



群名称: Python程序设计2023

群 号: 682459216

Python程序设计

刘潇

机械科学与工程学院

2023年10月9日

2023秋季

这是一门什么课

- 一门程序语言课
- 一门计算机工具课
- 一门体会Python语言特点的课

课程目的

□ 本课程是机械工程专业选修课，主要介绍Python程序设计的基础知识，使学生对面向对象编程有基本的了解，结合示例掌握Python程序设计的基本方法

1. 使学生能够正确熟练地使用Python进行程序的设计，能够识读和编写较复杂程度的程序

2. 培养学生设计编程能力、创新能力和发现问题、分析问题和解决问题的能力

3. 综合运用所学知识，在今后学业和工作中扩展课堂所讲授的内容

本课程重点关注Python的语言特点

上一届学生的建议

1. 上机时间分配在各个章节结束后，不要放在一起
2. 平时作业点评
3. 前面基础知识快一点，多花时间在后面
4. 引入一些专业相关的例子讲解

课程内容

上课周数	上课日期	教学内容
第7周	10.09 (周一)	概述和基础语法
第7周	10.12 (周四)	数据类型 (数字和字符串)
第8周	10.16 (周一)	数据类型 (列表、元组、字典)
第8周	10.19 (周四)	语句控制
第9周	10.23 (周一)	函数
第9周	10.26 (周四)	函数编程
第10周	10.30 (周一)	类与对象
第10周	11.02 (周四)	模块
第11周	11.06 (周一)	串口通讯实例
第11周	11.09 (周四)	科学计算

上机

上机

注意： **上课时间**为周一3-4节（东九楼D505）和周四1-2节（东九楼D417）
上机时间第9周周一（10.23）9-10节（工程创新实践中心A座）
第10周周一（10.30）9-10节（工程创新实践中心A座）

课程安排

□ 任课老师

刘潇 副教授 (先进制造大楼西楼A503) xiaoliu@hust.edu.cn

□ QQ群



群名称: Python程序设计2023
群 号: 682459216

□ 考核

平时成绩 40% { 出勤
课后作业

课程设计 60%: 上机+思考

**作业会定期发布在QQ群，下次课上课前提交纸质版
课程设计提交截止时间11月16日，地点先进制造大楼西楼A503**

《Python程序设计》

Python概述

刘潇

机械科学与工程学院

2023年10月9日

2023秋季

主要内容

1. 概述

2. 应用领域

3. 开发环境

4. Python基础语法

什么是Python?

Python是一种结合了**解释性、互动性和面向对象**的高级脚本语言，Python语言注重的是如何解决问题而不是编程语言的语法和结构

动态
面向对象
跨平台
易集成
拓展库
易学
高生产力
代码质量高
可维护性高
...



Python的发展历史

1982年，龟叔获得荷兰阿姆斯特丹大学的数学和计算机科学的硕士学位，加入阿姆斯特丹的数学与计算机科学国家研究所参与**ABC语言**开发

1989年，龟叔在圣诞节期间创立**Python语言**，名字来源于他喜欢看著名的系列喜剧《蒙提·派森的飞行马戏团》（The Monty Python's Flying Circus）

1991年，Python公布了第一个公开发行版

2018年，龟叔正式宣布退居幕后



Guido van Rossum
吉多·范罗苏姆（龟叔）

Python开源，龟叔仁慈的独裁者

Python的设计目标

1999年，龟叔向DARPA（美国国防部高级研究计划局）提交了一条名为“Computer Programming for Everybody”的资金申请，并在后来说明了他对Python的目标：

- 一门**简单直观**的语言，并与主要竞争者一样强大
- **开源**，以便任何人都可以为它做贡献
- 代码像纯英语那样**容易理解**
- 适用于**短期**开发的日程任务

这些想法已经成为现实，Python已经成为一门流行的编程语言

<https://www.python.org/doc/essays/everybody/>

Python的设计哲学

Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

```
>>> 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!



优雅、明确、简单

Python的设计哲学

The Zen of Python, by Tim Peters

Beautiful is better than ugly.

优美胜于丑陋 (Python以编写优美的代码为目标)

Explicit is better than implicit.

明了胜于晦涩 (代码应当是简单明了和命名规范的)

Simple is better than complex.

简洁胜于复杂 (代码应当是简洁的, 不要有复杂的内部实现)

Complex is better than complicated.

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, 保持接口简洁)

Flat is better than nested.

扁平胜于嵌套 (代码应当是扁平化的, 不能有太多的嵌套)

Sparse is better than dense.

间隔胜于紧凑 (代码有适当的间隔, 不要奢望一行代码解决问题)

Readability counts.

可读性很重要 (优美的代码要是可读的)



Python的设计哲学

**Special cases aren't special enough to break the rules.
Although practicality beats purity.**

即便假借特例的实用性之名，也不可违背这些规则

**Errors should never pass silently.
Unless explicitly silenced.**

精准地捕获异常，不写except: pass风格的代码

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.

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。虽然这并不容易，因为你不是 Python之父

Python的设计哲学

Now is better than never.

Although never is often better than *right* now.

做也许好过不做，但不假思索就动手还不如不做（动手之前要想清楚）

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.

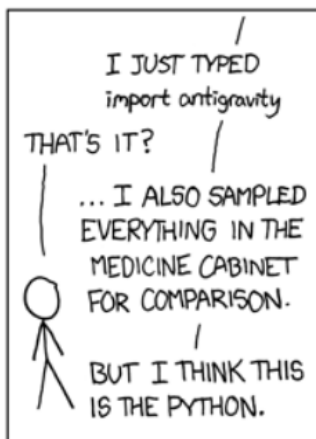
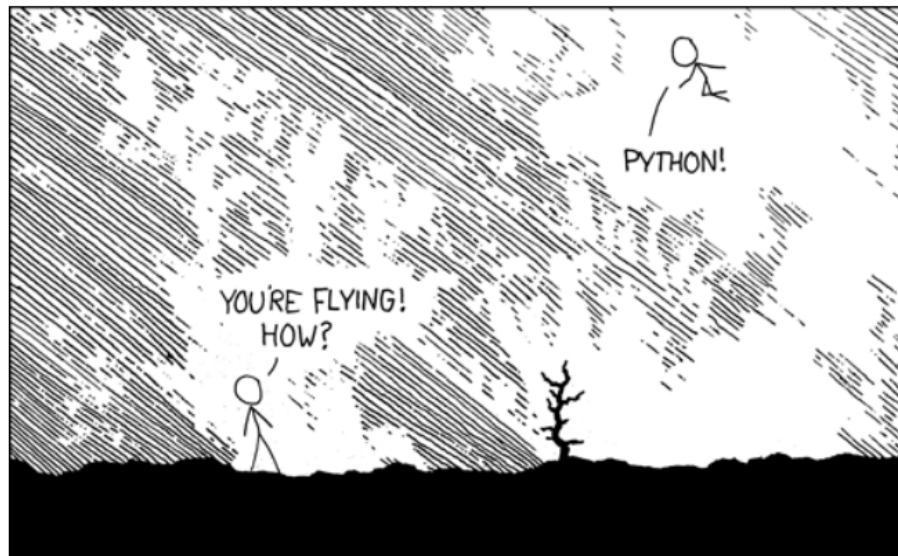
如果你无法向人描述你的方案，那肯定不是一个好方案；反之亦然

Namespaces are one honking great idea -- let's do more of those!

命名空间是一种绝妙的理念，我们应当多加利用

漫画彩蛋

import antigravity



上图:

“你在飞！怎么做到的？”

“Python!”

下左:

“我昨晚学习了 Python，一切都是那么简单”

“运行 HELLO WORLD 只需要 print “Hello World!””

下中:

“我还是不明白.....动态类型，还是空格？”

“来加入我们吧，编程又再次变得有趣起来了，Python 是一个全新的世界”

“但你是怎么飞起来的？”

下右:

“我只是输入了 import antigravity”
“就这样？”

“我还对药品柜中的所有东西进行了采样比较”（暗指他对比过多种编程语言，但还是觉得 Python 最简单）

“但我想这就是 Python。”

应用领域-系统管理

Python默认安装在Linux系统中，可以方便地编写脚本管理系统，拥有丰富的服务器管理工具，广泛应用于Linux系统管理与自动化运维

配置管理：saltstack

批量执行：fabric, saltstack

监控：Zenoss, nagios插件

虚拟化管理：python-libvirt

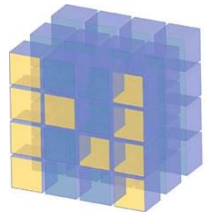
进程管理：supervisor

云计算：openstack



应用领域-数据分析

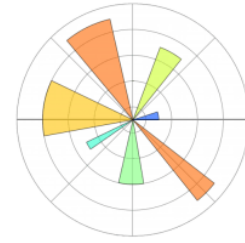
科学计算 (矩阵运算、科学计算函数、数据可视化、大数据统计分析等)



NumPy



SciPy



Matplotlib

人工智能 (机器学习、深度学习、神经网络等)



TensorFlow

theano



Keras





电视剧《天才基本法》

应用领域-Web开发



Google的核心代码是用Python语言写的，国内著名的豆瓣网也使用Python技术

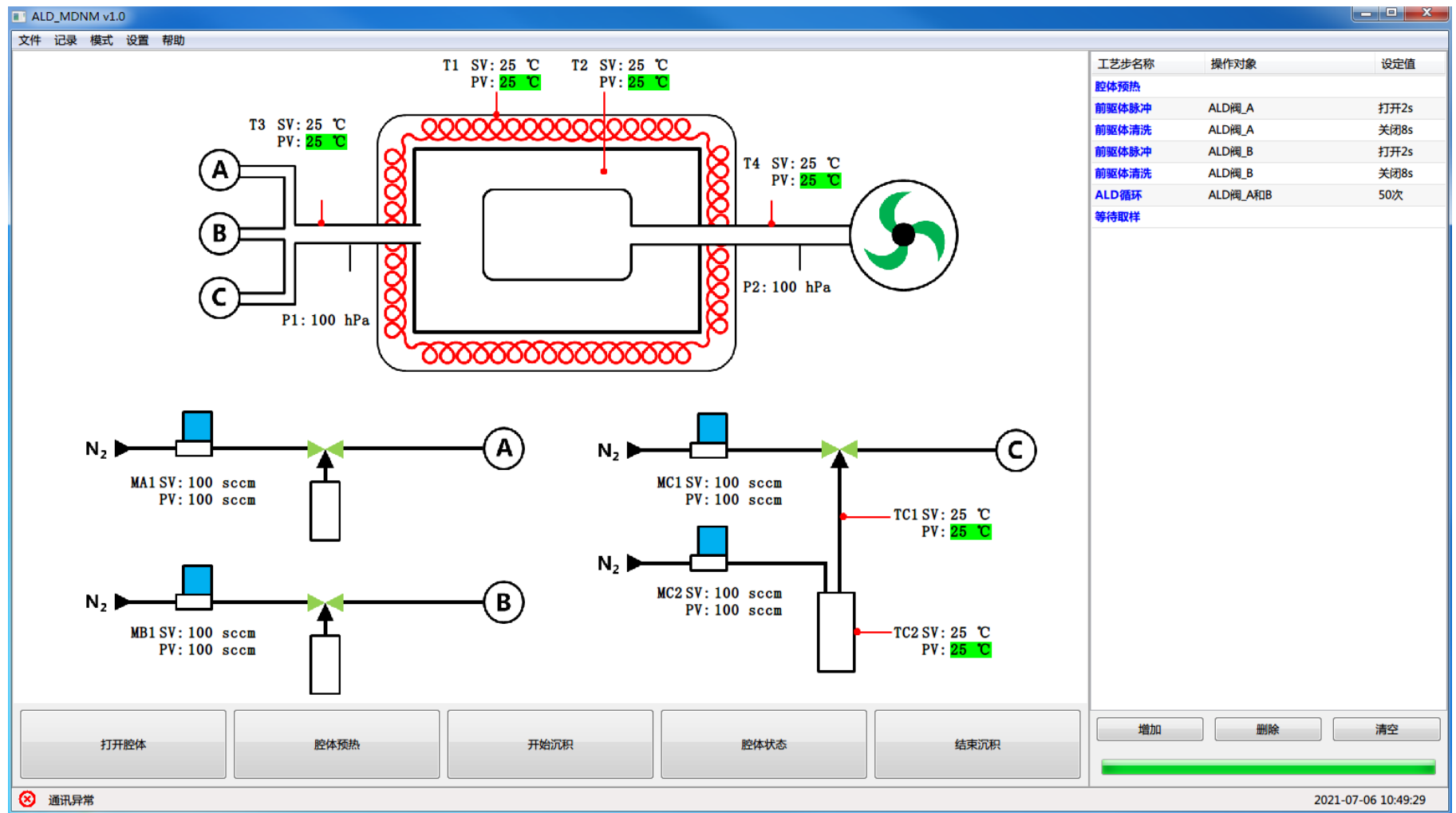
serialcom.py



serialcom



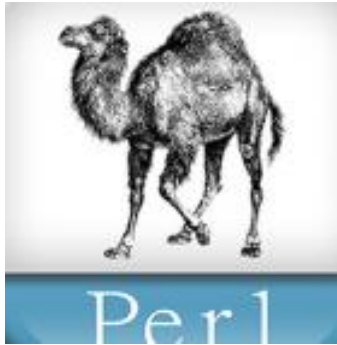
Python代码170行



Python代码数千行

与其他语言的对比-动态语言

动态语言（或称动态编程语言，Dynamic Programming Language）是指程序在运行时可以改变其结构的语言，比如新的函数可以被引进，已有的函数可以被删除等结构方面发生的变化



与其他语言的对比-静态语言

静态类型语言的优点在于基结构规范、类型安全，编译器语法检查；缺点是为需要写类型相关代码，导致不便于阅读。**动态类型**语言接近自然语言方便阅读，变量无需声明；缺点是无法语法检查。

从概念上说，动态类型语言就是类型检查在运行时完成，而静态语言的类型判断是在运行前判断（如编译阶段）

C/C++

```
a=range(10)
b=map(lambda x:x*x,a)
print b
```

```
#include "iostream.h"
int main()
{
    int i=0;  int a[10],b[10];
    while(i<10)
    {
        a[i]=i;    i+=1;
    }
    for (i=0;i<10;i++)
    {
        b[i]=a[i]*a[i];
        cout<<b[i];
    }
    return 0;
}
```

- 执行效率高
- 资源丰富
- 过分复杂

Python有大量用
C/C++编写的库！

与其他语言的对比-静态语言

静态类型语言的优点在于基结构规范、类型安全，编译器语法检查；缺点是为需要写类型相关代码，导致不便于阅读。**动态类型**语言接近自然语言方便阅读，变量无需声明；缺点是无法语法检查。

从概念上说，动态类型语言就是类型检查在运行时完成，而静态语言的类型判断是在运行前判断（如编译阶段）

Java

Java	Python
<pre>public class HelloWorld { public static void main (String[] args) { System.out.println("Hello, world!"); } }</pre>	<pre>print "Hello, world!" print("Hello, world!") # Python version 3</pre>
Java	Python
<pre>public class Employee { private String myEmployeeName; private int myTaxDeductions = 1; private String myMaritalStatus = "single"; //----- constructor #1 ----- public Employee(String EmployeeName) { this(employeeName, 1); } //----- constructor #2 ----- public Employee(String EmployeeName, int taxDeductions) { this(employeeName, taxDeductions, "single"); } //----- constructor #3 ----- public Employee(String EmployeeName, int taxDeductions, String maritalStatus) { this.employeeName = employeeName; this.taxDeductions = taxDeductions; this.maritalStatus = maritalStatus; } }</pre>	<pre>class Employee(): def __init__(self, employeeName, taxDeductions=1, maritalStatus="single"): self.employeeName = employeeName self.taxDeductions = taxDeductions self.maritalStatus = maritalStatus ... In Python, a class has only one constructor. The constructor method is simply another method of the class, but one that has a special name: __init__</pre>

- 相比 C++ 语法简洁优雅
- 去掉 C++ 中 C 的部分（C++-），纯面向对象

与其他语言的对比

TIOBE Index for August 2023

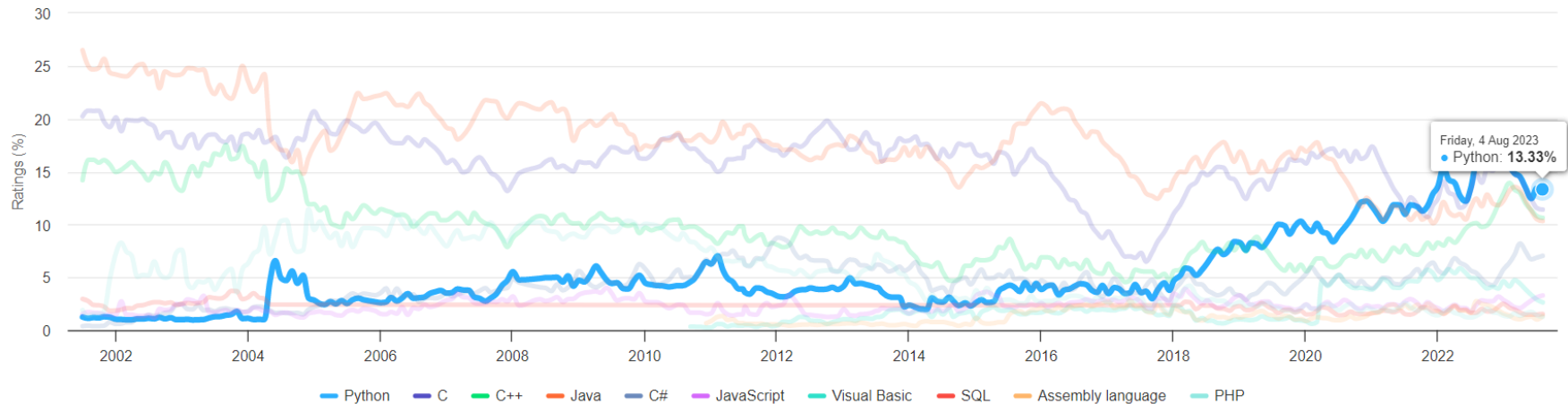
August Headline: Julia enters the TIOBE index top 20 for the first time

Aug 2023	Aug 2022	Change	Programming Language		Ratings	Change
1	1			Python	13.33%	-2.30%
2	2			C	11.41%	-3.35%
3	4	▲		C++	10.63%	+0.49%
4	3	▼		Java	10.33%	-2.14%
5	5			C#	7.04%	+1.64%

开发语言排行榜

TIOBE Programming Community Index

Source: www.tiobe.com



<https://www.tiobe.com/tiobe-index/>

Python开发环境

Python可用于Windows、Unix/Linux、 macOS等多平台

Python标准版下载

<https://www.python.org/downloads/>

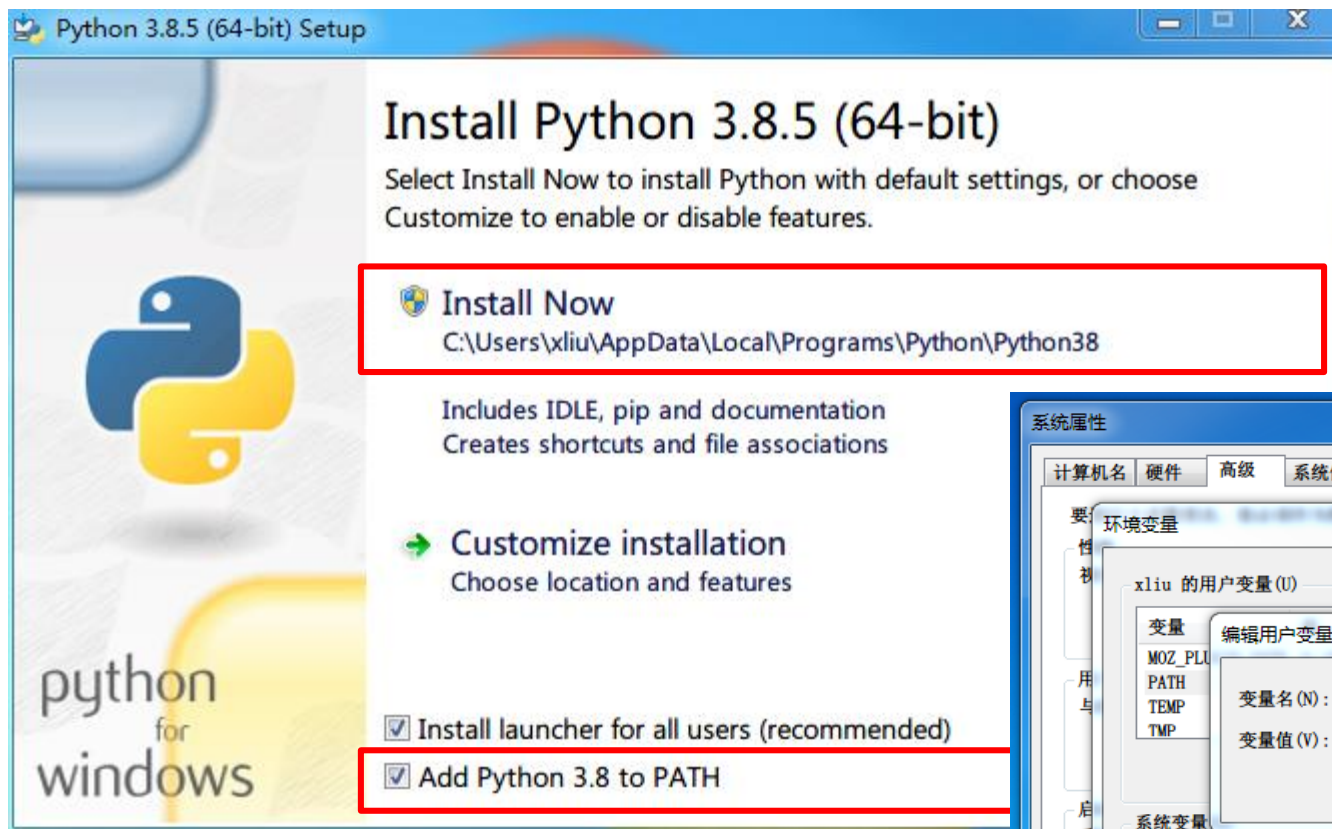


Active Python Releases

For more information visit the [Python Developer's Guide](#).

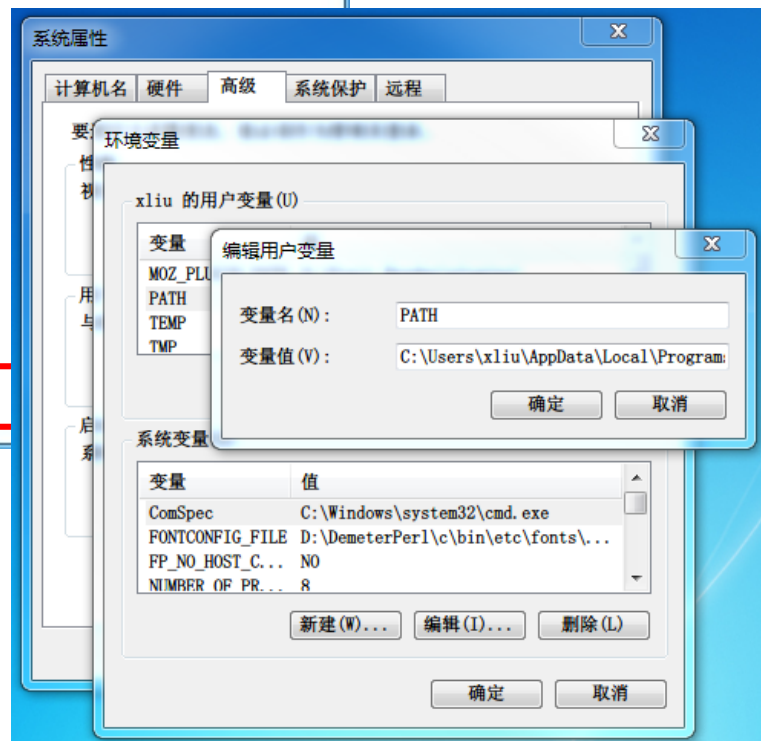
Python version	Maintenance status	First released	End of support	Release schedule
3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	bugfix	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537
2.7	end-of-life	2010-07-03	2020-01-01	PEP 373

Python安装

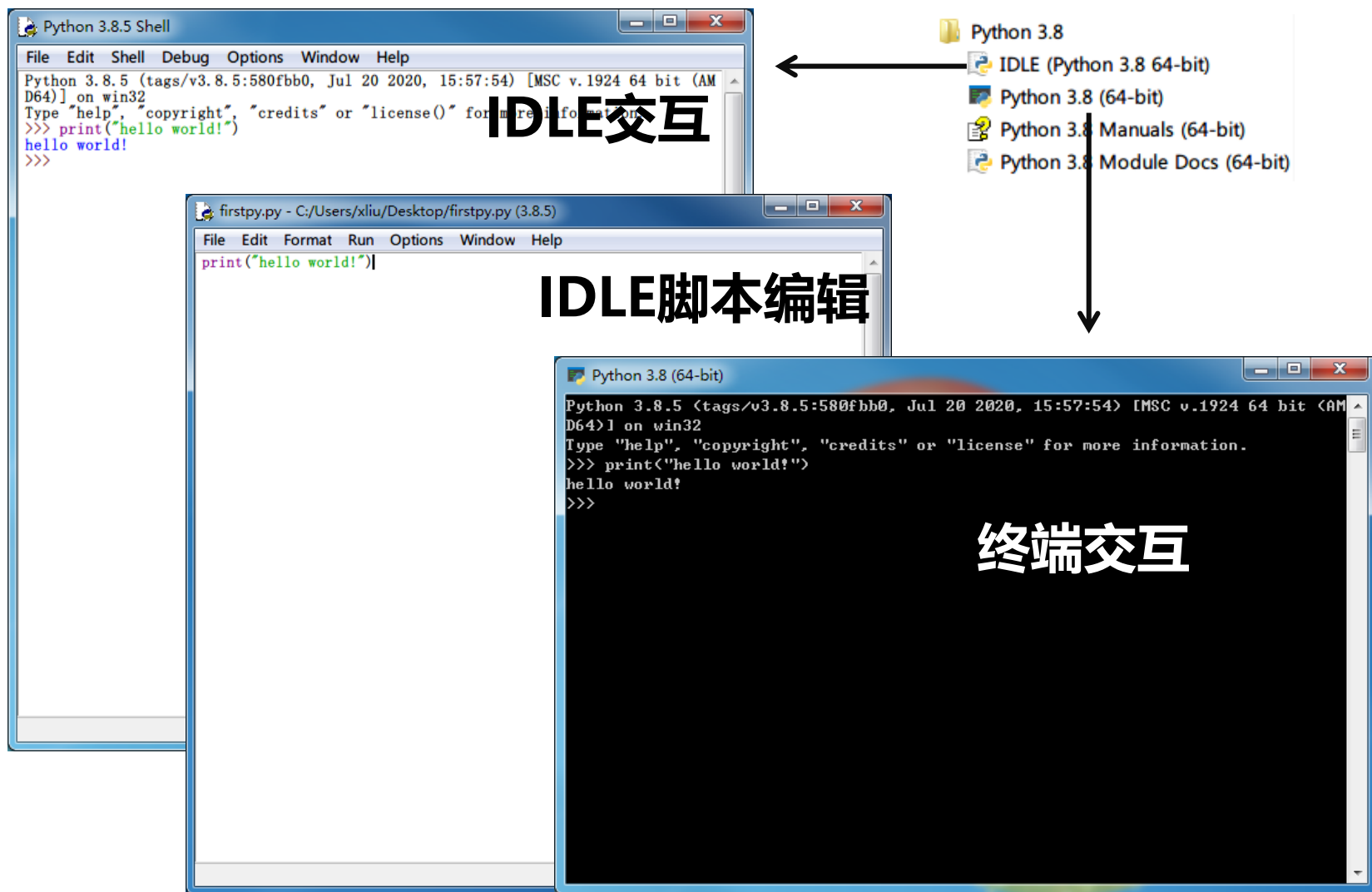


确认PATH环境变量

C:\Users\xliu\AppData\Local\Programs\Python\Python38\Scripts\
C:\Users\xliu\AppData\Local\Programs\Python\Python38\



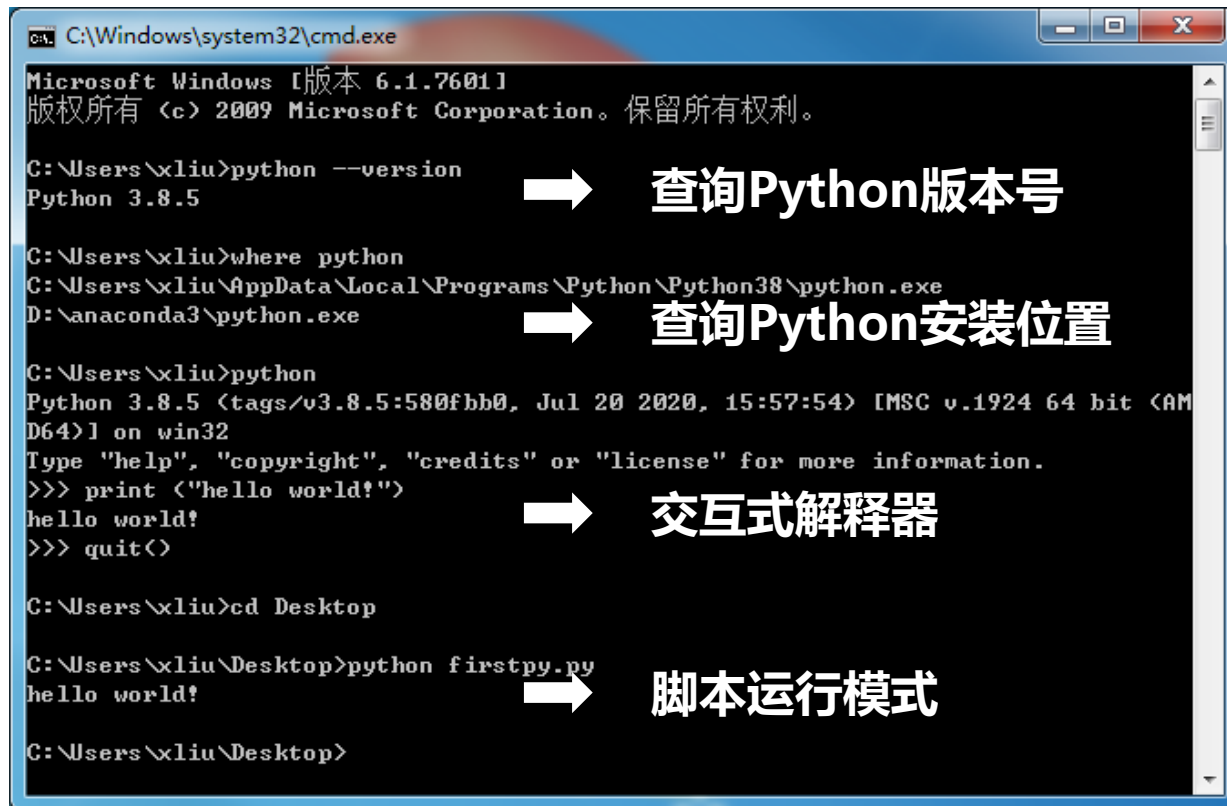
Python运行



Python IDE推荐: PyCharm、Vim、Eclipse、Sublime Text ...

终端运行

Windows 终端



```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\xliu>python --version
Python 3.8.5

C:\Users\xliu>where python
C:\Users\xliu\AppData\Local\Programs\Python\Python38\python.exe
D:\anaconda3\python.exe

C:\Users\xliu>python
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print ("hello world!")
hello world!
>>> quit()

C:\Users\xliu>cd Desktop

C:\Users\xliu\Desktop>python firstpy.py
hello world!

C:\Users\xliu\Desktop>
```

→ 查询Python版本号

→ 查询Python安装位置

→ 交互式解释器

→ 脚本运行模式

```
[xliu@master01 ~]$ python --version
Python 2.7.12
[xliu@master01 ~]$ which python
~/bin/python
[xliu@master01 ~]$ python
Python 2.7.12 (default, Jan 8 2020, 16:44:02)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux2
Type "help", "copyright", "credits" or "license()" for more information.
>>> print ("hello world!")
hello world!
>>> quit()
[xliu@master01 ~]$
```

Linux终端

Python发行版

Anaconda是在 conda（一个包管理器和环境管理器）上发展出来的，附带了 conda、Python 和 150 多个科学包及其依赖项。

- **包管理**：安装、卸载和更新包
- **环境管理**：建立不同python版本和包版本的运行环境

可用于Windows、Mac OS X 和 Linux多个平台，支持32位和64位操作系统



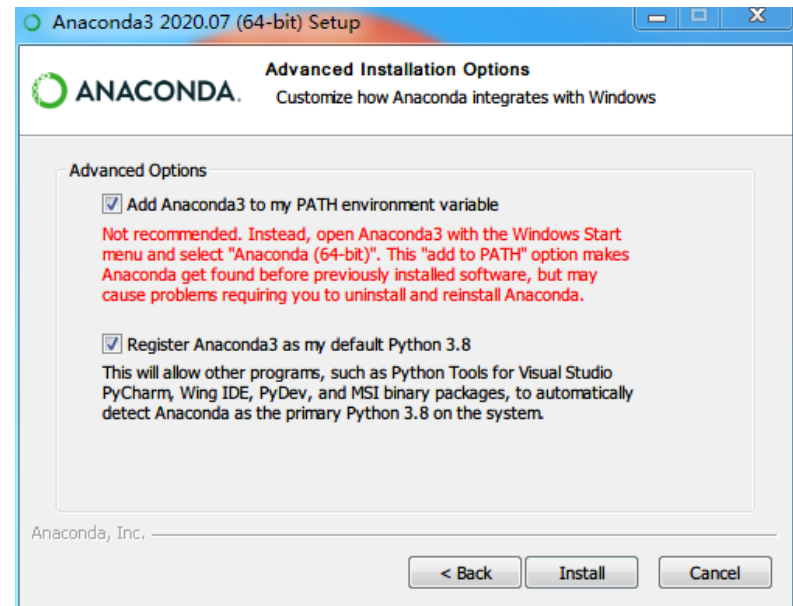
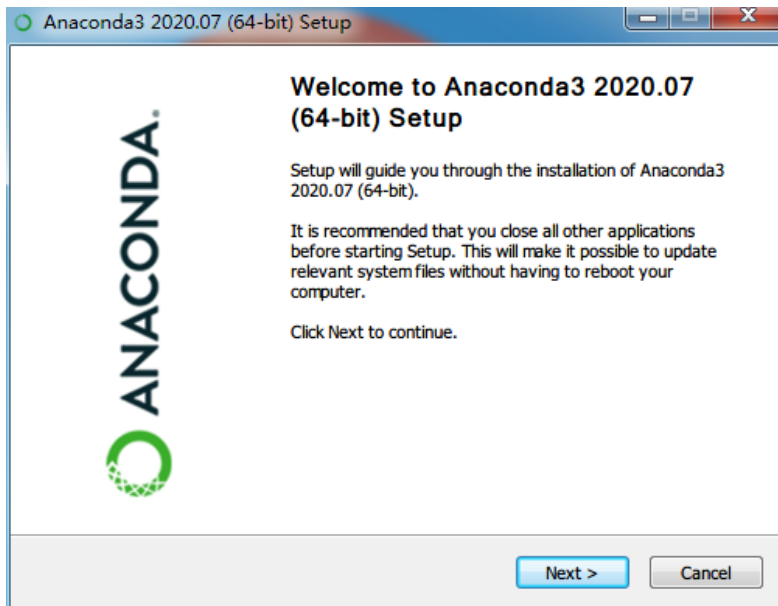
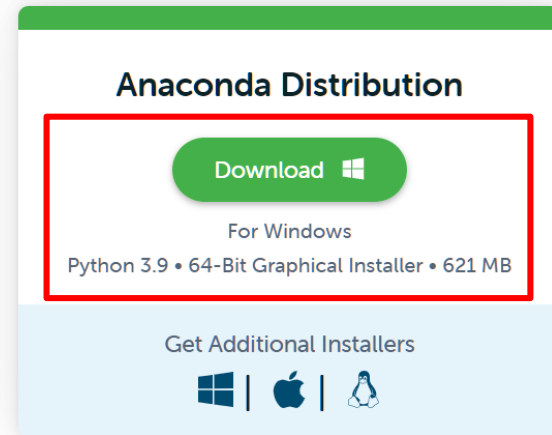
Anaconda下载和安装

<https://www.anaconda.com/products/individual>

Individual Edition is now

ANACONDA DISTRIBUTION

The world's most popular open-source Python distribution platform



Anaconda运行

Web IDE

Anaconda Navigator (64-bit)

- Anaconda Navigator (anaconda3)
- Anaconda PowerShell Prompt (anaconda3)
- Anaconda Prompt (anaconda3)
- Jupyter Notebook (anaconda3)
- Reset Spyder Settings (anaconda3)
- Spyder (anaconda3)

Sign in to Anaconda Cloud

Applications on base (root)

Environments

Learning

Community

Documentation

Developer Blog

Launch

Launch

Launch

Launch

Launch

Launch

Install

Install

Install

终端

本地IDE

Jupyter notebook

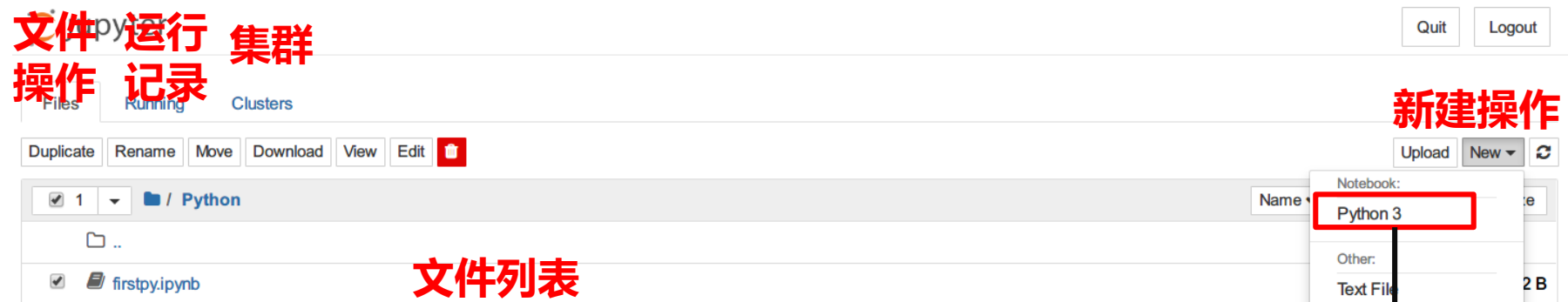
Jupyter notebook (<http://jupyter.org/>) 是一种 Web 应用，能让用户将说明文本、数学方程、代码和可视化内容全部组合到一个易于共享的文档中。



Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.

Jupyter notebook运行 (Web应用)

文件界面



代码界面



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。



Anaconda创建Python2.7环境

The screenshot shows the Anaconda environment manager interface. On the left, a list of installed packages is visible, including networkx, nltk, nose, notebook, numba, numexpr, numpy, numpy-base, numpydoc, olefile, openpyxl, openssl, packaging, pandas, pandoc, pandocfilters, paramiko, parso, and partd. A red box highlights the 'Create' button at the bottom left. In the center, a 'Create new environment' dialog box is open, showing the environment name 'python27', the location 'D:\anaconda3\envs\python27', and the packages 'Python 2.7' and 'R'. A red box highlights the 'Create' button in the dialog. On the right, a list of installed packages is visible, including ca-certificates, certifi, pip, python, setuptools, sqlite, vc, vs2008_runtime, wheel, and wincertstore. A red box highlights the 'python27' environment in the list. A large black box with white text 'Python 2.7' is overlaid on the right side. At the bottom, a terminal window shows the command 'python --version' and the output 'Python 2.7.18 :: Anaconda, Inc.'. Below the terminal window, the text 'print "hello world!"' is displayed.

base (root)

Search Environments

Installed Channels Update index... Search Packages

Name Description

- networkx Python package for creating and manipulating complex networks
- nltk Build python programs to work with human language data
- nose Nose extends unittest to make testing easier
- notebook Jupyter notebook
- numba Just-in-time compiler for Python
- numexpr Fast numerical expression evaluator using NumPy
- numpy Fundamental package for scientific computing with Python
- numpy-base Base package for NumPy
- numpydoc Sphinx documentation for NumPy
- olefile Python library to read/write excel 2003/2007 xlsm/xls/xltx files
- openpyxl Python library to read/write excel 2010 xlsm/xlsx files
- openssl Openssl is an open-source implementation of the SSL and TLS protocols
- packaging Core utilities for Python packaging
- pandas High-performance, easy-to-use data structures for data analysis
- pandoc Universal markup converter
- pandocfilters A python module for pandoc filters
- paramiko Ssh2 protocol library
- parso A python parser
- partd Data structure for on-disk

Create Clone Import Remove

base (root)

Search Environments

Installed Channels Update index... Search Packages

Name Description

- ca-certificates Certificates for use with other packages.
- certifi Python package for providing mozilla's ca bundle.
- pip Pypa recommended tool for installing python packages
- python General purpose programming language
- setuptools Download, build, install, upgrade, and uninstall python packages
- sqlite Implements a self-contained, zero-configuration, sql database engine
- vc A meta-package to impose mutual exclusivity among software built with Visual C++
- vs2008_runtime Visual Studio 2008 runtime
- wheel A built-package format for python.
- wincertstore Python module to extract ca and cert certs from windows' cert store

python27

Python 2.7

Create new environment

Name: python27

Location: D:\anaconda3\envs\python27

Packages: ☒ Python 2.7 ☐ R

Cancel Create

C:\Windows\system32\cmd.exe - python

```
<python27> C:\Users\xliu>python --version
Python 2.7.18 :: Anaconda, Inc.

<python27> C:\Users\xliu>python
Python 2.7.18 [Anaconda, Inc.] (default, Apr 23 2020, 17:26:54) [MSC v.1500 64 bit
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print "hello world!"
hello world!
>>>
```

print "hello world!"

Python模块

标准模块： math, cmath, sys, os等

第三方模块：

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, ...

使用numpy模块

矩阵相乘(A为m*p的矩阵, B为p*n的矩阵)

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix} \quad B = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix}$$

$$AB = \begin{bmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} + a_{1,3}b_{3,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} + a_{1,3}b_{3,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} + a_{2,3}b_{3,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} + a_{2,3}b_{3,2} \end{bmatrix}$$

$$(AB)_{i,j} = \sum_{k=1}^p a_{ik}b_{kj} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{ip}b_{pj}$$

使用numpy模块

module.function的方式使用模块中的函数

强大的Python模块

$$A = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}$$

$$B = \begin{bmatrix} 2 & 2 \\ -8 & 3 \end{bmatrix}$$

np.array: 定义矩阵并赋值
np.dot: 矩阵的乘法运算
np.linalg.inv: 矩阵求逆
np.transpose: 矩阵转置

计算矩阵A和矩阵B的乘积

```
1 # 加载模块
2 import numpy
3
4 # 定义矩阵
5 A = numpy.array([[3, 2], [2, 6]])
6 B = numpy.array([[2, 2], [-8, 3]])
7
8 # 计算乘积并输出
9 print(numpy.dot(A, B))
10 print(A*B)
```

`[[-10 12]
 [-44 22]]`
`[[6 4]
 [-16 18]]`

矩阵函数	说明
np.sin(a)	对矩阵a中每个元素取正弦,sin(x)
np.cos(a)	对矩阵a中每个元素取余弦,cos(x)
np.tan(a)	对矩阵a中每个元素取正切,tan(x)
np.arcsin(a)	对矩阵a中每个元素取反正弦,arcsin(x)
np.arccos(a)	对矩阵a中每个元素取反余弦,arccos(x)
np.arctan(a)	对矩阵a中每个元素取反正切,arctan(x)
np.exp(a)	对矩阵a中每个元素取指数函数,e ^x
np.sqrt(a)	对矩阵a中每个元素开根号√x

运算符	说明
+	矩阵对应元素相加
-	矩阵对应元素相减
*	矩阵对应元素相乘
/	矩阵对应元素相除，如果都是整数则取商
%	矩阵对应元素相除后取余数
**	矩阵每个元素都取n次方，如**2：每个元素都取平方

Python基础语法

注释

单行注释使用 # 号：

```
print(" Hello, Python!" ); # 单行注释
```

多行注释三个双引号或三个单引号：

""" 这是多行注释，使用双引号。 这是多行注释，使用双引号。 这是多行注释，使用双引号。 """

''' 这是多行注释，使用双引号。 这是多行注释，使用双引号。 这是多行注释，使用双引号 '''

程序员应遵守的首要戒律是 “汝应注释”

缩进

Python使用缩进来表示代码块，而不是使用大括号 {} 来表示代码块，缩进在python里有重要的语法意义。

- 从第一行开始，行首的空白非常重要
- 同一层次的语句（块）必须有**相同的缩进**
- 制表符和空格不要混用
- 一贯**坚持唯一的缩进风格**（单个制表符、两个或四个空格等）

建议使用四个空格来缩进代码，不建议使用制表符。

缩进

Python基础语法

```
1  #这是注释
2  x = 1
3  y = 1
4  if x == y:
5      print("x is equal to y")
6      ''' if y != 0:
7          print("therefore, x/y is", x/y) '''
8  elif x < y:
9      print("x is smaller than y")
10 else:
11     print("y is smaller than x")
12 print("thanks!")
```

双引号

多行注释也需要注意缩进

四个空格缩进

输入(input)&输出(print)

input函数可以实现等待并接收命令行中的用户输入，input读入的内容是**字符串类型**

print在Python 3.x中是一个函数（在Python 2.x中print是一条语句，无须将要打印的内容作为参数放在圆括号中）

`print(*objects, sep=' ', end='\n', file=sys.stdout)`

输出对象 间隔符号 结尾符号 文件对象

输入输出

```
1 a = int(input("a = "))
2 b = int(input("b = "))
3 print(a, b)
4 print(a, b, sep=", ", end="")
5 print("没有换行")
```

```
a = 1
b = 2
1 2
1, 2没有换行
```

赋值

Python没有声明变量的命令，变量（标识符）是在赋值的同时创建的，只能由字母、数字和下划线(_)构成，区分大小写，且不能以数字打头

赋值

```
1  #一行赋值多个变量
2  num1, num2, num3 = 1, 2, 3
3  num4 = num5 = num6 = 4
4  print(num5+num2)
5
6  #字符串赋值
7  str1 = "我是"
8  str2 = "字符串"
9  print(str1+str2)
```

变量不需要使用任何特定类型声明，甚至可以在设置后更改其类型

字符串变量可以使用单引号或双引号进行声明

变量就是变量，它没有类型，我们所说的"类型"是变量所指的内存中对象的类型

保留字

保留字是 Python 中一些已经被赋予特定意义的单词，**不能用保留字作为标识符**给变量、函数、类、模板以及其他对象命名。

and	as	assert	break	class	continue
def	del	elif	else	except	finally
for	from	False	global	if	import
in	is	lambda	nonlocal	not	None
or	pass	raise	return	try	True
while	with	yield			

区分大小写

表达式

产生或计算新数据值的程序代码片段称为“**表达式**”，将表达式转换为基础数据类型的过程称为求值

表达式

```
1 32
2 "hello world!"
3 x = 1+2+5
4 print(x)
```

字面量是最简单的表达式

表达式可看成是数据与运算符（加减乘除等）的组合

8

标识符

x

=

1+2+5

表达式

表达式赋值

Python的基础数据类型有哪些？

小结

- 解释性、互动性和面向对象的高级脚本语言
- 利用Anaconda搭建Python开发环境
- 强大的Python模块
- Python基础语法，注释、缩进、输入输出、赋值等

下一节：Python数据类型（数字和字符串）