Python编程技巧

目录

[1 环境配置与包安装 1](#_Toc152092433)

[1.1. Anaconda的配置 2](#_Toc152092434)

[1.1.1. 安装、环境变量、镜像源 2](#_Toc152092435)

[1.1.2. 常用Anaconda Prompt命令 2](#_Toc152092436)

[1.1.3. 改变Anaconda Prompt默认目录和环境 3](#_Toc152092437)

[1.2. VS Code配置Python解释器和调试器 3](#_Toc152092438)

[1.3. 导入包 4](#_Toc152092439)

[2 Python编程基础 4](#_Toc152092440)

[1.1. 代码风格规范 4](#_Toc152092441)

[1.2. 缩进 5](#_Toc152092442)

[1.3. 函数 5](#_Toc152092443)

[1.4. 条件 5](#_Toc152092444)

[1.5. 循环 6](#_Toc152092445)

[1.6. 基本文件操作 7](#_Toc152092446)

[1.7. 随机数生成 7](#_Toc152092447)

[3 Python数据分析 7](#_Toc152092448)

[1.1. 数据分析常用包 7](#_Toc152092449)

[1.2. Matplotlib数据绘图 7](#_Toc152092450)

[1.1.1. 基本绘图逻辑 8](#_Toc152092451)

[1.1.2. 常用函数简单说明 9](#_Toc152092452)

[1.3. Excel表格读取——xlrd 10](#_Toc152092453)

[1.1.1. 将xlrd读入的特殊格式数转换为数值 10](#_Toc152092454)

[1.4. PrettyTable命令行绘表 11](#_Toc152092455)

[4 Python前端开发 11](#_Toc152092456)

[1.1. Tkinter包 12](#_Toc152092457)

[5 Python机器学习 12](#_Toc152092458)

[6 其他 12](#_Toc152092459)

[1.1. .py文件打包成.exe文件——pyinstaller 12](#_Toc152092460)

# 环境配置与包安装

作为一种脚本语言，Python的运行不需要编译器，但需要专门的解释器和运行环境。解释器可以从官网下载，运行环境中需要安装的功能包则可由命令行下载。

## Anaconda的配置

Anaconda有两个主要功能：创建虚拟环境、管理包。不同的虚拟环境中可以安装不同类型、不同版本的包，可解决库版本的兼容性问题，提高Python运行环境的功能专门性。

虚拟环境保存在..\Anaconda\envs目录中。

### 安装、环境变量、镜像源

·下载地址：<https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/>

·更换镜像源方法：<https://mirrors.tuna.tsinghua.edu.cn/help/anaconda/>

·设置环境变量，在系统Path变量下增加三个目录：

|  |
| --- |
| ..\Anaconda  ..\Anaconda\Scripts  ..\Anaconda\Library\bin |

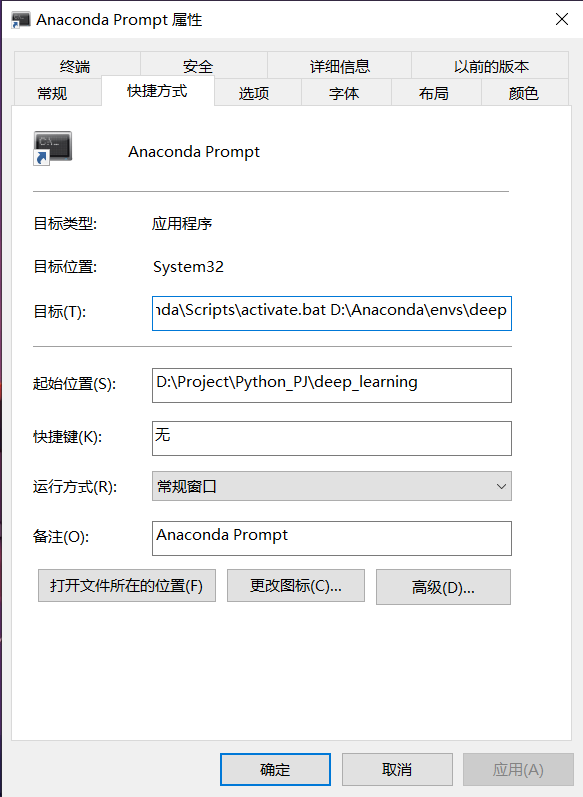
### 常用Anaconda Prompt命令

Anaconda Prompt是Anaconda自带的命令行控制台，正在使用的虚拟空间显示在打开目录前。（下表中蓝色参数不需要把尖括号打上）

|  |  |
| --- | --- |
| conda config --show-sources | 查看当前使用源 |
| conda config –add/-remove <channel> | 添加/删除源 |
| conda create --name <env\_name> | 创建名为env\_name的虚拟环境 |
| conda delete --name <env\_name> --all | 删除虚拟环境env\_name |
| conda activate/deactivate my\_env | 进入/退出（回到默认环境base）当前虚拟环境 |
| conda env list | 查看所有建立的虚拟环境 |
| conda list | 查看当前虚拟环境（conda下载）的包及版本 |
| conda install <package=xxx> | 用conda安装名为package的包xxx版本（可省） |
| conda uninstall <package> | 删除虚拟环境中的某个包 |
| pip install <package=xxx> | 用pip安装包（pip中的包比conda多） |
| conda env export > <env\_export.yml> | 导出虚拟环境为yml文件到当前目录 |
| conda env create -f <env\_export.yml> | 使用yml文件创建新环境 |
| C: | 进入C盘根目录 |
| cd C:\project\deep | 进入文件夹C:\project\deep |
| python <py\_test.py> | 运行当前目录下的python文件 |

### 改变Anaconda Prompt默认目录和环境

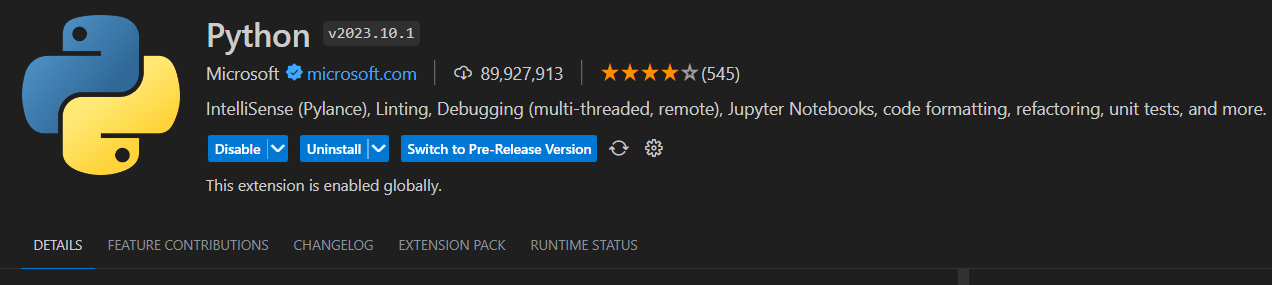
打开Prompt属性页面，起始位置即为设置的默认开启目录（deep\_learning文件夹），目标后面的目录是设置的默认起始虚拟环境deep的文件夹。





## VS Code配置Python解释器和调试器

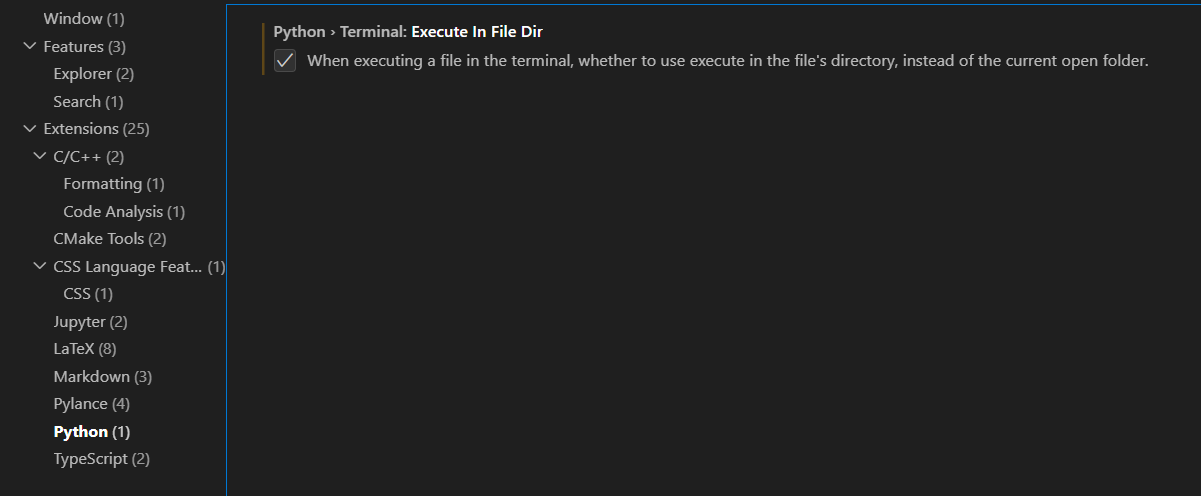
下载VS Code中的Python扩展，打开Python文件即可使用，若缺少必要外设，VS Code会提示下载。默认使用的调试器是Pylint，对代码风格要求严格。



右下角会显示当前使用语言为Python，版本为3.11.3，启用的虚拟环境为plot。点击虚拟环境按钮，在上方的搜索栏可以切换虚拟环境。



注意，若要在文件操作时正常使用相对路径，需要在VS Code的“设置-Python”中勾选“执行位置使用当前文件目录”。



## 导入包

Python文件使用包之前必要先用import语句从环境中导入包，每个包都包含模块和函数，可以全部导入也可只导入所需模块。使用as定义导入对象的缩写。

本质上，包和模块就是一个预定义了函数、变量的.py文件（或其他支持的文件格式）。包和模块、模块与模块之前可以存在多层嵌套关系。通用的包也被称为库。

导入方法有三种：

|  |
| --- |
| import numpy as np  from matplotlib import pyplot as plt  from tkinter import \* |

1. 表示导入整个numpy包并缩写为np，调用其内部的函数需写成np.arange()，表示访问包np（numpy）中的arange函数；
2. 表示仅从matlpotlib包里导入pyplot模块并缩写为plt，调用pyplot模块内部函数写作plt.plot()，此时调用函数不需要包名称；
3. 表示从tkinter包导入所有模块，此时调用也不需要包名称。

# Python编程基础

这里介绍Python基本的语句、运行逻辑和数据结构

## 代码风格规范

·定义顺序规范：包导入语句🡪常量定义🡪变量定义🡪函数定义🡪执行语句

·命名规范：常量全大写，变量全小写，单词之间用下划线隔开，变量名最好可以望文识意，尽量用英文名。

|  |
| --- |
| THIS\_IS\_A\_CONSTANT = 12  this\_is\_a\_variable = input() |

## 注释

Python单行注释以井号“#”开头，多行注释以英语三引号作为开头结尾。

## 缩进

Python语句末尾不需要加英文分号“;”，而是利用句首缩进判断语句在函数体、判断体、循环体的内部还是外部。下面的函数例子中，变量y的赋值语句前没有缩进，因此从它开始函数体结束。

## 基本数据类型

## 函数

Python定义函数的关键词是def，注意句末的顿号:，下面的例子展示了函数的用法：

|  |
| --- |
| x = input()  def function\_name(parameter1, parameter2 = 1):      global x  return x \* parameter1 + parameter2  y = function\_name(parameter1 = 1, 2) |

·参数默认值：若不传参给parameter2，执行函数时它默认为1；

·指定参数传参：parameter1 = 1，这种传参方法可以不按定义顺序传。

·改变全局变量：global关键词表示函数中出现的变量x是全局变量，若不加global，函数会创建一个同名的局部变量x；

## 条件

Python条件语句格式如下，与、或逻辑运算用相应的英文and、or表示，非仍然是!=；布尔型变量的比较用关键词is。注意句末需要顿号。

Python没有switch或case这种多路分支语句，只能用字典或者if代替。

|  |
| --- |
| if x == 1 or in\_cond is True:  y = 0  elif x == 2:      y = 1  else:      y = 2 |

两种典型的Python循环结构如下，注意for循环中用到的range函数不包括终止值stop，因此最后一次循环的i=9；while循环需要自己改变循环变量。

Python的赋值操作也有基于条件判断的三目运算符if else，用法如下。注意和C语言中三目运算符的不同是Python的三目运算符条件写在中间。

|  |
| --- |
| a = 1  b = 2  minimum = a if a < b else b  print(minimum) |

## 循环

与C语言一样，Python循环也支持continue和break。

for循环语句用到的range()函数三个变量分别：起始值start，默认为0；终止值stop，必须输入；步长step，默认为1。

仅输入一个参数时定义终止值，仅输入两个参数定义起始值和终止值，输入三个参数时终止值是中间那个变量。

另外，注意终止的判断条件是小于等于终止值，因此迭代变量等于终止值时就会跳出循环。

|  |
| --- |
| for i in range(1, 10, 1):      print(i)  i = 1  while i < 10:      print(i)      i += 1 |

Python中的for循环还可以枚举遍历列表list中的所有元素或字符串的所有字符，下面的例子中，迭代变量day就遍历了week列表的每一个字符串。

注意，对列表的循环遍历实际上是对列表每一个**位置**元素的顺序读取，因此若循环过程中对列表的元素进行了修改，遍历过程中也会体现，若遍历过程中使用append()增加了元素，循环次数也会因此变多。

|  |
| --- |
| week = ["Mon", "Tue", "Wed", "Thur", "Fri", "Sat", "Sun"]  for day in week:      print(day) |

若想在循环中同时实现“列表元素遍历”和“迭代计数”，则可以用enumerate()函数。函数enumerate()在循环迭代过程中同时输出计数和元素，具体用法如下。它还有第二个变量可以用于制定计数开始值。

|  |
| --- |
| letter\_list = ["a", "g", "t", "w", "w"]  for i, item in enumerate(letter\_list):      print(i, item) |

## 基本文件操作

文件操作需要用到Python标准库os, shutil, datatime

|  |  |
| --- | --- |
| os.listdir(path) | 返回一个装有路径下所有对象名称的列表 |
| os.isfile(path) | 判断路径指向的对象是否文件 |
| os.isdir(path) | 判断路径指向的对象是否文件夹 |
| shutil.copyfile(src, dst) | 复制文件（目标目录必须包含目标文件名） |
| shutil.copytree(src, dst) | 复制文件夹（目标目录必须包含目标文件夹名） |

## 随机数生成

# Python数据分析

借助几个常用的绘图包，Python联合Excel就能进行功能强大的数据分析。数据可以从本地Excel表格中读取，也能利用爬虫或者Excel的“自网站”功能在线获取。

## 数据分析常用包

|  |  |
| --- | --- |
| matplotlib | 函数图像、分析图表绘制，导入模块pyplot，简写为plt |
| numpy | 张量运算包 |
| xlrd | excel读写 |

## Matplotlib数据绘图

官网使用手册：<https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.figure.html>

Matplotlib的操作函数与Matlab非常类似，因此可以快速上手。调用包模块的函数时，记得要在前面写上pyplot.（一般缩写为plt.）

以下示例代码是一次函数的图像绘制：导入包、定义自变量序列x（数组）、计算因变量序列y、创建绘图图窗、定义子图排列和当前图位置、定义图的标题、定义图的坐标轴名称、绘制函数y(x)、显示图窗。

|  |
| --- |
| import numpy as np  from matplotlib import pyplot as plt  x = np.arange(1, 11)  y = 2\*x + 5  plt.figure("Linear Function")  plt.subplot(211)  plt.title("Function y = 2x + 5")  plt.xlabel("x axis")  plt.ylabel("y axis")  plt.plot(x, y)  plt.show() |

### 基本绘图逻辑

在Matlab中，plot()函数运行时一定会创建一个figure图窗，而且会即时显示图窗，利用hold on则可以保证plot()绘制新图像时，当前图的旧图像不被清除，而Matplotlib的逻辑与此不同。

一些名词的解释：

·图窗：由plt.figure()创建，可以理解成画画的“画板”；在显示时，它就是一个独立窗口。

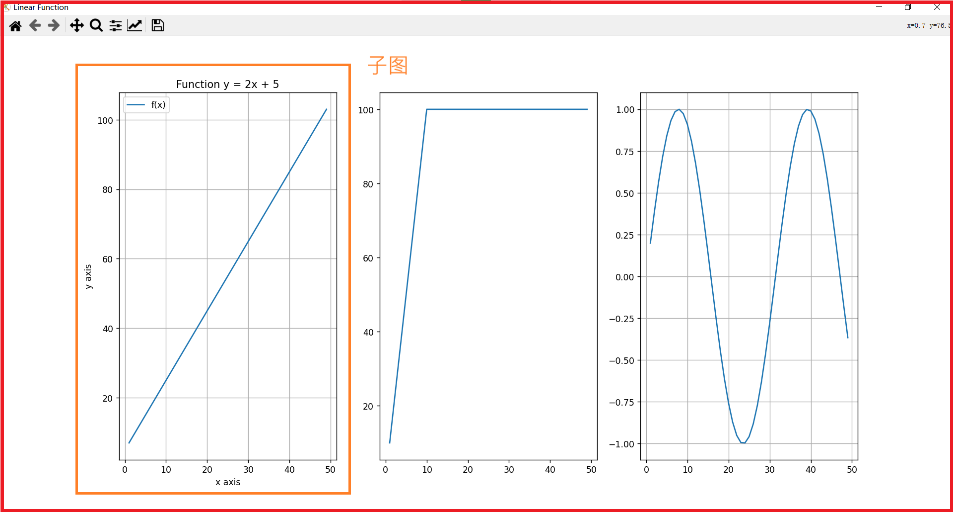
·子图：图像显示的区域，若不使用plt.subplot()，每个图窗默认仅有一个子图。

·图像：由plt.plot()在子图上绘制得，图例的指示对象。

·绘制：由进行，将图像绘制到子图上，这个过程不会显示。

·显示：plt.show()函数创建窗口显示图窗，在此之前，所有图像都是不可视的。

·激活：表示图窗、子图此时正在使用，执行的自定义和绘制函数作用于当前激活的子图。



Matplotlib绘图时，先使用plt.figure()函数创建新图窗，再使用plot()在新创建的图窗子图上绘制图像，最后再用plt.show()函数显示图窗。在show()之前，还可利用plt.xlabel()、plt.title()、plt.legend()等函数自定义子图坐标轴、标题、图例。

若要将多个子图放在一个图窗中，使用子图函数plt.subplot()，它和plt.figure()作用类似，但不会创建新图窗，而是会在激活的图窗上按照设定的行列添加子图。

### 常用函数简单说明

·plt.figure()：

1. 如果绘图前没有使用它创建图窗，即使plt.plot()、plt.show()也不会显示任何图像；
2. 程序运行到新的plt.figure()或plt.subplot()函数时，激活新的图窗或子图，新的自定义和绘制将用于下一个子图。

·plt.subplot()：

1. 同一个图窗上的子图创建时必须保证行列数设置一致，位置不能重复，否则子图会重叠

·plt.title()、plt.xlabel()、plt.ylabel()、plt.grid()：

1. 分别设置子图的标题、x坐标名称、y坐标名称、网格

·plt.plot()：

1. 在激活的子图上绘制图像，新图像不会覆盖之前已绘制的图像，即默认hold on

·plt.legend()：

1. 为子图上的各个图像加上图例，基于图像plt.plot()中定义的label标签；
2. 必须在当前子图所有图像绘制完成后再调用，否则无法正确显示图例。

·plt.show()：

1. 显示之前创建但未显示过的所有图窗，已显示过的图窗不会再被显示；
2. 执行后，程序暂停，直到打开的所有窗口被关闭。

此外，每个函数也提供了一些可选参数接口用于更细致的绘图或子图设置，如颜色、字体大小、线型、标签等。具体用法参考相关教程和官网手册。

## Excel表格读取——xlrd

xlrd是专门用于Excel表格读取的Python包，其函数操作直观，使用方便，结合Excel的数据导入功能可以进行数据分析。

下面的例子，导入xlrd包以后的三行分别实现了Excel文件data.xlsx读入、工作表storage\_number读入、读取工作表第一行第二列（行列数从0开始）的值。

|  |
| --- |
| import xlrd  table = xlrd.open\_workbook(r'data.xlsx')  number\_sheet = table.sheet\_by\_name("storage\_number")  need\_number = number\_sheet.cell\_value(0, 2) |

注意，xlrd在1.2.0以后的版本不支持xlsx格式的文件，因此安装时要指定xlrd版本。

|  |
| --- |
| conda install xlrd=1.2.0 |

### 将xlrd读入的特殊格式数转换为数值

xlrd似乎不能识别表格中带千分符（逗号）的数和百分数，会将这些数字逐字符读入为字符串str而不是数值int或float。Python的float()类型强转也无法处理这些数字字符串，因此要写一个特殊的转换函数处理这些特殊格式数。

在Excel中直接转换数字格式是无济于事的，因为如果数据是在线获取的，更新后还是会恢复原来的格式。

|  |
| --- |
| def str\_to\_float(input\_str):      if isinstance(input\_str, float) is True or isinstance(input\_str, int) is True:          return input\_str      good\_str = ""      for i in range(len(input\_str)):          if input\_str[i] == "," or input\_str[i] == "\*":              continue          if input\_str[i] == "%":              return float(good\_str) \* 0.01          else:              good\_str += input\_str[i]      return float(good\_str) |

也可以根据实际数据编写不同的数据读取预处理函数。

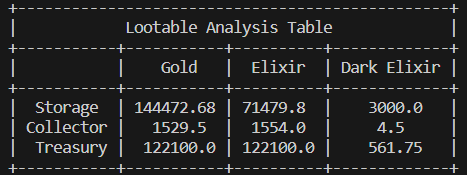
## PrettyTable命令行绘表

利用简单的几个函数，PrettyTable库可以在命令行输出简约美观的表格，此外，它还支持从数据库、CSV文件读入数据制表。

以下示例中利用PrettyTable绘制部落冲突可掠夺资源的表格，执行顺序是：创建表格、设置表格标题、设置表头、按行输入表格数据。

|  |
| --- |
| from prettytable import PrettyTable  output\_table = PrettyTable()  output\_table.title = "Lootable Analysis Table"  output\_table.field\_names = [" ", "Gold", "Elixir", "Dark Elixir"]  output\_table.add\_row(["Storage", lootable\_storage\_gold, lootable\_storage\_elixir, lootable\_storage\_dark\_elixir])  output\_table.add\_row(["Collector", lootable\_collector\_gold, lootable\_collector\_elixir, lootable\_collector\_dark\_elixir])  output\_table.add\_row(["Treasury", lootable\_treasury\_gold, lootable\_treasury\_elixir, lootable\_treasury\_dark\_elixir]) |

显示效果如下：



# Python前端开发

前端指的是程序展示数据及功能的模块，一般指的就是程序的用户交互界面UI。前端开发就是UI的设计与技术实现。

UI设计在技术实现方面可以粗略分为三步：

1. 概念设计：从功能、布局出发，用图画描述UI的布局设计，用文字描述UI的交互行为，暂不考虑具体技术实现；
2. 布局实现：创建并摆放UI控件，基本实现视觉上的设计布局；
3. 功能实现：通过回调函数、事件等方法实现控件的具体交互行为。

Python有很多包可以用于前端开发，包括Tkinter、wxPython、PyQT、PyGtk、kivy、FLTK、OpenGL、PySimpleGUI、Guietta、Pygame。

## Tkinter包

Tkinter是Python下载时自带的GUI图形用户界面开发工具，它功能较为受限，控件风格简朴，但足够简单，适合UI开发初学者。

Tkinter的控件内部参数繁杂，这里不一一介绍，可以参考CSDN上的这篇教程：《Python tkinter(GUI编程)模块最完整教程》（上、中、下）

·上：<https://blog.csdn.net/qq_48979387/article/details/125706562>

·中：<https://blog.csdn.net/qq_48979387/article/details/125806479>

·下：<https://blog.csdn.net/qq_48979387/article/details/125768994>

下面仅介绍一些Tkinter的重要概念和用法。

# Python机器学习

现阶段Python是机器学习领域最热门的语言，常用的机器学习库有Scikit-learn、Pytorch、Tensorflow等，数据处理则常要用到经典的Numpy、Pandas、Matplotlib库。

# 其他

## .py文件打包成.exe文件——pyinstaller

Python源文件格式.py的程序只能用命令行执行，很不方便，因此一般需要用pyinstaller包把它打包成一个.exe可执行文件，点击就能运行。

1. 在Prompt命令行中激活需要打包的文件test.py运行的虚拟环境，安装pyinstaller；
2. 命令行进入test.py所在目录，运行下面的代码，注意icon.ico是放在同一文件夹下，打包后生成的.exe文件所用的图标；

|  |
| --- |
| pyinstaller -F -w -i <icon.ico> <test.py> |

1. 在原目录中新创建的dist文件夹中找到打包后的test.exe文件，把它拉回到原目录中就可运行；
2. 移动.exe文件时，同一文件夹下程序调用的数据表、图片等外部文件也要一同移动，否则程序不能正常运行。