第1章　Python语言概述

https://www.bilibili.com/video/BV1oW4y1q7qv/?spm_id_from=333.337.search-card.all.click&vd_source=66b7318d84f8ecd0c0696e474c789b86

- There are two ways of constructing a software design: one way is to make it so simple that there are obviously no deficiencies; the other is to make it so complicated that there are no obvious deficiencies.
- 有两种方式构建软件设计：一种是把软件做得很简单以至于很明显没有缺陷；另一种是把它做得很复杂以至于没有明显的缺陷。

——C. A. R. Hoare

- Success in life is a matter not so much of talent and opportunity as of concentration and perseverance.
- 获得人生的成功所需要的专注与坚持不懈多过天赋与机遇。

——C.W.Wendte

1.1 Python是这样一种语言

1.1.1 Python语言的起源

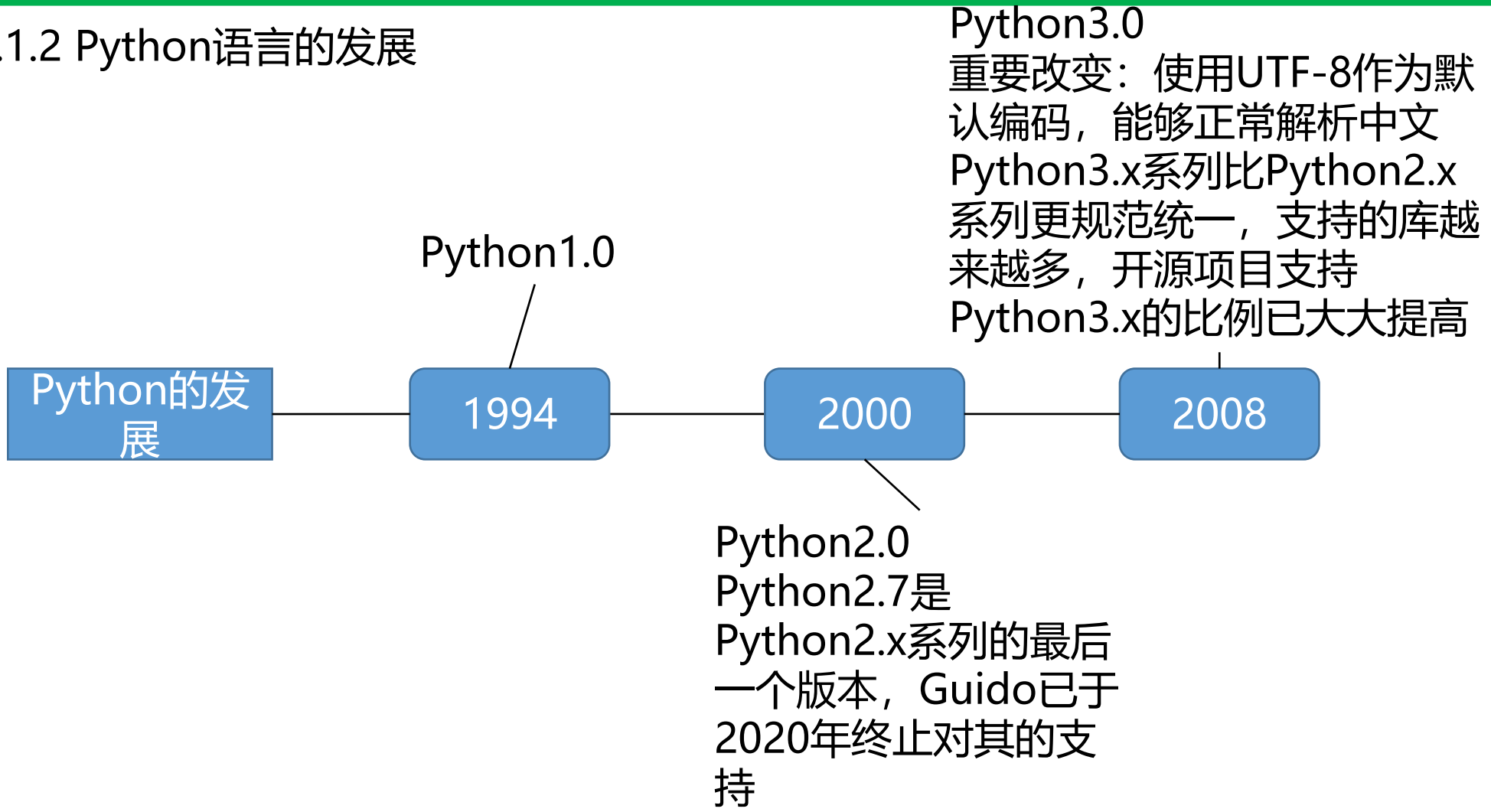
创始人：荷兰人Guido van Rossum(人称：龟叔)

创始过程：1989年的圣诞节，Guido为了打发时间，决定在ABC语言的基础上开发一款新型脚本解释程序。由于Guido非常喜欢一部名为《Monty Python’s Flying Circus》的英国肥皂剧，于是这门新语言便有了自己的名字——Python。

Python简介：1991年，Python的第一个公开版本发行。Python的源代码和解释器Cpython遵循通用公开许可证协议，其语法简洁清晰，强制用空白符作为语句缩进是其特色之一。Python具有丰富和强大的库，常被称为“胶水”语言，能够轻松地把其他语言制作的各种模块联结在一起。

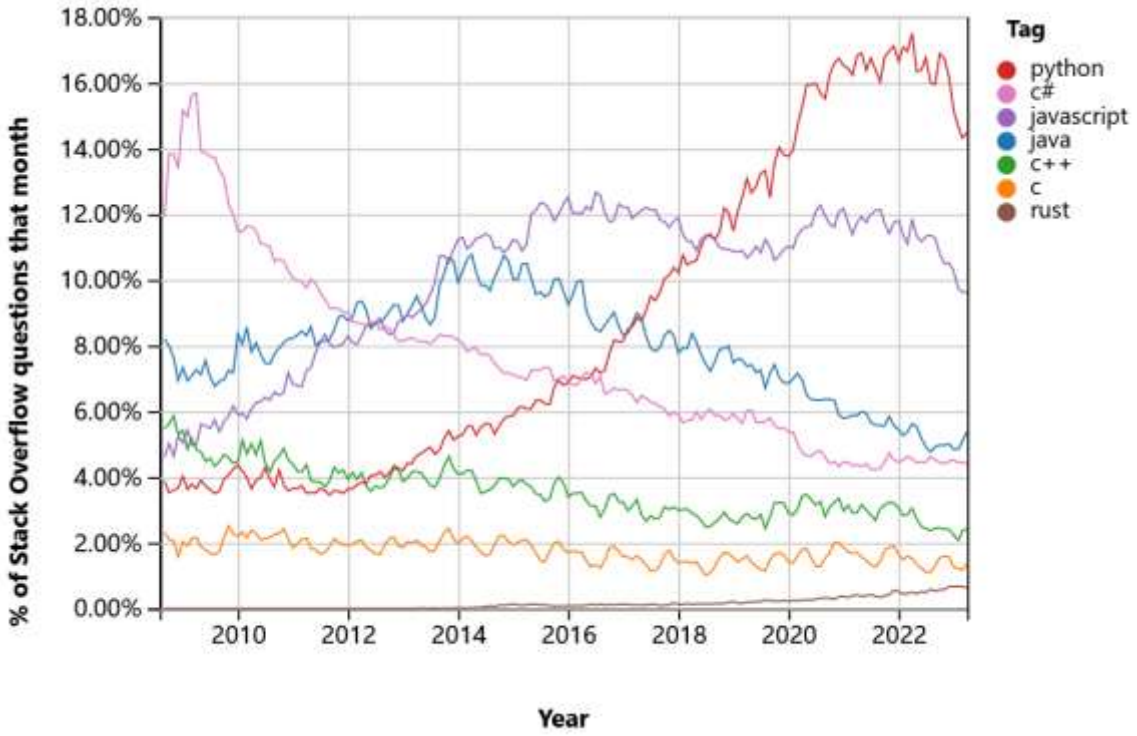
1.1 Python是这样一种语言

1.1.2 Python语言的发展



1.1 Python是这样一种语言

人生苦短，我用 Python



1.1 Python是这样一种语言

1.1.3 Python语言的特点：

1 简单、易学、免费、开源: Python简单、易学。我们可以自由发布其复制版本,阅读、修改其源代码,将其(部分)用于新软件中。
2 解释型: Python是边解释边执行的,Python解释器会将源代码转换为中间字节码形式,然后将其解释为机器语言并执行。
3 可移植: Python解释器已被移植到许多平台上,Python程序无须经过修改就可以在多个平台上运行。
4 代码规范: Python所采用的强制缩进的方式,使得其代码具有极佳的可读性。
5 面向对象: 与C++和Java等相比,Python以强大而简单的方式实现了面向对象编程。
6 胶水语言: 标准版本的Python调用C语言,并可以借助C语言的接口驱动调用所有编程语言。
7 丰富的库: Python的标准库种类繁多,可以帮助我们处理各种工作,我们不需要安装就可以直接使用这些库。
8 动态类型: Python不会检查数据类型,在声明变量时不需要指定数据类型。

1.1 Python是这样一种语言

1.1.4 Python语言的应用领域

1、Web应用开发

Python现已成为Web开发的主流语言之一，其有丰富的Web开发框架，如Django、Flask、Weppy、Zope2、Tornado、CubicWeb及Web2py等。其中，Python+Django框架最为流行，其应用范围广、开发速度快、学习门槛低，可以让程序员轻松地开发和管理复杂的Web程序。

1.1 Python是这样一种语言

1.1.4 Python语言的应用领域

2、网络安全

自2013年“棱镜门”事件后，网络安全问题越发引起世界各国的重视。Python作为高级编程语言的一种，编程简单、易学及易用，且有强大的第三方编程模块的支持，越来越受网络安全维护人员的青睐。一些常用的网络安全工具都是基于Python语言来编写的，如Scapy、Pcapy及Sulley等。

1.1 Python是这样一种语言

1.1.4 Python语言的应用领域

3、网络爬虫

在编写网络爬虫程序的众多语言中，Python起到了举足轻重的作用，其中Scrapy爬虫框架应用得非常广泛，可以应用在数据挖掘、信息处理或存储历史数据中。此外，Python还提供了多个网络爬虫框架以满足不同的应用需求，如PySpider、Crawley、Newspaper、Beautiful Soup、Grab及Cola等。

1.1 Python是这样一种语言

1.1.4 Python语言的应用领域

4、游戏开发

Python在游戏开发方面也有很多应用，如专门为游戏开发而设计的模块pygame，能够使开发者快速开发出自己的游戏。相比于Lua 或C++，Python比Lua具有更高阶的抽象能力，可以用更少的代码描述游戏的业务逻辑。例如，用C++开发游戏，有时必须写一些扩展，从而增加游戏的代码量。

1.1 Python是这样一种语言

1.1.4 Python语言的应用领域

5、数据分析

Python拥有丰富的第三方库，如Pandas、NumPy、Matplotlib及SciPy等，可以让程序编写人员完成各种数据分析需求。

1.1 Python是这样一种语言

1.1.4 Python语言的应用领域

6、人工智能

Python在人工智能、神经网络、深度学习方面，有强大而丰富的库。Python是面向对象的动态语言，且适用于科学计算，这就使得Python在人工智能方面备受青睐。目前，世界优秀的人工智能学习框架，如Google的TensorFlow、Facebook的PyTorch及开源社区的神经网络库Keras等均是用Python来实现的。

1.1 Python是这样一种语言

使用Python编写的著名应用

- Youtube - 视频社交网站
- Reddit - 社交分享网站
- Dropbox - 文件分享服务
- 豆瓣网 - 图书、唱片、电影等文化产品的资料数据库网站
- Django - 鼓励快速开发的Web应用框架
- Pylons - Web应用框架
- Zope - 应用服务器
- Plone - 内容管理系统
- Instagram - 是一款免费提供在线图片及视频分享的社交应用软件，使用Django作为后台
- TurboGears - 另一个Web应用快速开发框架
- Twisted (http://twistedmatrix.com) - Python的网络应用程序框架
- Fabric (http://fabfile.org/) - 用于管理成百上千台Linux主机的程序库
- Python Wikipedia Robot Framework - MediaWiki的机器人程式
- MoinMoinWiki (http://moinmo.in/) - Python写成的Wiki程序
- Trac (http://trac.edgewall.org/) - 使用Python编写的BUG管理系统
- Mailman (http://www.gnu.org/s/mailman) - 使用Python编写的邮件列表软件

1.1 Python是这样一种语言

核心物理领域专用库

- **经典力学**

SciPy (scipy.integrate.solve_ivp)

用途：解算常微分方程（ODE）的初值问题。

物理应用：数值模拟行星运动、弹簧振子、双摆等混沌系统。

- **电动力学与电磁学**

PyCharge

用途：计算移动点电荷产生的电磁场并进行模拟。

EMpy (Electromagnetic Python)

用途：计算电磁问题的解，如光纤模式、衍射。

1.1 Python是这样一种语言

- **量子力学**

QuTiP (Quantum Toolbox in Python)

用途：量子物理的终极工具包。用于模拟量子系统的动力学、能级、开放系统等。

ProjectQ

用途：量子计算模拟框架，可以编写和模拟量子算法。

- **热力学与统计物理**

NumPy / SciPy

用途：进行蒙特卡洛模拟、计算积分和统计量。

物理应用：模拟伊辛模型（Ising Model）、计算配分函数。

1.1 Python是这样一种语言

- **天文学与天体物理**

Astropy

用途：天文学家的“瑞士军刀”。提供天文学常用常数、单位转换、坐标转换、时间处理、光谱分析等功能。

物理应用：处理 FITS 格式的天文图像数据、计算宇宙学距离、坐标转换。

REBOUND

用途：高性能的 N 体模拟软件包。

物理应用：模拟太阳系演化、行星形成、恒星团动力学。

1.1 Python是这样一种语言

- **实验物理与数据分析**

Pandas

用途：处理结构化表格数据（类似 Excel），功能无比强大。

物理应用：整理、清洗、分析和可视化实验测量数据。例如，处理一系列在不同温度、压力下测量的数据点。

SciKit-Learn

用途：机器学习工具库。

物理应用：从实验数据中挖掘规律、分类信号、回归预测。例如，区分粒子碰撞事件、预测材料性质。

Uncertainties

用途：误差传播自动计算库。物理实验必备！

1.2 Python版本之争

- Python目前存在2.x和3.x两个系列的版本，互相之间不兼容。
- Python 2.x系列于2020年全面放弃维护和更新。
- 本课程将使用Python 3.6.0及以上

1.3 开发环境的安装与使用

- **默认编程环境：IDLE(Integrated Development and Learning Environment)**

- 在掌握了Python的环境配置后，为了能更高效地进行代码开发，建议读者选择集成开发环境（Integrated Development Environment，IDE）。常用的Python IDE有PyCharm、Anaconda(Jupter和Spyder)、Sublime Text、VS code等，**本课程选择的IDE是Spyder**。

1.3 开发环境的安装与使用

1.3.1 在Windows系统中安装Python默认环境IDLE的具体安装步骤如下：

（1）访问Python官网https://www.python.org/downloads/，进入Python的下载页面，如右图所示。单击矩形框中的链接地址，即可进入Windows系统的Python下载页面。Python同样为Windows提供了多个版本，用户可以根据自身的需求，选择适合自己需要的版本。本书以Python 3.6.3版本(其他更高版本是一样的)为例，演示Python的安装过程。

Python下载页面

1.3 开发环境的安装与使用

1.3.1 在Windows系统中安装Python默认环境IDLE的具体安装步骤如下：

（2）双击下载好的.exe文件，进入Python 3.6.3的安装界面，如下左图所示。建议在安装的过程中选中“Add Python 3.6 to PATH”复选框，如下右图所示。这样，Python路径就会自动添加到系统的环境变量中，无须再手动为Python配置环境变量。

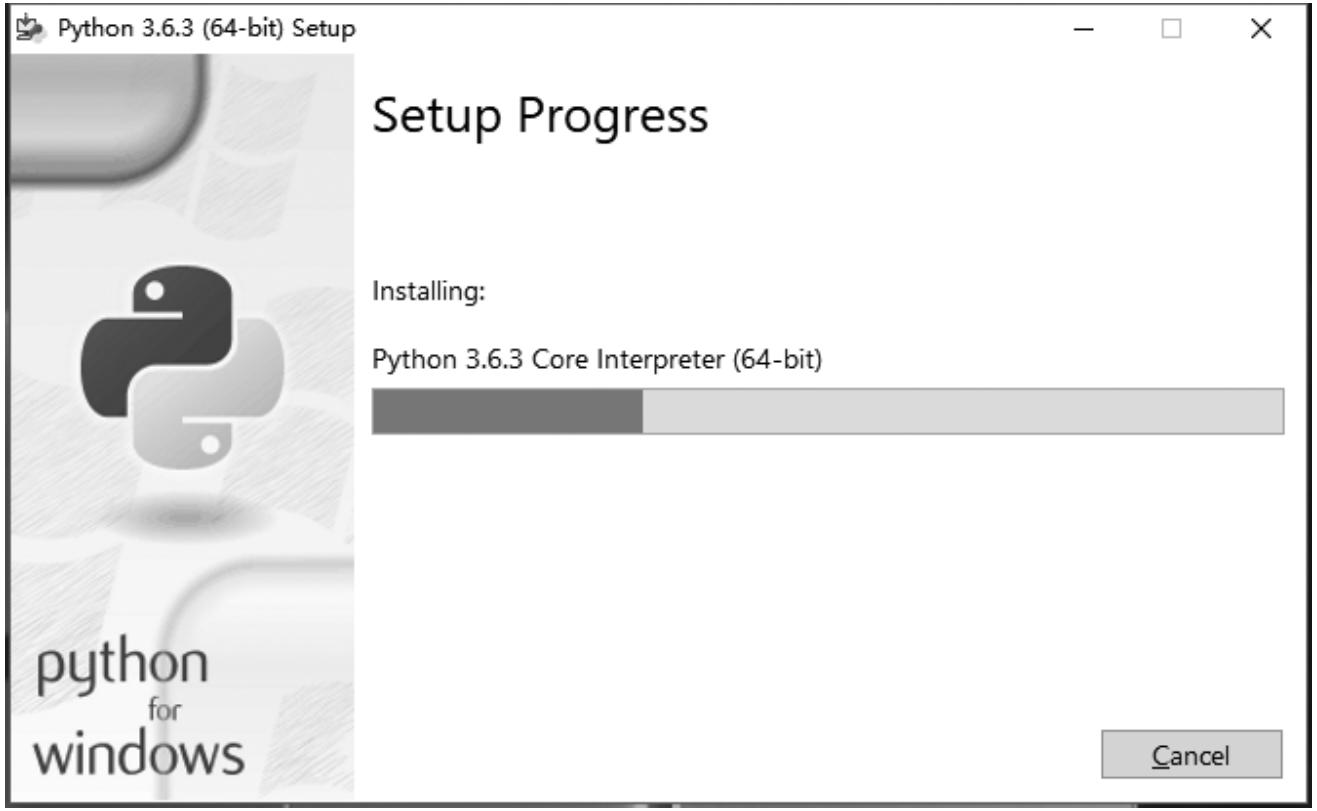
Python安装方式选择

选择添加Python路径

1.3 开发环境的安装与使用

1.3.1 在Windows系统中安装Python默认环境IDLE的具体安装步骤如下：

（3）选择第1种安装方式Install Now，安装界面如下图所示。若选择第2种安装方式Customize installation，则可以选择安装路径及可选项等内容。

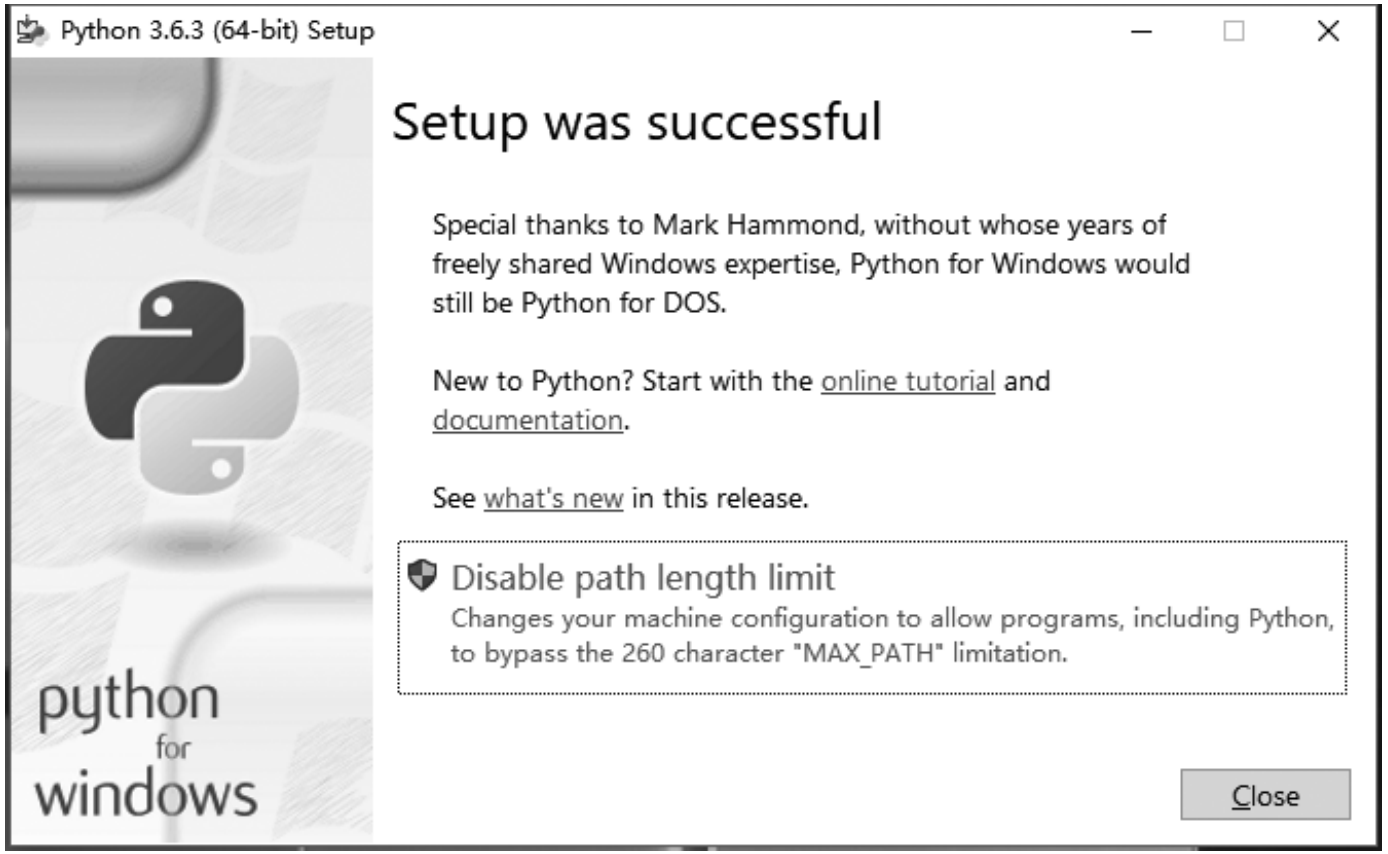


安装过程界面

1.3 开发环境的安装与使用

1.3.1 在Windows系统中安装Python默认环境IDLE的具体安装步骤如下：

（4）安装成功界面



安装成功界面

1.3 开发环境的安装与使用

1.3.1 IDLE的使用：

- 使用交互式解释环境

```
>>>print('Hello World')
Hello World
```

- 使用源文件

编辑器：使用Python自带的IDLE（语法高亮，自动缩进，自动补全）

```
#Filename: helloworld.py
print('Hello World')
```

Run → Run Module

注意：Python 是大小写敏感的

确保：每一行的开始字符前没有空格或者制表符

1.3 开发环境的安装与使用

1.3.2 在Windows系统中安装Python集成开发环境Spyder的具体安装步骤如下：

https://www.spyder-ide.org/

1.访问 Spyder 官网
2.下载 Windows 独立安装包
3.双击安装文件按向导完成安装

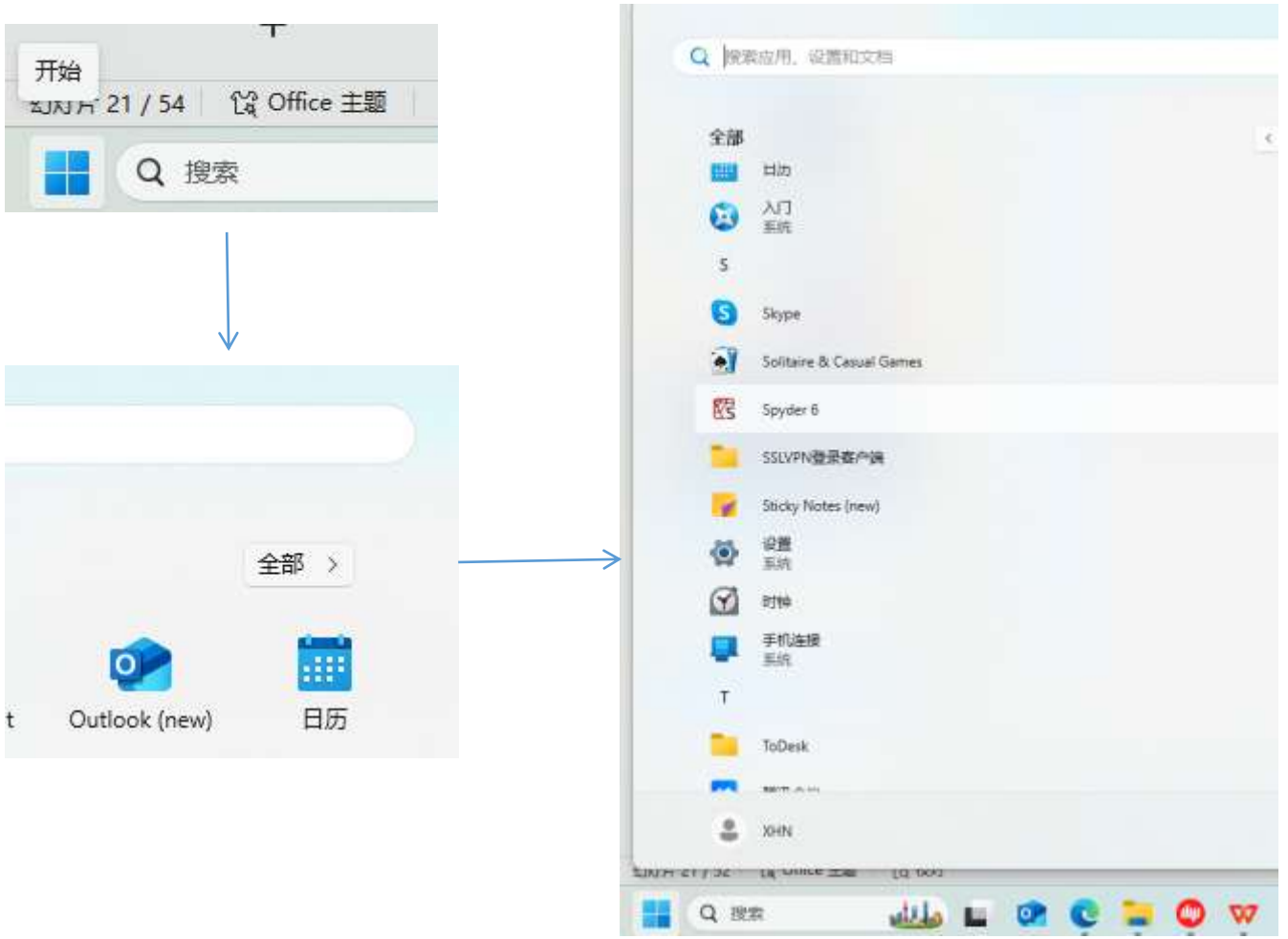
•操作系统: Windows 7+
•内存: 至少 4GB RAM（推荐 8GB）
•磁盘空间: 至少 1GB 可用空间

1.3 开发环境的安装与使用

1.3.3 Spyder的简单使用：

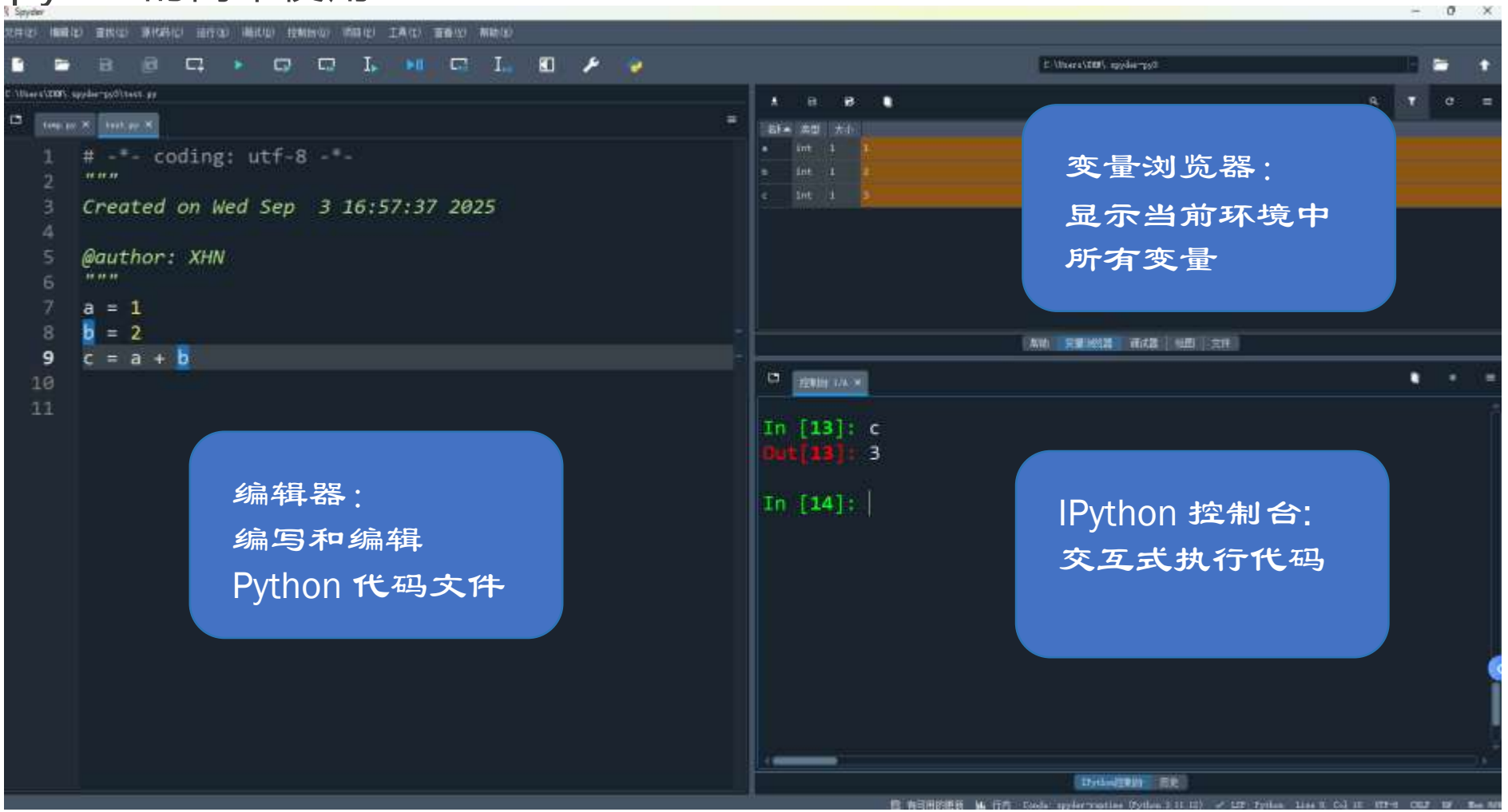
1. 打开Spyder

点击电脑左下角的开始图标，
然后点击右上角的全部，
下拉找到Spyder，
点击图标即可打开Spyder界面。



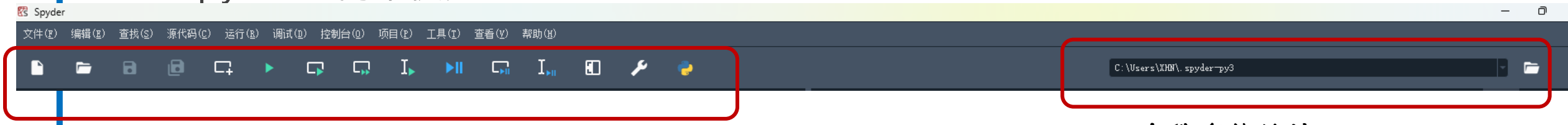
1.3 开发环境的安装与使用

1.3.3 Spyder的简单使用：



1.3 开发环境的安装与使用

1.3.3 Spyder的简单使用：



文件存放地址

一般使用这些功能键即可。

鼠标放到相应图标位置，会有功能显示。

1. 点击左1图标，新建一个test.py文档，保存位置会显示在右上角。
2. 在文档中简单输入如下代码，并点击保存。

```
a = 1
b = 2
c = a + b
```

3. 按运行文件按钮(左6)，会在变量浏览器(右上窗口)中看到三个变量a,b,c的名称、类型、大小、值等变量具体信息，并在右下Ipython控制台输入c，Out显示3。
4. 点击左5，可以新建一个单元格，输入: print(c)，点击左7，运行单元格，即可在Ipython控制台得到输出结果为：Hello。
5. 任选一个单元格，可点击右5图标，可以对此单元格进行调试。右6图标可以调试整个test.py文件。

1.4 Python编程规范与代码优化建议

（1）缩进

✓python程序是依靠代码块的缩进来体现代码之间的逻辑关系的，缩进结束就表示一个代码块结束了。

✓需要缩进的代码块：类定义、函数定义、选择结构、循环结构、with块。

✓行尾的冒号表示缩进的开始；同一个级别的代码块的缩进量必须相同。

✓一般而言，以4个空格为基本缩进单位。

```
with open(fn) as fp:
    for line in csv.reader(fp):
        if line:
            print(*line)
```

1.4 Python编程规范与代码优化建议

✓

```
if True:
    print("如果为真，输出：")
    print(True)
else:
    print("否则，输出：")
    print(False)
```

✗

```
if True:
    print("如果为真，输出：")
    print(True)
else:
 print("否则，输出：")
print(False)
```

IndentationError: unindent does not match any outer indentation level #缩进异常

1.4 Python编程规范与代码优化建议

（2）每个import语句只导入一个模块，最好按标准库、扩展库、自定义库的顺序依次导入。

```
import csv
import random
import datetime
import pandas as pd
import matplotlib.pyplot as plt
```

1.4 Python编程规范与代码优化建议

（3）最好在每个类、函数定义和一段完整的功能代码之后增加一个空行，在运算符两侧各增加一个空格，逗号后面增加一个空格。

```
    sockDianming.close()
    sockDianming = None

# 开始点名
def buttonStartDianmingClick():
    # 如果还没有注册，拒绝运行
    if int_zhuce.get() == 0:
        tkinter.messagebox.showerror('很抱歉', '请联系作者进行软件注册！')
        return
    if int_zuoye.get() == 1:
        tkinter.messagebox.showerror('很抱歉', '现在正在收作业')
        return
    if int_canDianming.get() == 1:
        tkinter.messagebox.showerror('很抱歉', '现在正在点名')
        return
    tkinter.messagebox.showinfo('恭喜', '设置成功，现在开始点名')
    #开始点名
    int_canDianming.set(1)
    global tDianming_id
    t = threading.Thread(target=thread_Dianming)
    t.start()
    tDianming_id = t.ident
buttonStartDianming = tkinter.Button(root, text='开始点名', command=buttonStartD
buttonStartDianming.place(x=20, y=60, height=30, width=100)

def buttonStopDianmingClick():
    # 如果还没有注册，拒绝运行
    if int_zhuce.get() == 0:
        tkinter.messagebox.showerror('很抱歉', '请联系作者进行软件注册！')
```

空行

有空格

没有空格

空行

1.4 Python编程规范与代码优化建议

（4）尽量不要写过长的语句。如果语句确实太长，最好使用续行符“\”，或者使用圆括号将多行代码括起来表示是一条语句(推荐！)。（建议一行最多79个字符）

```
x = 1 + 2 + 3\
    + 4 + 5\
    + 6

y = (1 + 2 + 3
     + 4 + 5
     + 6)
```

1.4 Python编程规范与代码优化建议

（5）虽然Python运算符有明确的优先级，但对于复杂的表达式建议在适当的位置使用括号使得各种运算的隶属关系和顺序更加明确、清晰。

#没有括号的写法

```
result = 2 + 3 * 4 ** 2 / 8 - 1
print(result)  # 输出：7.0
```

#有括号的清晰写法：

```
result = 2+(3*(4**2))/8-1
print(result)   # 输出：7.0
```

1.4 Python编程规范与代码优化建议

（6）注释

✓ 以符号#开始，表示本行#之后的内容为注释。

```
#在这里单行注释
print("单行注释") #或在这里
```

✓ 包含在一对三引号'''...'''或"""..."""之间且不属于任何语句的内容将被解释器认为是注释。

```
'''
多行注释
三对单引号注释
'''
print("多行注释")
```

```
"""
多行注释
三对双引号注释
"""
print("多行注释")
```

1.4 Python编程规范与代码优化建议

**（7）在开发速度和运行速度之间尽量取得最佳平衡。

✓内置对象运行速度最快，标准库对象次之，用C或Fortran编写的扩展库速度也比较快，而纯Python的扩展库往往速度慢一些。

✓在开发项目时，应优先使用Python内置对象，其次考虑使用Python标准库提供的对象，最后考虑使用第三方扩展库。

1.4 Python编程规范与代码优化建议

**（8）根据运算特点选择最合适的数据类型来提高程序的运行效率。

✓如果定义一些数据只是用来频繁遍历并且关心顺序，最好优先考虑元组。

✓如果需要频繁地测试一个元素是否存在于一个序列中并且不关心其顺序，尽量采用集合。

✓列表和元组的in操作的时间复杂度是线性的，而对于集合和字典却是常数级的，与问题规模几乎无关。

1.4 Python编程规范与代码优化建议

**（9）充分利用关系运算符以及逻辑运算符and和or的惰性求值特点，合理组织条件表达式中多个条件的先后顺序，减少不必要的计算。

**（10）充分利用生成器对象或类似迭代对象的惰性计算特点，尽量避免将其转换为列表、元组等类型，这样可以减少对内存的占用，降低空间复杂度。

1.4 Python编程规范与代码优化建议

更详细的编程规范：

https://www.python.org/dev/peps/pep-0008/

PEP 8 -- Style Guide for Python Code

PEP:	8
Title:	Style Guide for Python Code
Author:	Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>
Status:	Active
Type:	Process
Created:	05-Jul-2001
Post-History:	05-Jul-2001, 01-Aug-2013

Practice

- 缩进的作用及形式?

✓ 用缩进表示语句所属的代码块,缩进结束表示代码块的结束。
✓ 行尾的冒号表示缩进的开始，下一行缩进四个空格。

- 两种为Python程序添加注释的方法?

✓ #和独立的一对三引号

1.4 Python编程规范与代码优化建议

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

1.4 Python编程规范与代码优化建议

- Python之禅 by Tim Peters
- 优美胜于丑陋（Python 以编写优美的代码为目标）
- 明了胜于晦涩（优美的代码应当是明了的，命名规范，风格相似）
- 简洁胜于复杂（优美的代码应当是简洁的，不要有复杂的内部实现）
- 复杂胜于凌乱（如果复杂不可避免，那代码间也不能有难懂的关系，要保持接口简洁）
- 扁平胜于嵌套（优美的代码应当是扁平的，不能有太多的嵌套）
- 间隔胜于紧凑（优美的代码有适当的间隔，不要奢望一行代码解决问题）
- 可读性很重要（优美的代码是可读的）
- 即便假借特例的实用性之名，也不可违背这些规则（这些规则至高无上）
- 不要包容所有错误，除非你确定需要这样做（精准地捕获异常，不写 except:pass 风格的代码）
- 当存在多种可能，不要尝试去猜测而是尽量找一种，最好是唯一一种明显的解决方案（如果不确定，就用穷举法）
- 虽然这并不容易，因为你不是 Python 之父（这里的 Dutch 是指 Guido ）
- 做也许好过不做，但不假思索就动手还不如不做（动手之前要细思量）
- 如果你无法向人描述你的方案，那肯定不是一个好方案；反之亦然（方案测评标准）
- 命名空间是一种绝妙的理念，我们应当多加利用（倡导与号召）

翻译：赖勇浩

https://blog.csdn.net/gzlaiyonghao/article/details/2151918

1.5 安装扩展库的几种方法

Python中的库分为3类：

- **内置标准库** Python的官方库，可以直接导入程序
- **第三方扩展库** 非官方制作发布的库，需要用户安装后才能使用
- **自定义库** 用户自行编写的库，对功能性代码块进行复用

- pip在线安装（命令提示符环境，切换至Python安装目录中的scripts文件夹执行pip.exe）
- pip离线安装：https://www.lfd.uci.edu/~gohlke/pythonlibs/ (需要手动解决依赖问题)
- 如果机器上安装了多个Python开发环境，那么在一个环境下安装的扩展库无法在另一个环境下使用，需要分别安装

1.5 安装扩展库的几种方法

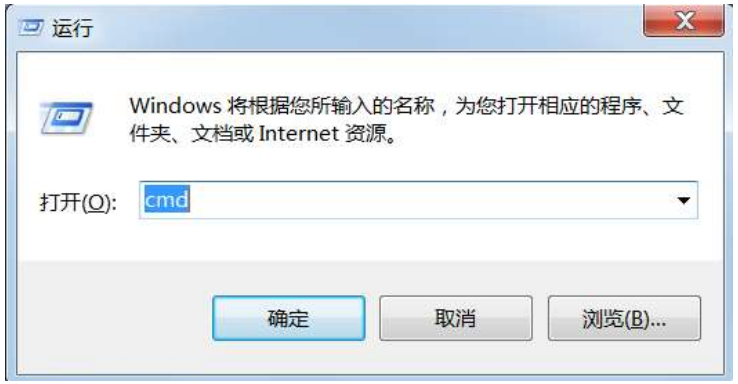
pip命令示例	说明
pip download SomePackage[==version]	下载扩展库的指定版本，不安装
pip freeze [> requirements.txt]	以requirements的格式列出已安装模块
pip list	列出当前已安装的所有模块
pip install SomePackage[==version]	在线安装SomePackage模块的指定版本
pip install SomePackage.whl	通过whl文件离线安装扩展库
pip install package1 package2 ...	依次（在线）安装package1、package2等扩展模块
pip install -r requirements.txt	安装requirements.txt文件中指定的扩展库
pip install --upgrade SomePackage	升级SomePackage模块
pip uninstall SomePackage[==version]	卸载SomePackage模块的指定版本

把SomePackage替换为实际要安装或卸载的扩展库名

1.5 安装扩展库的几种方法

1.IDLE环境在线安装：

Windows-Key + R



然后点：确定

如果python安装在C盘根目录下，直接安装即可

如果python安装在D盘根目录下，需要切换文件夹到D盘，

输入：

```
d:
cd D:\Python\Python310   # 根据你的实际安装路径修改
```

尝试安装numpy库：

```
C:\Users\XHN>pip install numpy
Collecting numpy
  Using cached numpy-1.19.5-cp36-cp36m-win_amd64.whl (13.2 MB)
Installing collected packages: numpy
Successfully installed numpy-1.19.5
```

```
C:\Users\XHN>pip list
Package                Version
---------------------- -----------
beautifulsoup4         4.12.3
certifi                2025.4.26
charset-normalizer     2.0.12
cycler                 0.11.0
idna                   3.10
jieba                  0.42.1
kiwisolver             1.3.1
matplotlib             3.3.4
mpmath                 1.3.0
numpy                  1.19.5
```

1.5 安装扩展库的几种方法

1.IDLE环境在线安装：

尝试安装numpy库：

安装：pip install numpy

```
C:\Users\XHN>pip install numpy
Collecting numpy
  Using cached numpy-1.19.5-cp36-cp36m-win_amd64.whl (13.2 MB)
Installing collected packages: numpy
Successfully installed numpy-1.19.5
```

安装成功：Succeesfully installed numpy...

查看：pip list

```
C:\Users\XHN>pip list
Package            Version
------------------ -----------
beautifulsoup4     4.12.3
certifi            2025.4.26
charset-normalizer 2.0.12
cycler             0.11.0
idna               3.10
jieba              0.42.1
kiwisolver         1.3.1
matplotlib         3.3.4
mpmath             1.3.0
numpy              1.19.5
```

1.5 安装扩展库的几种方法

2.Spyder环境在线安装：(最好为每个环境单独安装)

尝试安装numpy库：
(在控制台窗口直接输入命令即可，只是需要操作命令前加‘！’)

安装：!pip install numpy

```
控制台 1/A ×

In [2]: !pip install numpy
Collecting numpy
  Using cached numpy-1.19.5-cp36-cp36m-win_amd64.whl (13.2
MB)
Installing collected packages: numpy
Successfully installed numpy-1.19.5
```

安装成功：Succeesfully installed numpy...

查看：!pip list

```
In [3]: !pip list
Package            Version
------------------ -----------
beautifulsoup4     4.12.3
certifi            2025.4.26
charset-normalizer 2.0.12
cycler             0.11.0
idna               3.10
jieba              0.42.1
kiwisolver         1.3.1
matplotlib         3.3.4
mpmath             1.3.0
numpy              1.19.5
```

1.5 安装扩展库的几种方法

3.离线安装：(常用清华镜像)

IDLE环境：在命令提示符中输入

pip install numpy -i https://pypi.tuna.tsinghua.edu.cn/simple

Spyder环境：在控制台中输入

!pip install numpy -i https://pypi.tuna.tsinghua.edu.cn/simple

1.6 标准库与扩展库对象的导入与使用

- Python启动时仅加载了基本模块，在需要时再显式地导入标准库和第三方扩展库（需正确安装），这样可以减小程序运行的压力，并且具有很强的可扩展性。
- Python标准库在线文档：https://docs.python.org/3/library/

数学计算：

```
math       # 数学函数
random     # 随机数生成
statistics # 统计计算
```

日期和时间：

```
datetime # 日期时间处理
         #（最常用）
time     # 时间相关功能
```

文件和系统处理：

```
os         # 文件和目录操作
sys        # 系统参数和函数
pathlib    # 现代路径操作
```

1.6 标准库与扩展库对象的导入与使用

导入模块有三种方法

- **import 模块名 [as 别名]**
- **from 模块名 import 对象名 [as 别名]**
- **from 模块名 import *** #导入全部对象

1.6.1 import 模块名 [as 别名]

```
>>> import math                #导入标准库math，最常用方式
>>> help(math)                 #输出库的帮助内容
```

```
>>> import math
>>> help(math)
Help on built-in module math:

NAME
    math

DESCRIPTION
    This module is always available.  It provides access to the
    mathematical functions defined by the C standard.

FUNCTIONS
    acos(...)
        acos(x)

        Return the arc cosine (measured in radians) of x.

    acosh(...)
        acosh(x)

        Return the inverse hyperbolic cosine of x.
```

网上中文版帮助文档
https://docs.python.org/zh-cn/3.8/library/math.html

1.6.1 import 模块名 [as 别名]

>>> help(math.sin)　　#输出库中函数的帮助内容，注意sin无括号

```
>>> help(math.sin)
Help on built-in function sin in module math:

sin(...)
    sin(x)

    Return the sine of x (measured in radians).
```

1.6.1 import 模块名 [as 别名]

```
>>> import math                    #导入标准库math，最常用方式
>>> math.sin(0.5)                 #求0.5（单位是弧度）的正弦
0.479425538604203

>>> import random                  #导入标准库random
>>> n = random.random()           #获得[0,1) 内的随机小数

>>> import os.path as path         #导入标准库os.path，并设置别名为path
>>> path.isfile(r'C:\windows\notepad.exe')
True

>>> import numpy as np            #导入扩展库numpy，并设置别名为np
>>> a = np.array((1,2,3,4))       #通过模块的别名来访问其中的对象
>>> a
array([1, 2, 3, 4])
```

- 一点.表示成员访问符
- 函数名(参数) 表示函数调用，一对()是必需的

1.6.2 from 模块名 import 对象名[as 别名]

```
>>> from math import sin,cos        #只导入模块中的指定对象，访问速度略快
>>> sin(3)
0.1411200080598672
>>> from math import sin as f    #给导入的对象起个别名
>>> f(3)
0.1411200080598672
>>>from math import log as _log, exp as _exp, pi as _pi, e as _e, ceil as _ceil
```

1.6.3 from 模块名 import *

```
>>> from math import *        #导入标准库math中所有对象
>>> sin(3)                    #求正弦值
0.1411200080598672
>>> gcd(36, 18)               #最大公约数
18
>>> pi                        #常数π
3.141592653589793
>>> e                         #常数e
2.718281828459045
>>> log2(8)                   #计算以2为底的对数值
3.0
>>> log10(100)                #计算以10为底的对数值
2.0
>>> radians(180)              #把角度转换为弧度
3.141592653589793
```

在当前交互式解释器环境中运行源代码文件，如ch01.py

```
from ch01 import *
```

Practice

```
>>> ceil(1.2)
>>> math.ceil(1.2)
>>> mt.ceil(1.2)
```

为正确调用math模块中的ceil()函数，对上面三个命令，应分别使用什么语句来导入math模块？

from math import *
import math
import math as mt

习题(只提问不上交)

习　题

1.1　Python 安装扩展库常用的工具是_____和 conda，其中后者需要安装 Python 集成开发环境 Anaconda3 之后才可以使用。

1.2　Python 程序文件的扩展名主要有_____和_____两种，其中后者常用于 GUI 程序。

1.3　多选题：下面属于 Python 语言特点的有（　　）。

A. 开源　　B. 免费　　C. 跨平台　　D. 解释执行

1.4　多选题：下面导入标准库对象的语句中正确的有（　　）。

A. from math import sin

B. from random import random

C. from math import *

D. import *

1.5　多选题：下面描述中符合 Python 语言的有（　　）。

A. 跨平台　　B. 开源　　C. 解释执行　　D. 支持函数式编程

1.6　多选题：下面描述中符合 Python 语言的有（　　）。

A. 跨平台　　B. 开源　　C. 免费　　D. 扩展库丰富

习题(只提问不上交)

1.9 多选题：下面（　　）是正确的 Python 标准库对象导入方式。

A. import math.sin　　B. from math import sin

C. import math.*　　D. from math import *

1.10 判断对错：安装 Python 扩展库时只能使用 pip 工具联网在线安装，如果安装不成功就没有别的办法了。

1.11 判断对错：不管计算机上安装了多少个 Python 的版本，每个扩展库只需要安装一次，就可以在所有 Python 版本的开发环境中使用了。（ ）

习题(只提问不上交)

1.16 判断对错：如果只需要 math 模块中的 sin() 函数，建议使用 from math import sin 来导入，不推荐使用 import math 导入整个模块然后再使用 math.sin() 的方式进行调用。

1.17 判断对错：内置对象在程序中任何位置都可以直接使用，不需要导入任何内置模块、标准库或扩展库。

1.18 判断对错：Python 官方安装包没有包含任何扩展库，只有内置对象、内置模块和标准库，这些是 Python 自带的，不需要导入就可以直接使用。

1.19 判断对错：在编写 Python 程序时，对代码进行缩进只是为了好看，不缩进也不影响程序的正常执行。

1.20 判断对错：在编写程序时，不管一条语句有多长，都必须写在同一行中，不能换行。

作业(上交)

1. 使用pip install安装requests包，然后写下pip list完整的输出

2. 设三角形三条边长分别为a,b,c，计算边长c所对的角(theta)的度数
(单位为度，不是弧度)
help(math)，**表示幂运算，其他算术运算符同C语言

3.

4.