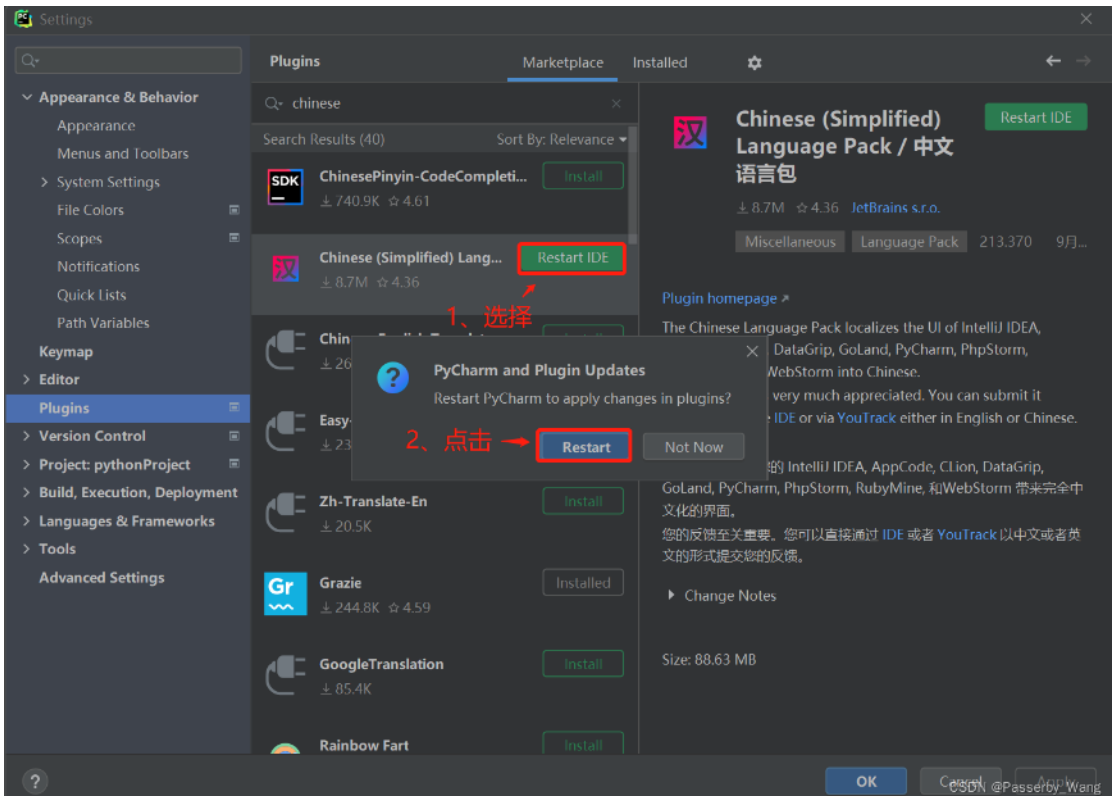
至此就激活就完成了


Licensed to ChinaNB
Subscription is active until December 31, 2099.
Remove License

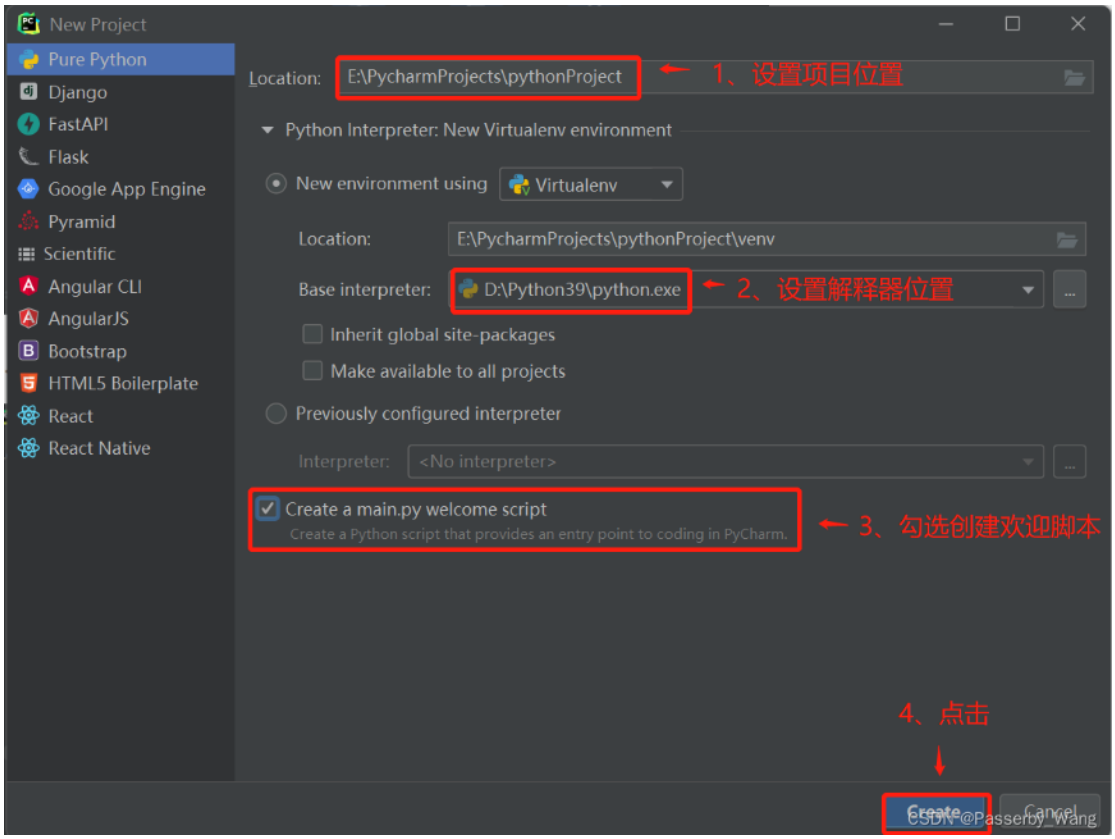
1.2.4. Pycharm 汉化

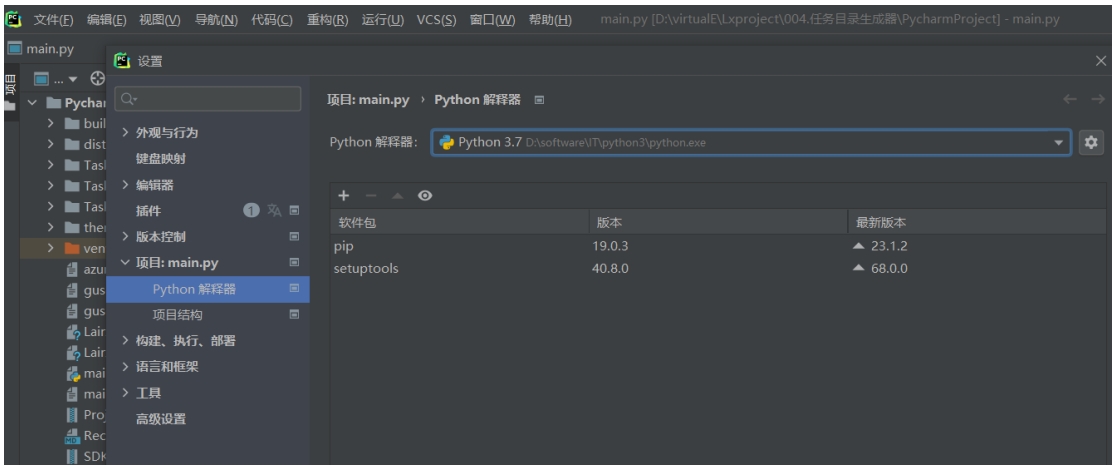



Settings
Plugins
Marketplace
Installed
chinese
2、输入
Search Results (40)
Sort By: Relevance
ChinesePinyin-CodeCompleti...
740.9K ☆4.61
Install
Chinese (Simplified) Languag...
8.7M ☆4.36
Install
3、点击
Chinese-English Translate
26.6K
Easy-Translation
23.6K ☆4.10
Zh-Translate-En
20.5K
Grazie
244.8K ☆4.59
Installed
GoogleTranslation
85.4K
Rainbow Fart
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
File Colors
Scopes
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
1、点击
Version Control
Project: pythonProject
Build, Execution, Deployment
Languages & Frameworks
Tools
Advanced Settings
Chinese (Simplified) Language Pack / 中文语言包
Install
8.7M ☆4.36 JetBrains s.r.o.
Miscellaneous
Language Pack
213.370
Plugin homepage
The Chinese Language Pack localizes the UI of IntelliJ IDEA, AppCode, CLion, DataGrip, GoLand, PyCharm, PhpStorm, RubyMine, and WebStorm into Chinese.
Your feedback is very much appreciated. You can submit it directly from the IDE or via YouTrack either in English or Chinese.
中文语言包将为您的 IntelliJ IDEA, AppCode, CLion, DataGrip, GoLand, PyCharm, PhpStorm, RubyMine, 和WebStorm 带来完全中文化的界面。
您的反馈至关重要。您可以直接通过 IDE 或者 YouTrack 以中文或者英文的形式提交您的反馈。
Change Notes
Size: 88.63 MB
OK
Cancel
Apply
CSDN @Passerby_Wang


Settings
Plugins
Marketplace
Installed
chinese
Search Results (40)
Sort By: Relevance
ChinesePinyin-CodeCompleti...
740.9K ☆4.61
Install
Chinese (Simplified) Lang...
8.7M ☆4.36
Restart IDE
1、选择
PyCharm and Plugin Updates
Restart PyCharm to apply changes in plugins?
2、点击
Restart
Not Now
Zh-Translate-En
20.5K
Grazie
244.8K ☆4.59
Installed
GoogleTranslation
85.4K
Rainbow Fart
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
File Colors
Scopes
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Version Control
Project: pythonProject
Build, Execution, Deployment
Languages & Frameworks
Tools
Advanced Settings
Chinese (Simplified) Language Pack / 中文语言包
Restart IDE
8.7M ☆4.36 JetBrains s.r.o.
Miscellaneous
Language Pack
213.370
Plugin homepage
DataGrip, GoLand, PyCharm, PhpStorm,
WebStorm into Chinese.
very much appreciated. You can submit it
IDE or via YouTrack either in English or Chinese.
的 IntelliJ IDEA, AppCode, CLion, DataGrip, GoLand, PyCharm, PhpStorm, RubyMine, 和WebStorm 带来完全中文化的界面。
您的反馈至关重要。您可以直接通过 IDE 或者 YouTrack 以中文或者英文的形式提交您的反馈。
Change Notes
Size: 88.63 MB
OK
Cancel
Apply
CSDN @Passerby_Wang

1.2.5. Pycharm 使用





1.3 Anadonca

1.3.1. Anadonca 简介

Anaconda 是 Python 的一个开源发行版本，包含了 Python 环境管理工具 conda，以及很多科学计算包，如 numpy, scipy 等等.

Python 环境：对应一个目录和相应的配置信息(环境变量等)，目录里包含了需要的 Python 库文件. 在 Python 代码中使用某个库，需要提前安装好. 利用 conda 管理环境非常方便，可以在一台主机上创建多个环境，使用时只要激活相应的环境即可.

1.3.2. Anadonca 安装

不展开说明

1.4 可视化插件 VisualTkinter

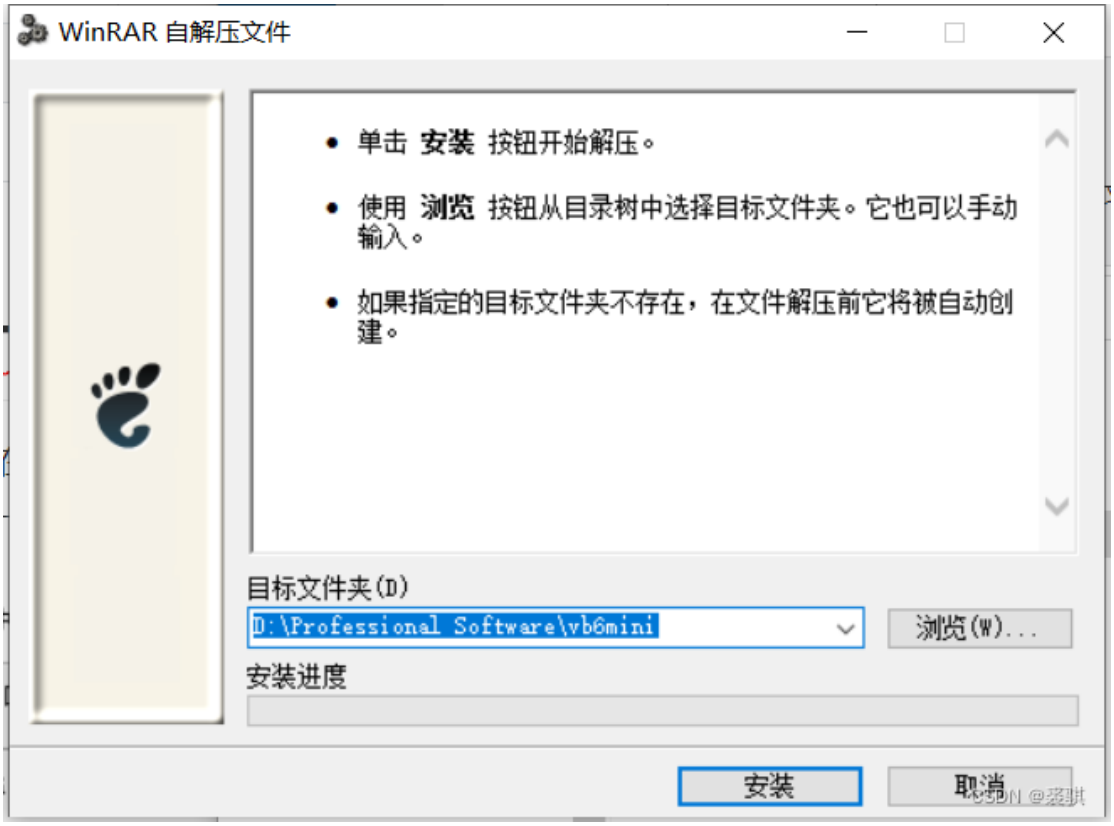
1.4.1. VisualTkinter 简介

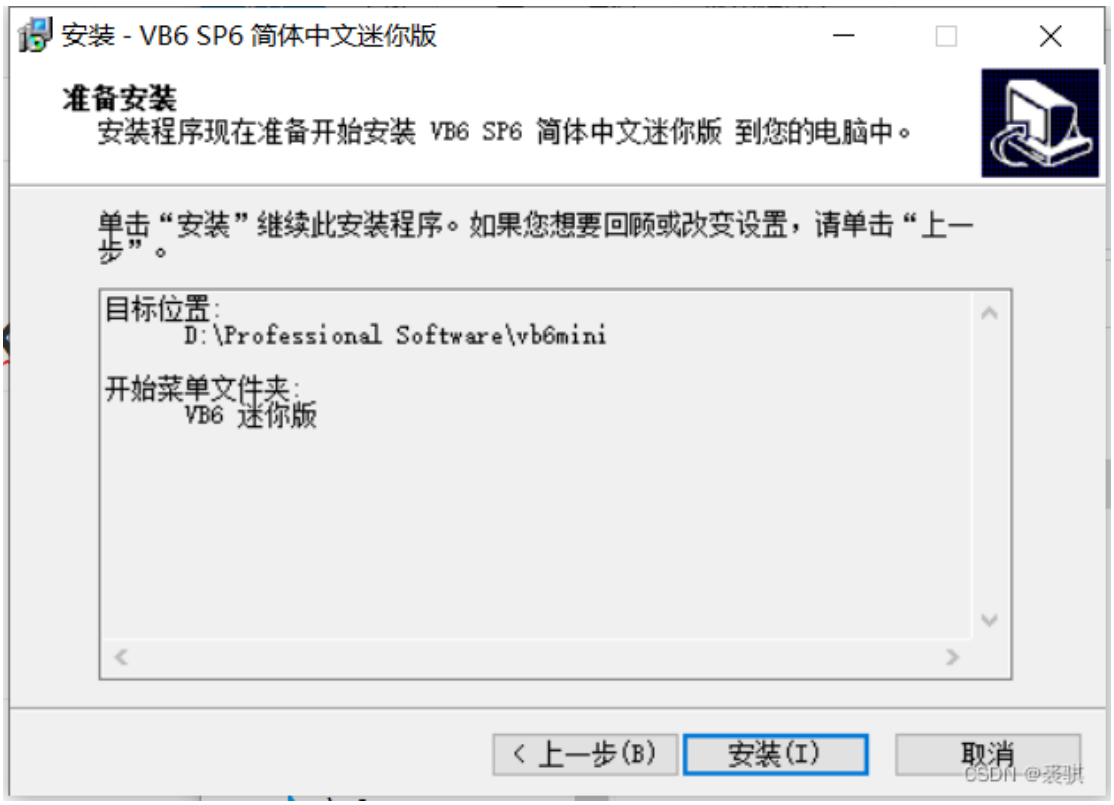
Visual Tkinter 是一个可以让用户在 Visual Basic 6 中设计界面，然后将转换为 Python 的代码的方便工具。

1.4.2. VisualTkinter 安装

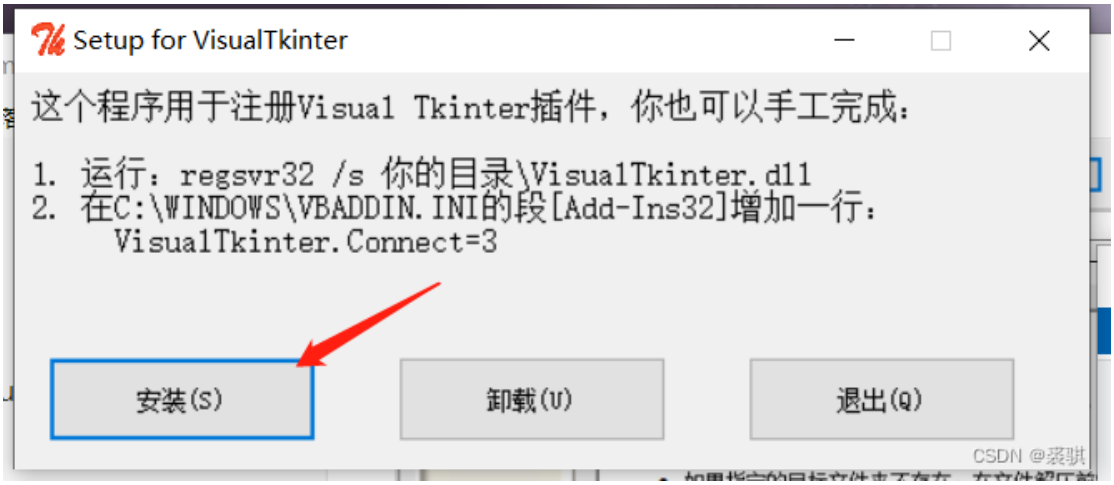
软件获取链接:链接：https://pan.baidu.com/s/1amd1GRaP5lpc7a9nJr1aNQ
提取码：1023

使用 VisualTkinter 需要安装 vb6 安装和 vb 控件（建议装在非 c 盘同一目录

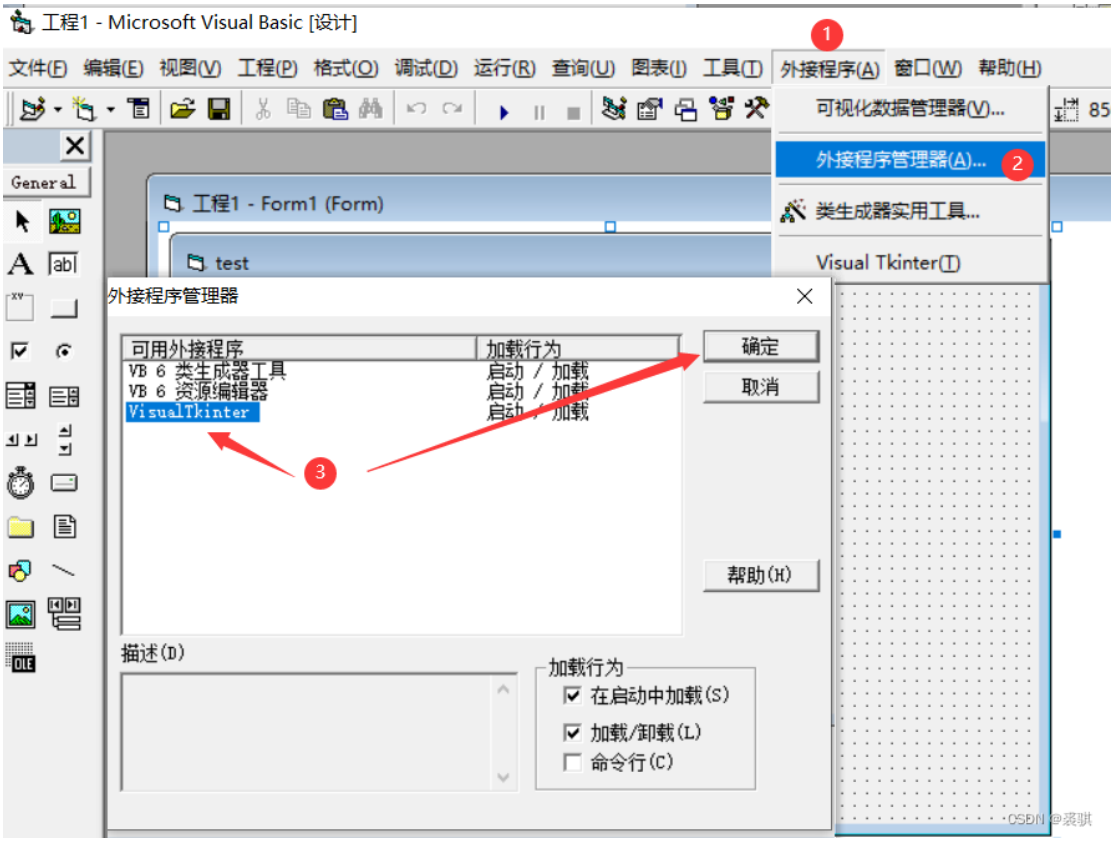




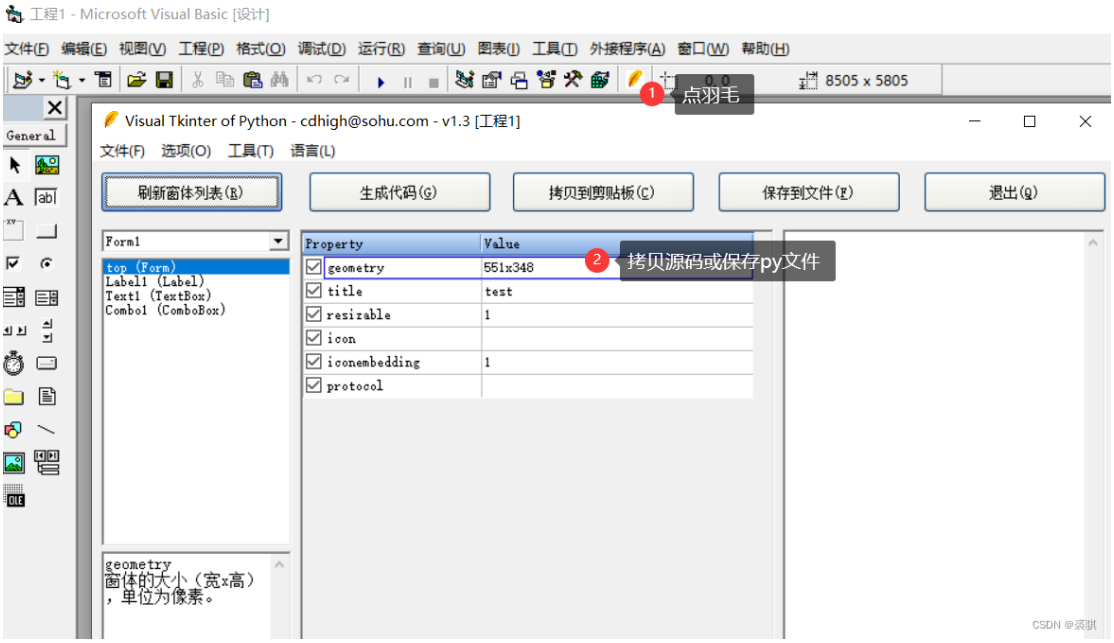
安装外接程序插件 Visual Tkinter


Setup for VisualTkinter
这个程序用于注册Visual Tkinter插件，你也可以手工完成：
1. 运行：regsvr32 /s 你的目录\VisualTkinter.dll
2. 在C:\WINDOWS\VBADDIN.INI的段[Add-Ins32]增加一行：
VisualTkinter.Connect=3
安装(S)
卸载(U)
退出(Q)

在 vb6 添加外接程序


工程1 - Microsoft Visual Basic [设计]
文件(F) 编辑(E) 视图(V) 工程(P) 格式(O) 调试(D) 运行(R) 查询(U) 图表(I) 工具(T) 外接程序(A) 窗口(W) 帮助(H)
可视化数据管理器(V)...
外接程序管理器(A)...
类生成器实用工具...
Visual Tkinter(T)
General
工程1 - Form1 (Form)
test
外接程序管理器
可用外接程序
加载行为
VB 6 类生成器工具
VB 6 资源编辑器
VisualTkinter
确定
取消
帮助(H)
描述(D)
加载行为
在启动中加载(S)
加载/卸载(L)
命令行(C)
1
2
3

1.4.3. VisualTkinter 使用



第2章 功能说明与软件框架

2.1 软件功能说明

2.1.1. 功能概述

Lairx4 Total Directory Report，简称 Lx4-TDR，是一个用于自动整合命名，快速生成任务目录和记录文档并解压 SDK 的工具。

本工具最基本功能是自动识别日期，根据填写信息合规范命名并生成目录结构，用户可使用 cfg 文件设置序号，新增芯片和代码版本，此外还具备 eso 模式、菜单栏快速导航、切换主题等其他功能。

2.1.2. 功能列表

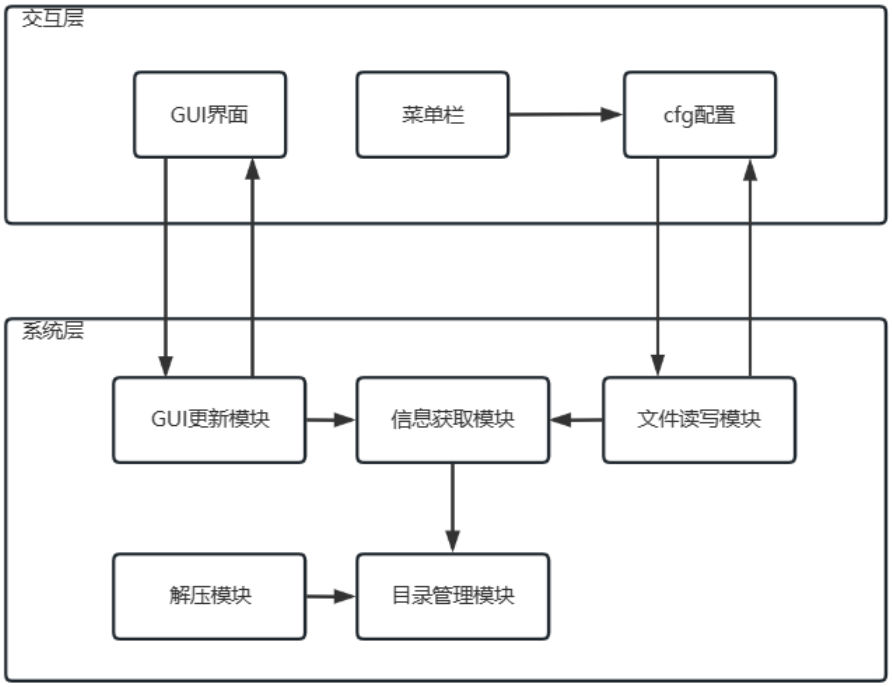
序号	功能	说明
1	获取系统时间	用于生成统一规范项目名
2	提示输入信息	GUI 通过输入框、下拉框勾选框引导用户输入信息用于生成统一规范项目名
3	ESO 模式	内部模式，使用后项目名为另一组序号 0x
4	序号自加	自动记录序号，生成新项目目录后自加一
5	CFG 配置	用户可通过外部 cfg 文档配置工具
6	结构生成	在工具目录下自动生成文件结构
7	主题	明和暗主题切换
8	快捷导航	快速打开任务目录、任务文档、源 SDK
9	帮助文档	帮助和关于提示框信息

生成目录

序号	功能	说明
1	自动合成项目名	生成统一规范项目名
2	模板创建目录	根据模板创建单项目录文件
3	模板创建文档	根据模板创建单项目录对应文档
4	复制对应 SDK	在 SDK 源解压对应 SDK
5	一键生成	一键合成规范命名，生成目录和文档，并复制对应 SDK
6	自动打开生成目录	目录生成后自动收起工具，打开目录文件夹

2.2 应用整体框架

应用分为交互层（前端）和系统层（后台）

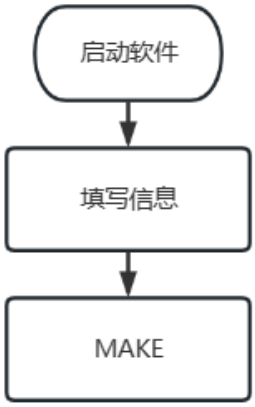

交互层
GUI界面
菜单栏
cfg配置
系统层
GUI更新模块
信息获取模块
文件读写模块
解压模块
目录管理模块

2.3 工程目录结构

```
工程目录工具文件夹

- 任务目录生成器.exe

- guser_config.txt（cfg配置文件）

- ProjectTemplate.zip（工程模板压缩包）

-RecordTemplate.md

- TaskDirectory（任务目录）
  • 0x1.20230410_700v201_同事A_一个标题
  • 0x2.20230410_700v201_同时B_一个标题
  • 1.20230410_700v201_新闻达_一个标题
  • 2.20230410_700v201_新闻达_一个标题

- TaskRecord（任务记录）
  • 1.20230410_700v201_新闻达_一个标题.md
  • 1.20230410_700v201_新闻达_一个标题.md

- TaskSourceSDK（SDK源库）
  • AC696N
    -101
    696V101.zip
   -102
    696V102.zip
  • AC702N
    -101
    701v201.zip
   -102
    701v202.zip
```

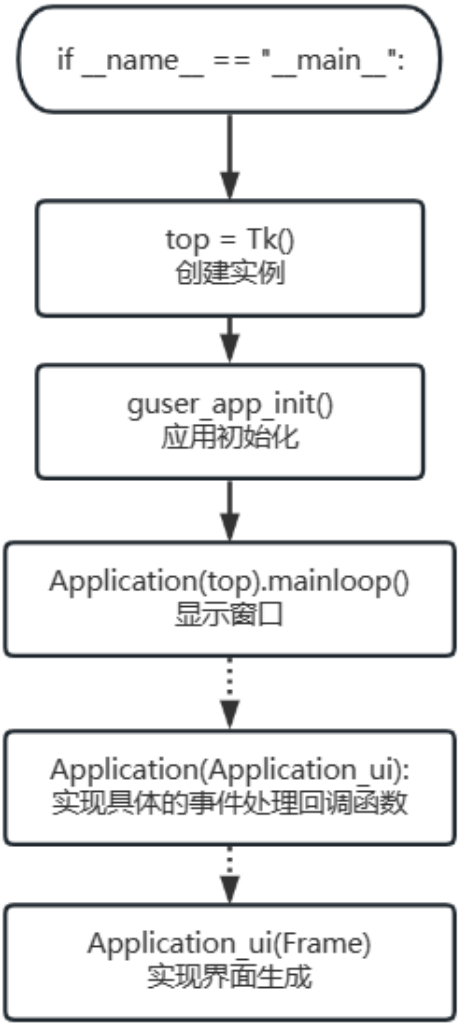
2.4 用户使用说明

(1) 启动软件
(2) 在 GUI 界面的输入框、下拉框、勾选框中填写信息
(3) 在预览框中确认目录标题
(4) 按 MAKE 生成目录并自动打开



2.5 软件流程

启动流程如图所示:



```
if __name__ == "__main__":
    top = Tk()                                  # 实例化一个Tk窗口对象
    guser_app_create_menu()                     # 创建菜单栏

    # top.wm_attributes("-topmost", True)       # 设置GUI置顶
    top.wm_attributes("-alpha", 0.85)           # 设置GUI透明度(0.0~1.0)
    # top.attributes("-alpha", 0.9)             # 设置GUI透明度(0.0~1.0)

    si = Sizegrip(top, style='1.TSizegrip')     # 右下角定位三角
    si.pack(side=BOTTOM, anchor=SE)

    top.tk.call("source", "azure.tcl")          # 导入主题资源
    top.tk.call("set_theme", "dark")            # 默认主题：dark
    # top.tk.call("set_theme", "light")         # 默认主题：light
    # top.iconbitmap(icon_file) # icon_file就是一个.ico的图标文件，使用绝对或相对路径
    guser_app_init()                            # 应用初始化
    Application(top).mainloop()                 # mainloop()显示窗口
    try: top.destroy()
    except: pass
```

```
if __name__ == "__main__":
    top = Tk()                                  # 实例化一个 Tk 窗口对象
    guser_app_create_menu()                     # 创建菜单栏

    # top.wm_attributes("-topmost", True)       # 设置 GUI 置顶
    top.wm_attributes("-alpha", 0.85)           # 设置 GUI 透明度(0.0~1.0)
    # top.attributes("-alpha", 0.9)             # 设置 GUI 透明度(0.0~1.0)

    si = Sizegrip(top, style='1.TSizegrip')     # 右下角定位三角
    si.pack(side=BOTTOM, anchor=SE)

    top.tk.call("source", "azure.tcl")          # 导入主题资源
    top.tk.call("set_theme", "dark")            # 默认主题：dark
    # top.tk.call("set_theme", "light")         # 默认主题：light
    # top.iconbitmap(icon_file) # icon_file 就是一个.ico 的图标文件，使用绝对或相
对路径
    guser_app_init()                            # 应用初始化
    Application(top).mainloop()                 # mainloop()显示窗口
    try: top.destroy()
    except: pass
```

2.6 常量变量说明

命名规则:

1.cfg-外部配置相关

2.sys-内部系统相关

3.app-软件前端相关

4.serial1：日常支持，serial2：测试协助

```
cfg_data = []           # 存储外部配置
cfg_serial0 = ""        # 日常支持序号
cfg_serial1 = ""        # 测试协助序号
cfg_sdk_path = ""       # SDK获取地址，该功能暂不开放，默认路径在exe文件同级目录
cfg_target_path = ""    # 目录生成地址，该功能暂不开放，默认路径在exe文件同级目录
sys_time = ""           # init读取的系统时间
sys_theme = 0           # 系统主题 # 暂时不存在cfg中，会导致每次打开都是默认主题
sys_ui_reset_flag = 0   # UI复位标志
app_mode = "0"         # 模式0：普通模式，模式1：eso模式，内部模式，测试协助模式
# 系统存储芯片系列的列表，获取cfg信息后会把这里覆盖掉
sys_series_list = ['AC695N', 'AC696N', 'AC697N', 'AD698N', 'AC700N', 'JL701N', 'AC702N']
# 系统存储芯片系列对应SDK版本的列表，与sys_series_list对应
sys_version_list = [['100', '101', '102'],
                    ['200', '201', '202'],
                    ['300', '301', '302'],
                    ['400', '401', '402'],
                    ['500', '501', '502'],
                    ['600', '601', '602'],
                    ['700', '701', '702']]
```

```
# 文件目录名配置
task_directory_path = "TaskDirectory"
task_record_path = "TaskRecord"
task_source_sdk_path = "TaskSourceSDK"
app_title = "Lairx4 Total Directory Report v2.0"    # 窗口标题

sys_cfg_path = "guser_config.txt"  # 存储外部配置的txt文件
sys_project_template_path = "ProjectTemplate.zip"# 工程目录模板，必须包含PublicSDK文件夹用于解压SDK
sys_record_template_path = "RecordTemplate.md"      # 工程记录模板

# cfg文件解析配置
cfg_serial0_num_line = 0
cfg_serial1_num_line = 1
cfg_series_check_line = 2
cfg_series_list_line = 3
cfg_version_check_line = 4
cfg_version_list_line = 5

# cfg文件解析校验配置
cfg_series_check_str = "series list:\n"
cfg_version_check_str = "version list:\n"
```

第3章 GUI 实现与使用

3.1 GUI 和 TKinter 简介

GUI 是 Graphical User Interface 的缩写，意为图形用户界面。它是一种人机交互界面，通过图形化的方式展示信息和操作元素，使用户可以通过鼠标、触摸屏等输入设备进行操作。GUI 提供了一种直观的方式来与计算机进行交互，使用户能够更轻松地使用各种应用程序和系统功能

Tkinter 是 Python 标准库中的一个 GUI（Graphical User Interface，图形用户界面）工具包，其目的是为 Python 开发者提供快捷创建 GUI 应用程序的方式。

Tkinter 基于 Tcl/Tk 图形库，允许我们使用 Python 代码来创建和管理窗口、标签、按钮、复选框、文本框、列表框、滚动条、画布、菜单等多种控件和组件。 Tkinter 对多数平台都有良好的支持，而无需安装额外的软件或库。

3.2 TKinter 基本使用

参考例程：

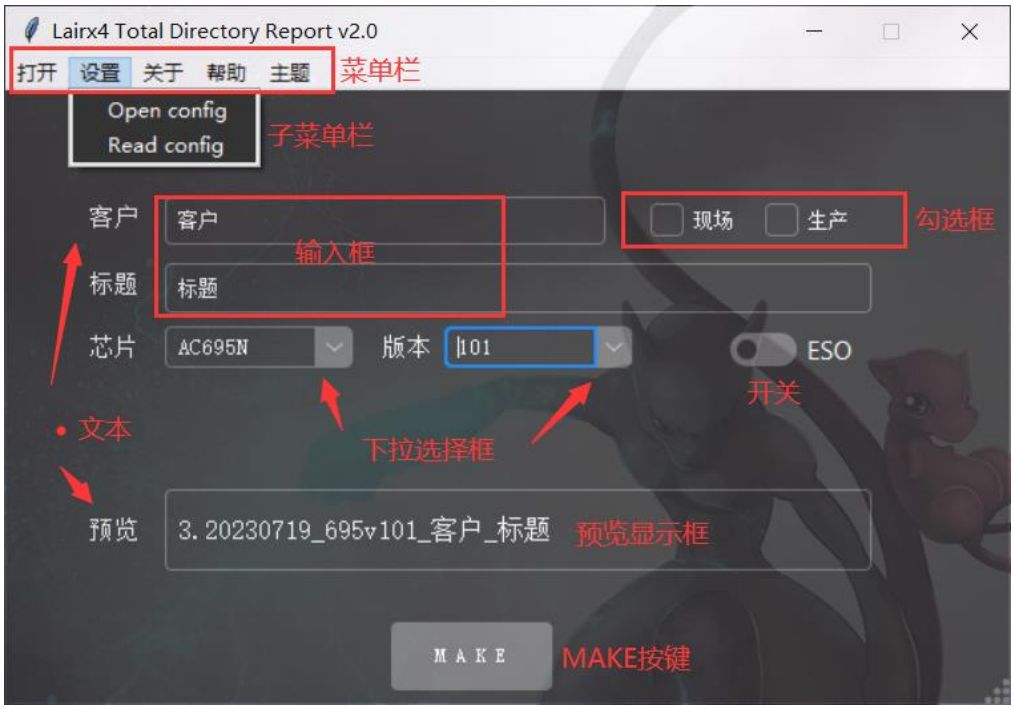
```
import tkinter as tk
from tkinter import messagebox
root = tk.Tk()  # 创建窗口
root.title('演示窗口')
root.geometry("300x100+630+80")  # (宽度 x 高度)+(x 轴+y 轴)

btn1 = tk.Button(root)  # 创建按钮，并且将按钮放到窗口里面
btn1["text"] = "点击"  # 给按钮一个名称
btn1.pack()  # 按钮布局


def test(e):
    '''创建弹窗'''
    messagebox.showinfo("窗口名称", "点击成功")


btn1.bind("<Button-1>", test)  # 将按钮和方法进行绑定，也就是创建了一个事件
root.mainloop()  # 让窗口一直显示，循环
```

参考连接：

(51 条消息) python 之 Tkinter 使用详解_python tkinter_檬柠 wan 的博客-CSDN 博客

3.3 GUI 显示

GUI 显示基于 TKinter 库，软件上由 `createWidgets(self)` 实现



勾选框和开关：用于勾选选择

```
    def createWidgets(self):
# 客户输入框
        self.Text1Var = StringVar(value='客户')
        self.Text1 = Entry(self.top, text='Text1', textvariable=self.Text1Var, font=('
宋体',9))
        self.Text1.place(relx=0.159, rely=0.173, relwidth=0.439, relheight=0.08)
# 标题输入框
        self.Text2Var = StringVar(value='标题')
        self.Text2 = Entry(self.top, text='Text1', textvariable=self.Text2Var, font=('
宋体', 9))
        self.Text2.place(relx=0.159, rely=0.282, relwidth=0.704, relheight=0.08)
        self.style.configure('Check2.TCheckbutton', font=('宋体',9))
        self.Check2 = Checkbutton(self.top, text='生产', variable=self.Check2Var,
style='Check2.TCheckbutton')
        self.Check2.place(relx=0.750, rely=0.173, relwidth=0.134, relheight=0.08)
# switch 开关
        self.Check3Var = StringVar(value='0')
        self.style.configure('Check2.TCheckbutton', font=('宋体', 9))
        self.Check3 = Checkbutton(self.top, text='ESO', variable=self.Check3Var,
style='Switch.TCheckbutton')
        self.Check3.place(relx=0.715, rely=0.385, relwidth=0.150, relheight=0.078)
```

输入框：用于键盘输入信息

下拉选择框：用下拉方式选择输入内容，也可直接键盘输入
预览框：用于预览生成项目名，可键盘输入直接更改
MAKE 按键
文本：GUI 文本显示

```
# 文本
        self.style.configure('Label2.TLabel',anchor='w', font=('宋体',9))
        self.Label2 = Label(self.top, text='预览', style='Label2.TLabel')
        self.Label2.place(relx=0.079, rely=0.684, relwidth=0.055, relheight=0.068)

        self.style.configure('Label2.TLabel',anchor='w', font=('宋体',9))
        self.Label2 = Label(self.top, text='客户', style='Label2.TLabel')
```

```
# 系列下拉框
        self.Combo1List = sys_series_list
        self.Combo1 = Combobox(self.top, values=self.Combo1List, font=('宋体', 10))
        self.Combo1.place(relx=0.159, rely=0.385, relwidth=0.187, relheight=0.068)
        self.Combo1.set(self.Combo1List[0])
    # 版本下拉框
        lcnt = 0
        for i in sys_series_list:
```

```
# 预览显示框
        self.Text3Var = StringVar(value=guser_get_preview_output(self, app_mode))
        self.Text3 = Entry(self.top, text='Text1', textvariable=self.Text3Var, font=('
宋体', 12))
        self.Text3.place(relx=0.159, rely=0.65, relwidth=0.704, relheight=0.133)
        self.Combo2.place(relx=0.437, rely=0.385, relwidth=0.187, relheight=0.068)
```

```
# make 按钮
        self.style.configure('Command1.TButton',font=('宋体',9))
        self.Command1 = Button(self.top, text='M A K E', command=self.Command1_Cmd,
style='Command1.TButton')  # 绑定回调 Command1_Cmd
        self.Command1.place(relx=0.384, rely=0.867, relwidth=0.161, relheight=0.111)
```

3.4 GUI 更新模块

3.4.1. GUI 预览框更新接口

函数原型	update_preview(self)
输入参数	self :tk 实例
返回参数	void
函数说明	更新预览框

根据日期和用户填写信息合并标题并在预览框显示，需要依赖【GUI 预览框输入输出接口 guser_get_preview_output(self, mode)】

```
    # 更新预览框
    def update_preview(self):
        self.Text3Var = StringVar(value=guser_get_preview_output(self, app_mode))
        self.Text3 = Entry(self.top, text='Text1', textvariable=self.Text3Var,
font=('宋体', 12))
        self.Text3.place(relx=0.159, rely=0.65, relwidth=0.704, relheight=0.133)
```

3.4.1. GUI 前端更新初始化接口

函数原型	update_check_init (self)
输入参数	self :tk 实例
返回参数	void
函数说明	前端 UI 更新初始化

APP 启动时调用该函数，获取用于更新对比的【old】相关参数

```
    # 更新前端初始化
    def update_check_init(self):
        global old_app_series
        global old_app_version
        global old_app_scene
        global old_app_production
        global old_app_client
        global old_app_title
        global old_app_mode
        global old_app_time
        old_app_series = self.Combo1.get()
        old_app_version = self.Combo2.get()
        old_app_scene = self.Check1Var.get()
        old_app_production = self.Check2Var.get()
        old_app_client = self.Text1.get()
        old_app_title = self.Text2.get()
        old_app_mode = guser_get_eso_switch(self)
        old_app_time = guser_get_time()
```

3.4.2. GUI 前端更新检查接口

函数原型	update_check(self)

输入参数	self :tk 实例
返回参数	0：无更变 1：更新预览框 2：更新版本下拉框和预览框
函数说明	检查前端 UI 更变

间歇获取当前 GUI 输入框、勾选框、开关等信息，与上一次获取对比，根据信息发生改变状况返回对应更新参数，配合【GUI 前端更新接口 `update(self)`】实时更新 GUI。

```
    # 更新前端检查
    def update_check(self):
        global old_app_series
        global old_app_version
        global old_app_scene
        global old_app_production
        global old_app_client
        global old_app_title
        global old_app_mode
        global app_mode
        global old_app_time
        global sys_time
        app_series = self.Combo1.get()
        if app_series != old_app_series:
            old_app_series = app_series
            return 2
        app_mode = guser_get_eso_switch(self)
        if app_mode != old_app_mode:
            old_app_mode = app_mode
            return 1
        app_version = self.Combo2.get()
        if app_version != old_app_version:
            old_app_version = app_version
            return 1
        app_scene = self.Check1Var.get()
        if app_scene != old_app_scene:
            old_app_scene = app_scene
            return 1
        app_production = self.Check2Var.get()
        if app_production != old_app_production:
            old_app_production = app_production
            return 1
        app_client = self.Text1.get()
        if app_client != old_app_client:
            old_app_client = app_client
            return 1
        app_title = self.Text2.get()
        if app_title != old_app_title:
            old_app_title = app_title
            return 1
        sys_time = guser_get_time()
        if sys_time != old_app_time:
            old_app_time = sys_time
            return 1
        return 0
```

3.4.3. GUI 前端实时更新接口

函数原型	update(self)
输入参数	self :tk 实例
返回参数	void
函数说明	实时更新 UI

实时更新前端 GUI，需依赖【GUI 前端更新检查接口 update_check(self)】获取更新状态。

```
    # 动态更新前端
    def update(self):
        global sys_ui_reset_flag
        if sys_ui_reset_flag == 1:
            sys_ui_reset_flag = 0
            self.reset_gui()
        update_check_flag = self.update_check()  # 获取 GUI 数据更新
        if update_check_flag == 0:
            self.Combo2.after(500, self.update)  # 无更新，重设定时器
            print(".")
            return
        else:
            print("GUI:change")
            if update_check_flag == 2:
                lcnt = 0
                for i in sys_series_list:
                    if self.Combo1.get() == i:
                        self.Combo2List = sys_version_list[lcnt]
                        break
                    else:
                        lcnt = lcnt + 1
                # 更新版本下拉框
                self.Combo2 = Combobox(self.top, values=self.Combo2List, font=('宋体
', 10))
                self.Combo2.place(relx=0.437, rely=0.385, relwidth=0.187,
relheight=0.068)
                self.Combo2.set(self.Combo2List[0])
            self.update_preview()# 更新预览框
            # 完成更新，重设定时器
            self.Combo2.after(500, self.update)
        self.style.configure('Label2.TLabel',anchor='w', font=('宋体',9))
        self.Label2 = Label(self.top, text='标题', style='Label2.TLabel')
        self.Label2.place(relx=0.079, rely=0.282, relwidth=0.055, relheight=0.068)
```

3.4.4. GUI 配置更新接口

函数原型	reset_gui(self)
输入参数	self :tk 实例
返回参数	void
函数说明	根据 cfg 文件读取配置更新 UI

cfg 配置信息只在软件初始化时读取并保存在软件系统中，软件启动后更改的 cfg 配置不会马上更新到软件，需要调用该接口更新同步配置。

该接口结合 UI 更新标志位【sys_ui_reset_flag】使用，【GUI 前端更新 update(self)】时，检查该标志位，若置一，会调用该接口重新更新系列下拉框和版本下拉框选择列表。

```
    #  读 cfg 文件后更新 UI
    def reset_gui(self):
        # 系列下拉框
        self.Combo1List = sys_series_list
        self.Combo1 = Combobox(self.top, values=self.Combo1List, font=('宋体', 10))
        self.Combo1.place(relx=0.159, rely=0.385, relwidth=0.187, relheight=0.068)
        self.Combo1.set(self.Combo1List[0])
        # 版本下拉框
        self.Combo2List = sys_version_list[0]   # 默认显示第一个就行
        self.Combo2 = Combobox(self.top, values=self.Combo2List, font=('宋体', 10))
        self.Combo2.place(relx=0.437, rely=0.385, relwidth=0.187, relheight=0.068)
        self.Combo2.set(self.Combo2List[0])
        # 预览显示框
        self.Text3Var = StringVar(value=guser_get_preview_output(self, app_mode))
        self.Text3 = Entry(self.top, text='Text1', textvariable=self.Text3Var,
font=('宋体', 12))
        self.Text3.place(relx=0.159, rely=0.65, relwidth=0.704, relheight=0.133)
```

```
    # 动态更新前端
    def update(self):
        global sys_ui_reset_flag
        if sys_ui_reset_flag == 1:
            sys_ui_reset_flag = 0
            self.reset_gui()
        update_check_flag = self.update_check()  # 获取 GUI 数据更新
        if update_check_flag == 0:
            self.Combo2.after(500, self.update)  # 无更新，重设定时器
            print(".")
            return
        else:
            print("GUI:change")
            if update_check_flag == 2:
                lcnt = 0
                for i in sys_series_list:
                    if self.Combo1.get() == i:
                        self.Combo2List = sys_version_list[lcnt]
                        break
                    else:
                        lcnt = lcnt + 1
                # 更新版本下拉框
                self.Combo2 = Combobox(self.top, values=self.Combo2List, font=('宋体',
10))
                self.Combo2.place(relx=0.437, rely=0.385, relwidth=0.187,
relheight=0.068)
                self.Combo2.set(self.Combo2List[0])
            # 更新预览框
            self.update_preview()
            # 完成更新，重设定时器
            self.Combo2.after(500, self.update)
        self.style.configure('Label2.TLabel',anchor='w', font=('宋体',9))
        self.Label2 = Label(self.top, text='标题', style='Label2.TLabel')
        self.Label2.place(relx=0.079, rely=0.282, relwidth=0.055, relheight=0.068)
```

3.5 菜单栏

3.5.1. 菜单栏创建接口

函数原型	guser_app_create_menu()
输入参数	void
返回参数	void
函数说明	创建菜单栏

```
    # 创建菜单栏
def guser_app_create_menu():
    # 创建一个菜单
    menu = tkinter.Menu(top)
    # 创建子菜单
    filemenu = tkinter.Menu(menu, tearoff=0)
    filemenu.add_command(label=task_directory_path,
command=guser_menu_open_directory)
    filemenu.add_command(label=task_record_path, command=guser_menu_open_record)
    filemenu.add_command(label=task_source_sdk_path,
command=guser_menu_open_source_sdk)
    filemenu2 = tkinter.Menu(menu, tearoff=0)
    filemenu2.add_command(label="Open config", command=guser_menu_setting)
    filemenu2.add_command(label="Read config", command=guser_menu_read_cfg)
    # 将子菜单加入到菜单条中
    menu.add_cascade(label=u"打开", menu=filemenu)
    menu.add_cascade(label=u"设置", menu=filemenu2)
```

3.5.2. 菜单栏打开相关接口

函数原型	guser_menu_open_directory ()
输入参数	void
返回参数	void
函数说明	打开任务目录

```
    def guser_menu_open_directory():
    path = task_directory_path
    os.startfile(path)
    print("[menu]:open < %s > success" % path)
```

函数原型	guser_menu_open_record()
输入参数	void
返回参数	void
函数说明	打开任务记录目录

函数原型	guser_menu_open_source_sdk()
输入参数	void
返回参数	void
函数说明	打开 SDK 源目录

3.5.3. 菜单栏设置相关接口

```
def guser_menu_open_source_sdk():
path = task_source_sdk_path
os.startfile(path)
print("[menu]:open < %s > success" % path)
```

```
def guser_menu_setting():
path = sys_cfg_path
os.startfile(path)
print("[menu]:open < %s > success" % path)
```

函数原型	guser_menu_setting()
输入参数	void
返回参数	void
函数说明	打开 cfg 文件设置

函数原型	guser_menu_read_cfg()
输入参数	void
返回参数	void
函数说明	读取 cfg 文件设置

```
def guser_menu_read_cfg():
global cfg_data
global sys_ui_reset_flag
cfg_data = guser_read_cfg(sys_cfg_path)
guser_cfg_decode(cfg_data)
sys_ui_reset_flag = 1
print("[menu]:read < %s > success" % sys_cfg_path)
```

3.5.4. 菜单栏关于相关接口

函数原型	guser_menu_about()
输入参数	void
返回参数	void
函数说明	打开关于弹窗

```
def guser_menu_about():
guser_popup_window("关于", pop_about_str)
print("[menu]:pup < about > success")
```

3.5.5. 菜单栏帮助相关接口

函数原型	guser_menu_help()
输入参数	void
返回参数	void
函数说明	打开帮助弹窗

```
def guser_menu_help():
guser_popup_window("使用说明", pop_help_str)
print("[menu]:pup < help > success")
pass
```

3.5.6. 菜单栏主题相关接口

函数原型	guser_menu_theme_switch()
输入参数	void
返回参数	void
函数说明	切换主题

```
    # 菜单栏主题切换 #不加这么多花里胡哨的功能，不保存在 cfg 配置里，每次打开会恢复默认
def guser_menu_theme_switch():
    global sys_theme
    if sys_theme == 0:
        top.tk.call("set_theme", "light")
        # top.update()
    else:
        top.tk.call("set_theme", "dark")
        # top.update()
    sys_theme = (sys_theme + 1) % 2     # 取余，奇偶切换
    print("[menu]:theme_switch success")
```

3.6 弹窗接口

函数原型	guser_popup_window(title_str, show_str)
输入参数	title_str:标题、show_str：弹窗文档
返回参数	void

```
    # 弹窗
def guser_popup_window(title_str, show_str):
    my_font = Font(family="宋体",size=12)
    tkinter.messagebox.showinfo(title_str, show_str)
```

第4章 接口与功能实现

4.1 文件读写模块

该模块主要用于读写 txt 文件获取 cfg 信息

4.1.1. 读文件接口

函数原型	guser_read_cfg(path)
输入参数	path:目标地址
返回参数	cfg 配置数据
函数说明	读 cfg 配置文档

从外部读取 cfg 文件配置，若该文件不存在，新建文件【guser_config.txt】

```
    # 从外部文件读取配置信息
def guser_read_cfg(path):
    if os.path.exists(path) == 0:        # 没有 cfg 文件，通常是第一次使用
        guser_popup_window("使用说明", pop_explain_str)            # 弹出使用说明
        guser_write_cfg(sys_data, path)  # 写 cfg 文件
        print("[warning] no cfg file! New file successfully")
    with open(path, mode="r", encoding="utf-8") as f:
        data = f.readlines()  # read()    一次性读全部内容，以列表的形式返回结果。#
readlines() 一行行读放在列表里
    f.close()
    print("sys read cfg data succeed\n")
    return data
```

4.1.2. 写文件接口

函数原型	guser_write_cfg(datalist,savepath)
输入参数	datalist：数据列表、savepath：目标文件地址
返回参数	void
函数说明	把数据列表写入目标文件

向外部 cfg 文件写配置，每次写都会写入【del_datalist】列表删除全部数据，再使用【datalist】

列表重写 cfg 配置

```
    # 向外部 config 文件写数据
def guser_write_cfg(datalist,savepath):
    del_datalist =['',]  # 写空列表删除数据
    for data in del_datalist:
        with open(savepath, mode="w", encoding="utf-8") as f:
            f.write(data)   # 写数据
    f.close()
    for data in datalist:
        with open(savepath, mode="a", encoding="utf-8") as f:
            f.write(data)       # 写数据
            # f.write("\n")     # 换行
    f.close()
    print("sys write cfg data succeed\n")
```

4.1.3. 序号加一接口

函数原型	guser_serial_add(mode)
输入参数	mode：eso 模式
返回参数	返回序号+1
函数说明	创建目录后序号加一写入 cfg 文件

```
    # 序号+1,传入参数：1 是日常支持序号。2 是测试协助序号
def guser_serial_add(mode):
    global cfg_serial0
    global cfg_serial1
    if mode == '0':
        output_data = int(cfg_serial0)
        cfg_serial0 = str(output_data + 1)
        print("cfg_serial0 +1 = %s" % cfg_serial0)
        return cfg_serial0
    elif mode == '1':
        output_data = int(cfg_serial1)
        cfg_serial1 = str(output_data + 1)
        print("cfg_serial1 +1 = %s" % cfg_serial1)
        return cfg_serial1
```

4.2 信息获取模块

4.2.1. 日期字符串获取接口

函数原型	guser_get_time()
输入参数	void
返回参数	日期字符串
函数说明	获取日期字符串

获取日期并拼接日期字符串，需要依赖【time 模块】

```
    # 获取系统时间字符串
def guser_get_time():
    year = datetime.now().year
    month = datetime.now().month
    day = datetime.now().day
    if month < 10:
        str_month = '0' + str(month)
    else:
        str_month = str(month)
    if day < 10:
        str_day = '0' + str(day)
    else:
        str_day = str(day)
    str_time = str(year) + str_month + str_day
    return str_time
```

4.2.2. cfg 信息解包接口

函数原型	guser_cfg_decode(cfg_data)
输入参数	cfg_data :cfg 文件数据
返回参数	void
函数说明	系统读取 cfg 文件序号、芯片型号、SDK 版本信息

通过获取接口，解包 cfg 文件读取序号、芯片型号、SDK 版本信息存入系统全局变量 `cfg_serial0`、`cfg_serial1` 和全局列表 `sys_series_list`、`sys_version_list` 中后续功能只需要操作全局变量，减少反复读取 cfg 文件操作

```
# 解包
def guser_cfg_decode(cfg_data):
    global cfg_serial0   # 声明 cfg_seria0 为全局变量
    global cfg_serial1
    global sys_series_list
    global sys_version_list
    cfg_serial0 = guser_get_serial0(cfg_data)
    cfg_serial1 = guser_get_serial1(cfg_data)
    sys_series_list = guser_get_series_list(cfg_data)
    sys_version_list = guser_get_version_list(cfg_data)
    print("[decode]:decode cfg success")
```

4.2.3. cfg 序号获取接口

函数原型	guser_get_serial0(cfg_data)
输入参数	cfg_data :cfg 文件数据
返回参数	日常支持序号
函数说明	获取日常支持序号信息

函数原型	guser_get_serial1(cfg_data)
输入参数	cfg_data :cfg 文件数据
返回参数	测试协助序号
函数说明	获取测试协助序号信息

先进行 cfg 文件有效性检查，检查 cfg 文件是否有该行的信息，如果没有判定为无，直接设置序号为 0;

再进行数据有效性检查，强转 int 后判断数据是否在 0~1000 之间，也起到去除换行符 \0 的作用；若不在此范围内直接设置为 0，确保从这里开始后序号都是正确的;

此处兼容性处理未测试，如果该行内信息不是纯数字可能会出错;

两个获取序号函数实现十分相似，可以考虑优化为一个接口;

```
    # 获取 cfg 配置的日常支持序号信息
def guser_get_serial0(cfg_data):
    if len(cfg_data) > cfg_serial0_num_line:
        serial0_int_data = int(cfg_data[cfg_serial0_num_line])
        if serial0_int_data >= 0 and serial0_int_data < 1000:
            output_serial0_data =str(serial0_int_data)
        else:
            output_serial0_data = "0"
            print("[warning] serial0 illegality, use -> sys_serial:0")
    else:
        output_serial0_data = "0"
        print("[warning] no cfg data, use -> sys_serial:0")
    print("serial0:", output_serial0_data)
    return output_serial0_data

# 获取 cfg 配置的测试协助序号信息
def guser_get_serial1(cfg_data):
    if len(cfg_data) > cfg_serial1_num_line:
        serial1_int_data = int(cfg_data[cfg_serial1_num_line])
        if serial1_int_data >= 0 and serial1_int_data < 1000:
            output_serial1_data =str(serial1_int_data)
        else:
            output_serial1_data = "0"
            print("[warning] serial1 illegality, use -> sys_serial:0")
    else:
        output_serial1_data = "0"
        print("[warning] no cfg data, use -> sys_serial:0")
    print("serial1:", output_serial1_data)
```

4.2.4. cfg 芯片系列获取接口

函数原型	user_get_series_list(cfg_data)
输入参数	cfg_data :cfg 文件数据
返回参数	芯片系列列表
函数说明	获取芯片系列列表信息

和获取序号信息不同的是，芯片系列信息在 cfg 文件中有一行检验字符，cfg 有效性检验时，需要检验字符完全匹配，且芯片系列数据行存在。若检验通过，系统会读取芯片系列数据行，以“/”为分隔符创建一个列表；若检验失败，则使用系统自带的列表以保证后续功能正常，后面用户可自行在 cfg 文件更改信息；


```
    # 获取 cfg 配置的芯片系列信息
def guser_get_series_list(cfg_data):
    if len(cfg_data) > cfg_series_list_line and
cfg_data[cfg_series_check_line] == cfg_series_check_str:
        data_str = cfg_data[cfg_series_list_line]
        output_series_list = [str(x) for x in data_str.strip().split("/")]
    else:  # 如果 cfg 文件没有数据，使用系统默认列表
        output_series_list = sys_series_list
        print("[warning] no cfg data, use -> sys_series_list")
    print("series list:", output_series_list)
    return output_series_list
```

4.2.5. cfg 代码版本获取接口

函数原型	guser_get_version_list(cfg_data)
输入参数	cfg_data :cfg 文件数据
返回参数	SDK 代码版本列表
函数说明	获取 SDK 代码版本列表信息

和芯片系列获取接口一样，代码版本信息在 cfg 文件中有一行检验字符，cfg 有效性检验时，需要检验字符完全匹配，且芯片代码版本数据行存在。若检验通过，系统会读取代码版本数据行，以“/”为分隔符创建一个列表；若检验失败，则使用系统自带的列表以保证后续功能正常，后面用户可自行在 cfg 文件更改信息；

```
    # 获取 cfg 配置的 SDK 版本信息
def guser_get_version_list(cfg_data):
    long = len(sys_series_list)
    if len(cfg_data) > (cfg_version_check_line + long) and
cfg_data[cfg_version_check_line] == cfg_version_check_str:
        valid_cfg_data = cfg_data[cfg_version_list_line: cfg_version_list_line +
long]  # 在 cfg 数据中截取出 version 部分信息
        output_version_list = [[] for i in range(long)]  # 创建二维列表容器
        for i in range(long):
            output_version_list[i] = [str(x) for x in
valid_cfg_data[i].strip().split("/")]
    else:
        output_version_list = sys_version_list
        print("[warning] no cfg data, use -> sys_version_list")
    print("version_list:", output_version_list)
    return output_version_list
```

4.2.6. GUI 协助模式获取接口

函数原型	guser_get_eso_switch(self)
输入参数	self :tk 实例
返回参数	esc 模式开关状态
函数说明	获取 esc 模式开关状态

```
    # 获取 eso 模式开关状态
def guser_get_eso_switch(self):
    app_mode = self.Check3Var.get()
    return app_mode
```

4.2.7. GUI 芯片系列获取接口

函数原型	guser_get_chip_series(self)
输入参数	self :tk 实例
返回参数	芯片系列
函数说明	获取芯片系列

```
    # 获取 GUI:芯片系列
def guser_get_chip_series(self):
    app_series = self.Combo1.get()
    return app_series
```

4.2.8. GUI 软件版本获取接口

函数原型	guser_get_sdk_version(self)
输入参数	self :tk 实例
返回参数	版本系列
函数说明	获取版本系列

```
    # 获取 GUI:SDK 版本
def guser_get_sdk_version(self):
    app_version = self.Combo2.get()
    return app_version
```

4.2.9. GUI 预览框输入获取接口

函数原型	guser_get_preview_input(self)
输入参数	self :tk 实例
返回参数	预览框当前字符串
函数说明	获取预览框当前的字符串

```
    # 获取 GUI:预览框输入字符串
def guser_get_preview_input(self):
    output_str = self.Text3.get()
    return output_str
```

4.2.10. GUI 预览框输出获取接口

函数原型	guser_get_preview_output(self, mode)
输入参数	self :tk 实例、mode：模式
返回参数	即将输出到预览框的字符串
函数说明	获取即将输出到预览框的字符串

```
    # 获取预览框输出字符串
def guser_get_preview_output(self, mode):
    app_scene = self.Check1Var.get()
    app_production = self.Check2Var.get()
    app_client = self.Text1.get()
    app_title = self.Text2.get()
    app_time = sys_time
    app_series = guser_get_chip_series(self)
    use_series_str = ""         # 取出系列字符串中的数字部分用于组合目录文件名
    for i in range(len(app_series)):
        if app_series[i] >= "0" and app_series[i] <= "9":
            use_series_str = use_series_str + app_series[i]
    app_version = guser_get_sdk_version(self)
    if app_scene == '1':
        app_time = app_time + 'x'
    if app_production == '1':
        app_time = app_time + 's'
    if mode == '0':
        serial_num = cfg_serial0
        if(serial_num>= "0" and serial_num <= "9"):
            serial_num = '0' + serial_num
        app_preview = serial_num + '.' + app_time + '_' + use_series_str +
'v' + app_version + '_' + app_client + '_' + app_title
    elif mode == '1':
        serial_num = cfg_serial1
        app_preview = '0x' + serial_num + '.' + app_time + '_' +
use_series_str + 'v' + app_version + '_' + app_client + '_' + app_title
    else:
        app_preview = 'Error' + '.' + app_time + '_' + use_series_str + 'v'
+ app_version + '_' + app_client + '_' + app_title
        print("[preview]:mode error!")
    return app_preview
```

4.3 解压模块

4.3.1. 解压 zip 接口

函数原型	guser_unzip_file(zip_file,target_dir)
输入参数	zip_file:压缩文件地址、target_dir：解压后地址
返回参数	void
函数说明	解压 zip 文件

该接口用于解压 zip 格式 SDK 压缩包文件

```
    # 解压 zip
def guser_unzip_file(zip_file,target_dir):
    with zipfile.ZipFile(zip_file,"r") as zfile:
        for file in zfile.namelist():
            zfile.extract(file,target_dir)
```

4.3.2. SDK 拷贝器

函数原型	guser_sdk_copier(self,path)
输入参数	self:tk 实例、path：解压后地址
返回参数	void
函数说明	拷贝对应 SDK 文件到新建目录

SDK 拷贝器：获取 GUI 当前输入设置，在 SDK 源目录中查找对应版本的 SDK 并拷贝到新建的任务目录中，该接口需要依赖解压 zip 接口【guser_unzip_file】

```
    # SDK 拷贝器
def guser_sdk_copier(self,path):
    app_series = self.Combo1.get()
    app_version = self.Combo2.get()
    exist_zip_file = 0   # 存在 zip 文件的标志位
    # find_path：查找 zip 文件的文件目录；source_path：解压 zip 的源地址；
target_path：zip 解压目标地址
    find_path = task_source_sdk_path + '\\' + app_series + '\\' +
app_version + '\\'
    if os.path.exists(find_path) != 0:
        file_list = os.listdir(find_path)
        print("file_list:", file_list)
        for i in range(len(file_list)):
            if ".zip" in file_list[i]:
                exist_zip_file = 1
                source_path = find_path + file_list[i]
                target_path = path + '\\PublicSDK'
                guser_unzip_file(source_path, target_path)
                print("copier:uncompress <%s> succeed" % file_list[i])
        if exist_zip_file == 0:
            print("[warning] copier:uncompress no SDK zip file!\n")
    else:
        print("[warning] copier:uncompress no SDK source directory!\n")
```

4.4 目录管理模块

4.4.1. 目录结构

目录管理模块总工程目录包括三个文件夹

(1) TaskDirectory：任务目录
存储任务目录单项，目录单项根据模板生成

(2) TaskRecord：任务记录
存储任务目录单项对应的记录 md 文件，记录文件根据模板生成

(3) TaskSourceSDK：SDK 源库
保存各系列各版本的 SDK，用于拷贝器获取公版到新建目录单项

- TaskDirectory（任务目录）
  - 0x1.20230410_700v201_同事A_一个标题
  - 0x2.20230410_700v201_同时B_一个标题
  - 1.20230410_700v201_新闻达_一个标题
  - 2.20230410_700v201_新闻达_一个标题

- TaskRecord（任务记录）
  - 1.20230410_700v201_新闻达_一个标题.md
  - 1.20230410_700v201_新闻达_一个标题.md

- TaskSourceSDK（SDK源库）
  - AC696N
    -101
    696V101.zip
    -102
    696V102.zip
  - AC702N
    -101
    701v201.zip
    -102
    701v202.zip

4.4.2. 创建总工程目录

函数原型	guser_create_overall_directory()
输入参数	void
返回参数	void
函数说明	创建总工程目录

```
    # 创建总工程目录
def guser_create_overall_directory():
    if os.path.exists(task_directory_path) == 0:     # 检查任务目录总文件夹
        os.makedirs(task_directory_path)            # 生成任务目录总文件夹
    if os.path.exists(task_record_path) == 0:        # 检查任务目录总文件夹
        os.makedirs(task_record_path)               # 生成任务记录总文件夹
    if os.path.exists(task_source_sdk_path) == 0:    # 检查任务目录总文件夹
        os.makedirs(task_source_sdk_path)           # 生成 SDK 源总文件夹
        guser_create_source_sdk()                   # 生成 SDK 源目录结构
    else:
        guser_create_source_sdk()
```

4.4.3. 创建 SDK 源目录

函数原型	guser_create_source_sdk()
输入参数	void
返回参数	void
函数说明	创建 SDK 源目录

```
    # 创建 SDK 源目录
def guser_create_source_sdk():
    if os.path.exists(task_source_sdk_path) == 0:
        return
    else:
        for i in range(len(sys_series_list)):
            path = task_source_sdk_path + '\\' + sys_series_list[i]
            if os.path.exists(path) == 0:           # 检查目标文件夹是否已存在
                os.makedirs(path)                  # 生成系列文件夹
            # 在系列文件夹下再创建版本文件夹
            for j in range(len(sys_version_list[i])):
                path = task_source_sdk_path + '\\' + sys_series_list[i] +
'\\' + sys_version_list[i][j]
                if os.path.exists(path) == 0:       # 检查目标文件夹是否已存在
                    os.makedirs(path)              # 生成版本文件夹
```

4.4.4. 创建任务记录单项

函数原型	guser_create_record(app_preview)
输入参数	app_preview：任务记录文档名
返回参数	void
函数说明	创建任务记录文档

根据模板创建单项目录对应的任务记录 md 文档，若模板不存在则创建空 md 文件

```
    # 创建任务记录单项
def guser_create_record(app_preview):
    # 这里没兼容记录总目录被删或者新建 md 文件已存在的情况
    template_path = sys_record_template_path
    target = task_record_path + "\\" + app_preview + ".md"
    if os.path.exists(sys_record_template_path) == 0:  # 如果没有模板则生成一个
        print("[warning] no %s" % template_path)
        datastr = "请设置模板!\n"  # 写空列表删除数据
        with open(template_path, mode="w", encoding="utf-8") as f:
            f.write(datastr)  # 写数据
        f.close()
    shutil.copyfile(template_path, target)
```

4.4.5. 创建任务目录单项

函数原型	guser_create_directory(self, app_preview)
输入参数	self:tk 实例、app_preview：任务目录名
返回参数	void
函数说明	创建单项任务目录文件夹

该函数用于【MAKE】按键功能，根据模板创建任务目录单项，并将对应的公版 SDK 拷贝到【PublicSDK】文件夹

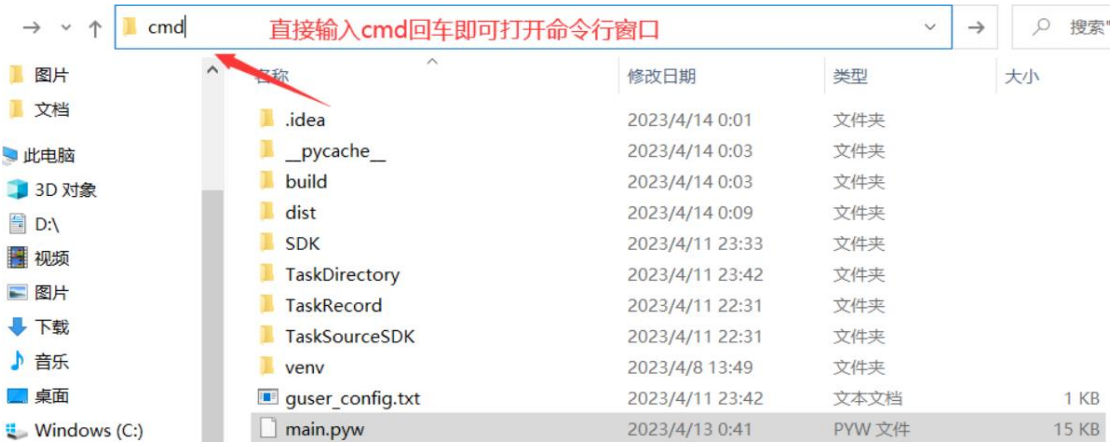
```
    # 创建任务目录单项
def guser_create_directory(self, app_preview):
    path = task_directory_path + '\\' + app_preview
    print(path)
    cnt = 0
    while (os.path.exists(path)):  # 如果路径下该文件已经存在 # 有序号自加存在，基本不可能有这种情况
        cnt = cnt + 1
        scnt = str(cnt)
        path = task_directory_path + '\\' + app_preview + '(' + scnt + ')'
    os.makedirs(path) # 生成文件夹
    if os.path.exists(sys_project_template_path):  # 查找模板
        guser_unzip_file(sys_project_template_path, path)  # 解压工程模板
        if os.path.exists(path + '\\' + "PublicSDK") == 0:  # 查找模板
            os.makedirs(path + '\\' + "PublicSDK")  # 生成 PublicSDK 文件夹，防止模板中没 PublicSDK 文件夹
    else:  # 如果没有工程模板，直接生成文件夹
        os.makedirs(path + '\\' + "PublicSDK")  # 生成 PublicSDK 文件夹
        os.makedirs(path + '\\' + "ClientSDK")  # 生成 ClientSDK 文件夹
        print("[warning] no ProjectTemplate.zip")
    guser_sdk_copier(self, path)  # 向生成目录拷贝公版 SDK

    os.startfile(path)  # 打开目录
```

第5章 相关库说明

5.1 EXE 可执行文件打包

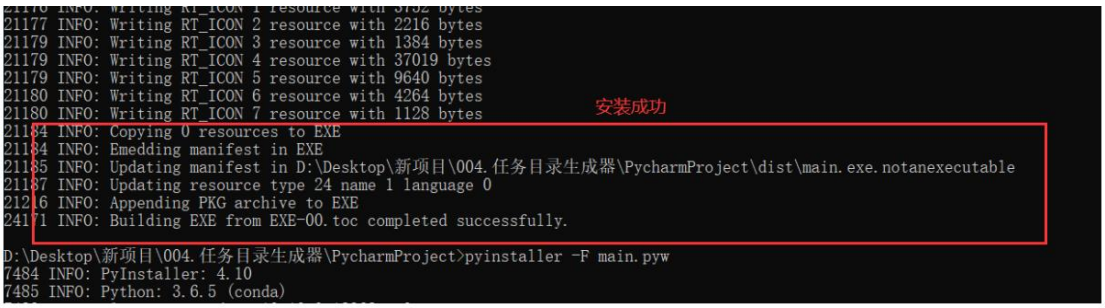
5.1.1. Pyinstalle 安装

在 py 项目 main.py 目录下打开 cmd 命令行窗口



cmd 使用 pip 安装 pyinstaller，输入：

pip install pyinstaller



5.1.2. Pyinstalle 使用

使用 pyinstaller 打包 exe 文件，cmd 输入：

pyinstaller -F main.py

使用 pyinstaller 打包 exe 文件，去除命令窗口 cmd 输入：

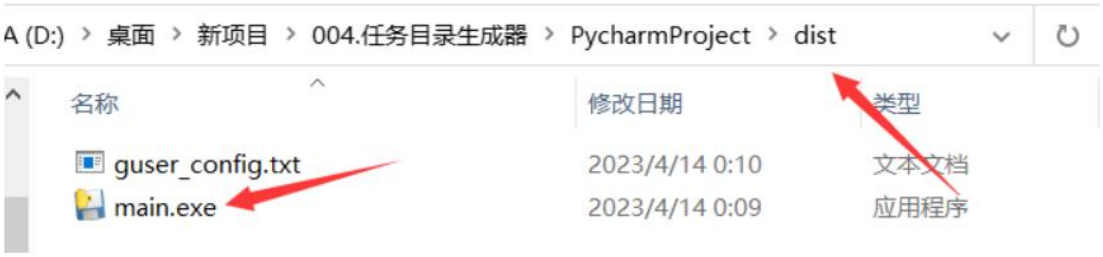
pyinstaller -F -w main.py

main.py 是自己的 py 文件，如果 py 项目有自己的 GUI，不需要打开命令行，可以把后缀.py 改为.pyw 再打包(main 文件也要改)

pyinstaller -F -w main.pyw

```
22397 INFO: Warnings written to D:\Desktop\新项目\004.任务目录生成器\PycharmProject\build\main\warn-main.txt
22435 INFO: Graph cross-reference written to D:\Desktop\新项目\004.任务目录生成器\PycharmProject\build\main\xref-main.html
22550 INFO: checking PYZ
22587 INFO: checking PKG
22654 INFO: Building because toc changed
22654 INFO: Building PKG (CArchive) main.pkg
31978 INFO: Building PKG (CArchive) main.pkg completed successfully.
31999 INFO: Bootloader d:\programdata\anaconda3\lib\site-packages\PyInstaller\bootloader\Windows-64bit\runw.exe
31999 INFO: checking EXE
32039 INFO: Building because console changed
32039 INFO: Building EXE from EXE-00.toc
32042 INFO: Copying bootloader EXE to D:\Desktop\新项目\004.任务目录生成器\PycharmProject\dist\main.exe.notanexecutable
32175 INFO: Copying icon to EXE
32175 INFO: Copying icons from ['d:\\programdata\\anaconda3\\lib\\site-packages\\PyInstaller\\bootloader\\images\\icon-windowe
32208 INFO: Writing RT_GROUP_ICON 0 resource with 104 bytes
32208 INFO: Writing RT_ICON 1 resource with 3752 bytes
32209 INFO: Writing RT_ICON 2 resource with 2216 bytes
32209 INFO: Writing RT_ICON 3 resource with 1384 bytes
32211 INFO: Writing RT_ICON 4 resource with 38188 bytes
32212 INFO: Writing RT_ICON 5 resource with 9640 bytes
32212 INFO: Writing RT_ICON 6 resource with 4264 bytes
32212 INFO: Writing RT_ICON 7 resource with 1128 bytes
32216 INFO: Copying 0 resources to EXE
32217 INFO: Embedding manifest in EXE
32219 INFO: Updating manifest in D:\Desktop\新项目\004.任务目录生成器\PycharmProject\dist\main.exe.notanexecutable
32220 INFO: Updating resource type 24 name 1 language 0
32224 INFO: Appending PKG archive to EXE
35159 INFO: Building EXE from EXE-00.toc completed successfully.          打包成功
```

exe 文件生成在 dist 文件夹下，注意不要被杀毒软件杀了



5.2 Time 模块

用于获取系统时间

```python
from datetime import datetime, date, timedelta


    # 获取系统时间字符串
def guser_get_time():
    year = datetime.now().year
    month = datetime.now().month
    day = datetime.now().day
    if month < 10:
        str_month = '0' + str(month)
    else:
        str_month = str(month)
    if day < 10:
        str_day = '0' + str(day)
    else:
        str_day = str(day)
    str_time = str(year) + str_month + str_day
    return str_time
```

5.3 os 模块

提供系统文件操作相关支持

```
import os
# 判断文件是否存在
os.path.exists(find_path)
# 获取文件列表
os.listdir(find_path)
# 打开文件
os.startfile(path)
# 生成文件夹
os.makedirs(path)
```

5.4 zipfile 模块

用于压缩解压 zip 文件，在本项目中只提供解压功能

```
import zipfile

# 解压 zip
def guser_unzip_file(zip_file,target_dir):
    with zipfile.ZipFile(zip_file,"r") as zfile:
        for file in zfile.namelist():
            zfile.extract(file,target_dir)
```