

Python程序设计

陈远祥

chenyxmail@gmail.com

北京邮电大学 电子工程学院





课程相关信息

- 学分/学时：2/32
- 上课时间：每周一10、11节（18：30-20：20）
- 上课地点：校本部教三楼3-308
- 适用专业：公选课，全校所有专业
- 任课教师：陈远祥, chenyxmail@gmail.com



课程简介

■ 课程目标

- ✓ 能使用python语言编写简单的程序
- ✓ 掌握程序设计的基本原理方法

■ 课程内容

Python基本程序设计、Python编程基础、Python程序流程控制、函数和模块的调用、面向对象编程、Python的异常处理以及Python数据库编程和交互式图像编程



课程简介

- 考核方式：平时成绩+期末大作业

- 参考书目：

- ✓ 《Python编程快速上手-让繁琐工作自动化》；
[美] Al Sweigart（斯维加特）著；王海鹏 译，
人民邮电出版社

- ✓ 《Python语言程序设计基础》嵩天，礼欣，黄
天羽 著，高等教育出版社

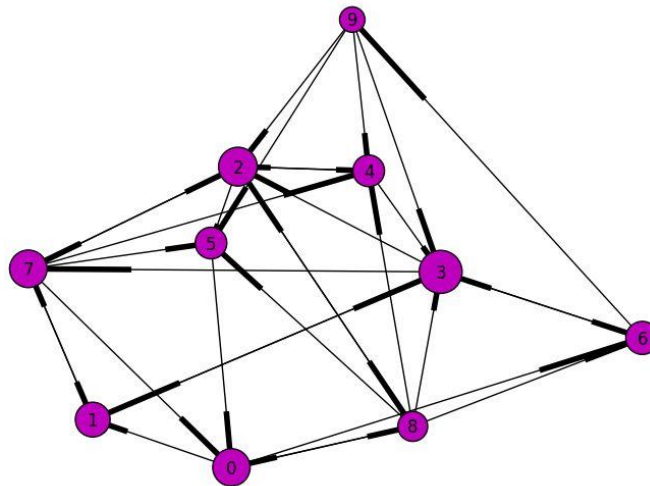


学习技巧

- 跟上进度：跟随课程进度，完成课程要求的学习内容
- 重视练习：请课后进行程序设计练习，熟能生巧



程序设计语言基础知识





程序设计语言

- 程序设计语言，也叫编程语言，是计算机能够理解和识别操作的一种交互体系
- 最好的程序设计语言是人类的自然语言



程序设计语言

■ 自然语言存在的问题：

- ✓ 存在表达歧义：陈老师这学期要上python课
- ✓ 文学色彩浓厚：春水初生，春林初盛，春风十里，不如你

因此，还无法借助自然语言进行程序设计



程序设计语言

程序设计语言种类

- 机器语言：01代码，CPU认识的语言
 - ✓ 例：2+3的运算 1101001000111011
- 汇编语言：在机器语言上增加了人类可读的助记符
 - ✓ 例：2+3的运算 add 2, 3, result
- 高级语言：向自然语言靠近的语言
 - ✓ 例：2+3的运算 result = 2 + 3



程序设计语言

程序设计语言种类

- 历史上出现过600多种程序设计语言
 - ✓ 这些语言的名字覆盖字母A到Z
- 常用的程序设计语言：100余种
 - ✓ C/C++/VB/Java/JavaScript/Ruby/Swift/Python（软件编程）
 - ✓ Verilog/VHDL（硬件描述）
 - ✓ PHP/HTML等（网络编程）



程序设计语言

执行方式：编译执行和解释执行

- 编译：将高级语言源代码一次性转换成目标代码（机器语言），程序便可以运行
- 解释：将高级语言源代码逐条转换成目标代码同时逐条执行，每次运行程序需要源代码和解释器，也叫源代码直接运行



程序设计语言

编译和解释各有优劣：

■ 编译的好处：

- ✓ 一次完性成，前期可进行多次优化，目标代码执行速度更快
- ✓ 目标代码在相同操作系统上使用灵活

■ 解释的好处：

- ✓ 保留源代码，便于维护源代码
- ✓ 良好的跨平台可移植性



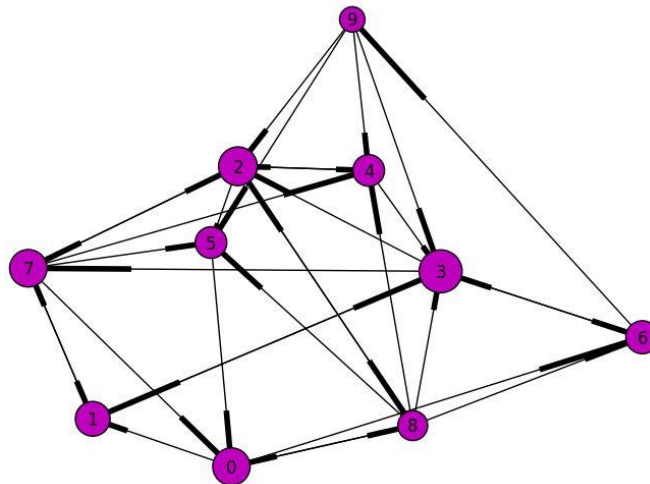
程序设计语言

■ 静态语言和脚本语言

- ✓ 静态语言：编译执行的编程语言，如C、Java等
- ✓ 脚本语言：解释执行的编程语言，如PHP等
- ✓ Python语言是脚本语言
- ✓ 对于计算机来说，都是根据计算指令，完成计算功能



Python语言介绍





Python语言介绍

- 人工智能：当红辣子鸡，Python语言被认为是跨入人工智能领域大门最好的一把钥匙
- IEEE 2017年编程语言排行榜首位
- 重视程度：
 - ✓ 北京、浙江信息技术教材编程语言将会从 VB 更换为 Python
 - ✓ 山东省小学信息技术六年级教材也加入了 Python，钢琴、奥数、绘画，Python
 - ✓ Python进入计算机二级，2018年9月首次开考



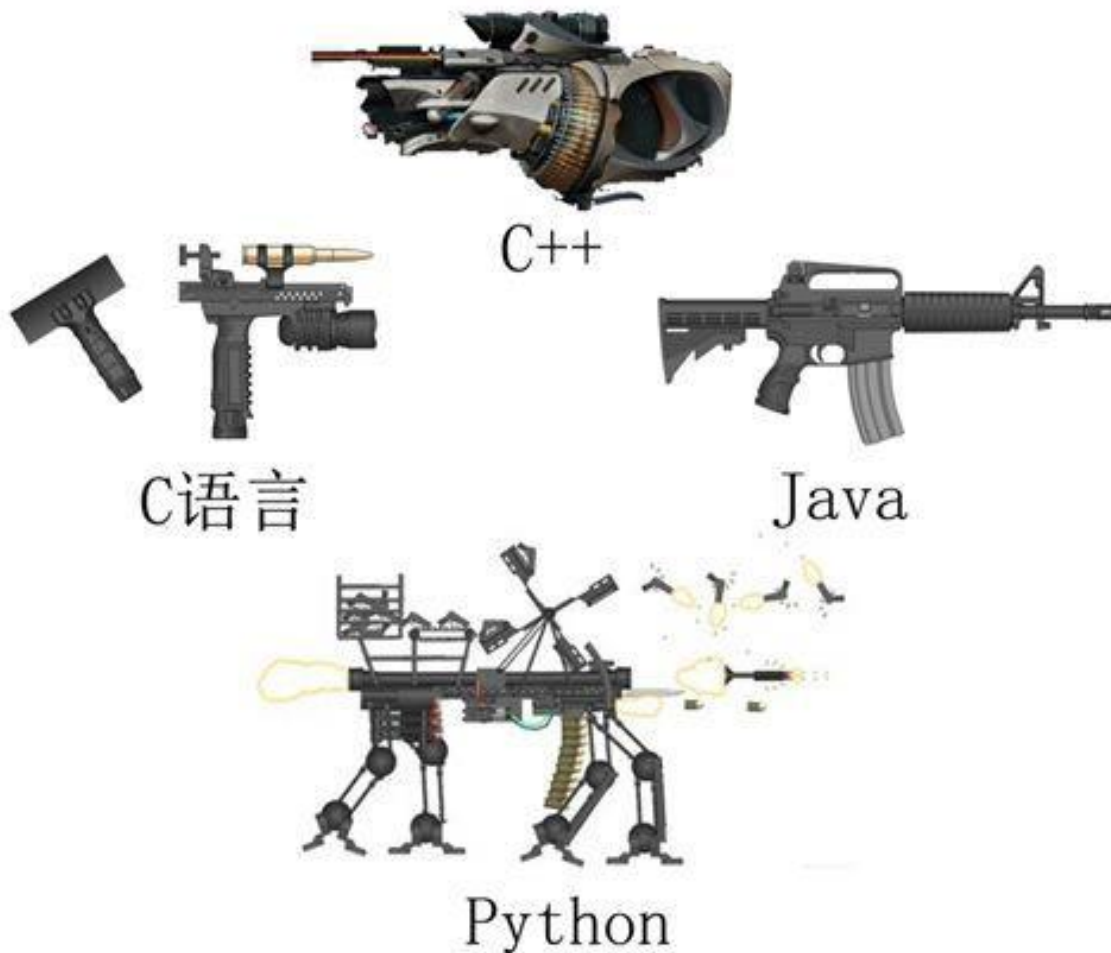
Python语言介绍





Python语言介绍

■ 程序员的好朋友：Python、Java、C、C++





Python语言介绍

- “最美丽的”编程语言：Python的魅力和影响力已经远超C、C++等编程语言前辈
- “胶水”语言：能够把用其他语言制作的各种模块（尤其是C/C++）很轻松地联结在一起
- 语法清楚、简洁强悍、简单易学、免费开源、丰富的库、开发效率高……（此处省略一万字）
- 全能：系统运维、图形处理、数学处理、文本处理、数据库编程、网络编程、web编程、多媒体应用、黑客编程、爬虫编写、机器学习、人工智能等



Python语言介绍

- Python语言是少有的一种可以称得上既简单又功能强大的编程语言。
- 你将惊喜地发现Python语言是多么地简单，它注重的是如何解决问题而不是编程语言的语法和结构



Python语言介绍

■ Python语言起源

- ✓ Python [``pai θ ən`], 译为“蟒蛇”
- ✓ 在1989年末, Guido van Rossum为了打发圣诞节的无聊, 决心开发一个新的脚本解释程序, 作为ABC的继承, 创造了Python
- ✓ ABC是由Guido参加设计的一种教学语言, 专门为非专业程序员设计
- ✓ 使用Python语言作为语言的名字, 英国幽默剧团“Monty Python”飞行马戏团的fans



Python语言介绍



- 创始人： Guido van Rossum, 荷兰人， 目前在google从事GAE/Python3.x方面研究



Python语言介绍

■ Python版本

- ✓ 1991年，公开发行
- ✓ 2002年，Python 2. x
- ✓ 2008年，Python 3. x
- ✓ 2018年，最新版本为3. 6



Python语言介绍

■ Python版本更迭

- ✓ 未来追求语言完美，更高级别的3.0系列不兼容早期2.0系列
- ✓ 2008年至今，版本更迭带来大量库函数的升级替换，Python语言的版本更迭痛苦且漫长
- ✓ Python 3.x系列已经成为主流



Python语言特色

■ 简单

✓ Python是一种代表简单主义思想的语言。阅读一个良好的Python程序就感觉像是在读英语一样，尽管这个英语的要求非常严格！Python的这种伪代码本质是它最大的优点之一。它使你能够专注于解决问题而不是去搞明白语言本身。

■ 易学

- ✓ Python极其容易上手
- ✓ Python有极其简单的语法



Python语言特色

■ 免费、开源

- ✓ Python是FLOSS（Free/Libre and Open Source Software，自由/开放源码软件）之一。简单地说，可以自由地发布这个软件的拷贝、阅读它的源代码、对它做改动、把它的一部分用于新的自由软件中。
- ✓ FLOSS是基于一个团体分享知识的概念。这是为什么Python如此优秀的原因之一——它是由一群希望看到一个更加优秀的Python的人创造并经常改进着的。



Python语言特色

■ 高层语言

- ✓ 当你用Python语言编写程序的时候，你无需考虑诸如如何管理程序使用的内存一类的底层细节



Python语言特色

■ 可移植性

- ✓ 由于它的开源本质，Python已经被移植在许多平台上（经过改动使它能够工作在不同平台上）。如果你小心地避免使用依赖于系统的特性，那么你的所有Python程序无需修改就可以在下述任何平台上面运行。
- ✓ 这些平台包括：Linux、Windows、FreeBSD、Macintosh、Solaris、OS/2、Amiga、AROS、AS/400、BeOS、OS/390、z/OS、Palm OS、QNX、VMS、Psion、Acom RISC OS、VxWorks等



Python语言特色

■ 解释性

- ✓ 用编译性语言比如C或C++写的程序可以从源文件（即C或C++语言）转换到一个计算机使用的语言（二进制代码，即0和1）。这个过程通过编译器和不同的标记、选项完成。当运行程序的时候，连接/转载器软件把程序从硬盘复制到内存中并且运行
- ✓ 而Python语言写的程序不需要编译成二进制代码。你可以直接从源代码运行程序。在计算机内部，Python解释器把源代码转换成称为字节码的中间形式，然后再把它翻译成计算机使用的机器语言并运行



Python语言特色

■ 面向对象

- ✓ 在面向过程的语言中，程序是由过程或仅仅是可重用代码的函数构建起来的
- ✓ 在面向对象的语言中，程序是由数据和功能组合而成的对象构建起来的
- ✓ 与其他主要的语言如C++和Java相比，Python以一种非常强大又简单的方式实现面向对象编程



Python语言特色

■ 可扩展性

✓ 如果你需要你的一段关键代码运行得更快或者希望某些算法不公开，你可以把你的部分程序用C或C++编写，然后在你的Python程序中使用它们

■ 可嵌入性

✓ 你可以把Python嵌入你的C/C++程序，从而向你的程序用户提供脚本功能



Python语言特色

■ 丰富的库

- ✓ Python标准库很庞大。它可以帮助你处理各种工作，包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器、CGI、FTP、电子邮件、XML、XML-RPC、HTML、WAV文件、密码系统、GUI（图形用户界面）、Tk和其他与系统有关的操作
- ✓ 除了标准库以外，还有许多其他高质量的库，如wxPython、Twisted和Python图像库等等



Python可以做什么？

■ 系统编程

- ✓ Python对操作系统服务的内置接口，使其成为编写可移植的维护操作系统的管理工具和部件的理想工具。Python程序可以搜索文件和目录树，可以运行其他程序，用进程或线程进行并行处理等
- ✓ Python的标准库绑定了POSIX以及其他常规操作系统工具：环境变量、文件、套接字、管道、进程、多线程、正则表达式、命令行参数、标准流接口、Shell命令启动器、文件名扩展等



Python可以做什么？

■ 用户图形接口

- ✓ Python的简洁以及快速的开发周期十分适合开发GUI程序
- ✓ 内置了Tk GUIAPI，可以生成可移植的本地观感的GUI，可以不做任何改变就可以运行在Windows、Xwindows、MacOS等平台
- ✓ PythonCard、Dabo等构建在wxPython 和 Tkinter 基础上的高级工具包
- ✓ 通过适当的库，可以使用其他GUI工具包



Python可以做什么？

■ 数据库编程

- ✓ 支持所有主流数据库：Oracle、Sybase、MySQL、PostgreSQL、Informix、SQLite
- ✓ 定义了标准的、可移植的数据库API
- ✓ 面向对象数据库系统：ZODB
- ✓ 从关系数据库映射到Python类（ORM）：
SQLAlchemy 、 SQLAlchemy



Python可以做什么？

■ 数值计算和科学计算

- ✓ NumPy

■ 游戏、图像、人工智能、机器人等

- ✓ Pygame/Bigworld

- ✓ PIL

- ✓ PyRO（机器人控制）

- ✓ 神经网络仿真器

- ✓ NLTK（自然语言分析）



与其他语言比较

- 比Tcl更强大
- 比Perl的语法和设计更简单
- 比Java更简单、更易于使用
- 比C++更简单、更易于使用
- 比Visual Basic更强大也具备跨平台特性
- 比Ruby更成熟、语法更具可读性



Python语言介绍

■ Python的缺点

- ✓ 运行速度不够快

■ 开发速度与运行速度之间的矛盾

- ✓ 至今还没有一门编程语言，开发速度比Python快，运行速度比C快



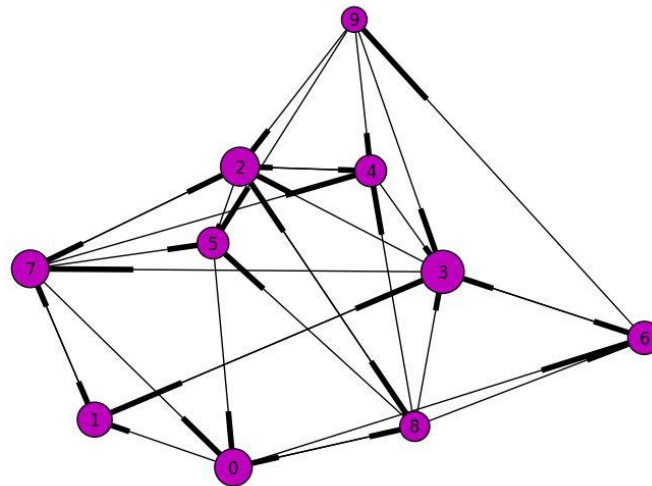
Python语言介绍

■ 谁在用Python?





Python开发环境配置





Python开发环境配置

■ 下载

- ✓ 到Python主页下载并安装Python基本开发和运行环境，网址：www.python.org/downloads/
- ✓ 根据操作系统不同选择不同版本
- ✓ 下载相应的Python 3.0系列版本程序



Python开发环境配置

■ 下载

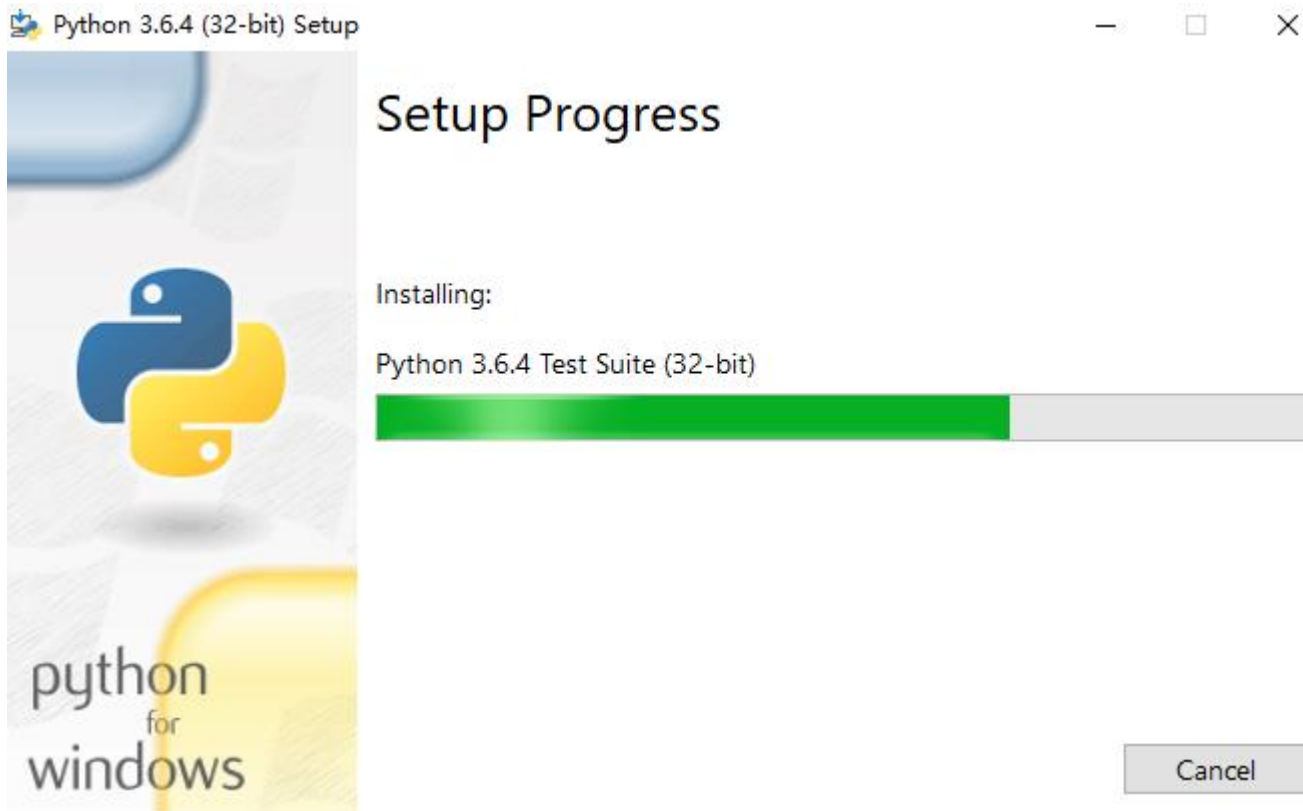
The screenshot shows the Python.org website. The top navigation bar includes the Python logo, a search bar, and a 'Socialize' button. The main navigation menu has links for 'About', 'Downloads', 'Documentation', 'Community', 'Success Stories', 'News', and 'Events'. The 'Downloads' menu is open, displaying a list of links: 'All releases', 'Source code', 'Windows', 'Mac OS X', 'Other Platforms', 'License', and 'Alternative Implementations'. A 'Download for Windows' section is highlighted, featuring buttons for 'Python 3.6.4' and 'Python 2.7.14'. Below these buttons, a note states: 'Note that Python 3.5+ cannot be used on Windows XP or earlier.' It also mentions that Python can be used on many operating systems and environments, and provides a link to 'View the full list of downloads.' On the left side of the 'Downloads' menu, a snippet of Python code is visible:

```
# Python 3: File
>>> def fib(n):
>>>     a, b =
>>>     while
>>>         pr
>>>         a,
>>>     print(
>>>     fib(1000)
0 1 1 2 3 5 8
```



Python开发环境配置

■ 安装

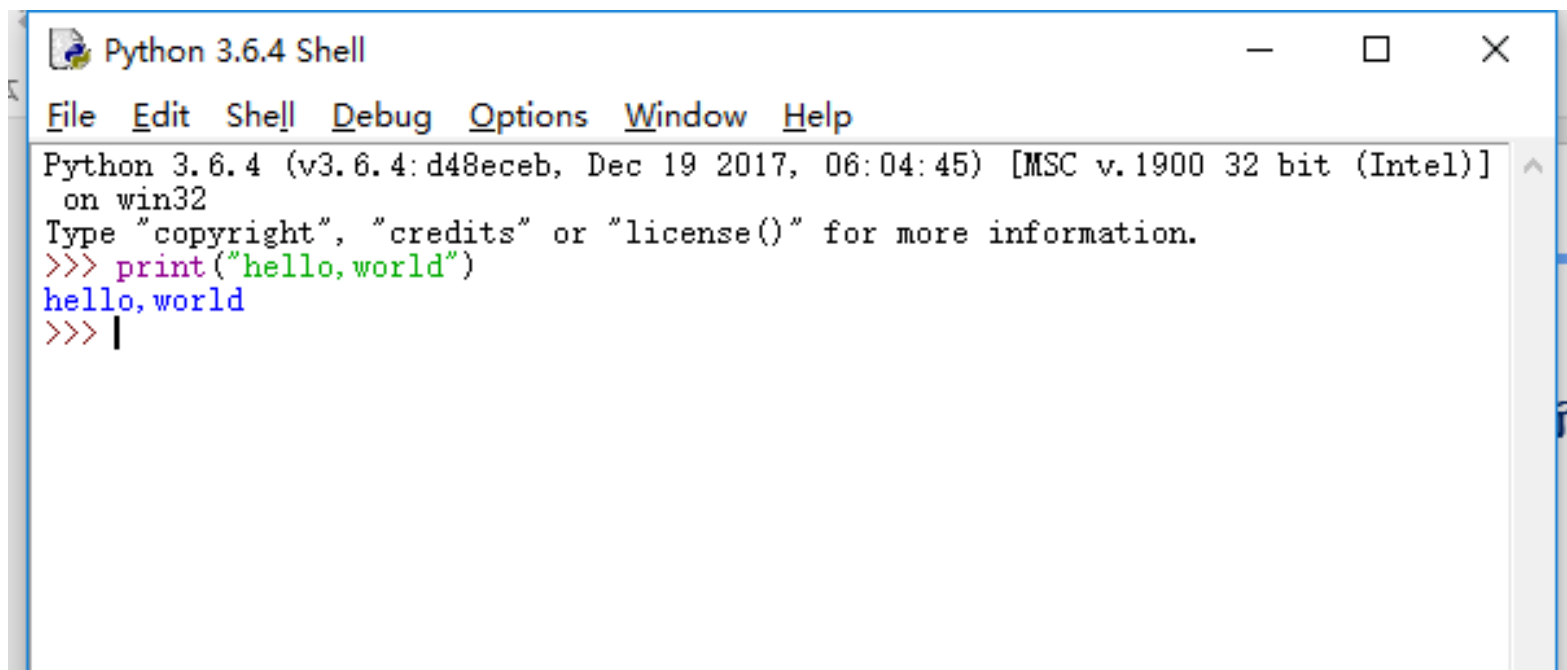




Python开发环境配置

■ 启动

- ✓ 方法1：调用IDLE来启动Python图形化运行环境



```
Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("hello,world")
hello,world
>>> |
```



Python开发环境配置

■ 启动

✓ 方法2：打开IDLE，点击Ctrl+N打开一个新窗口，输入语句并保存，使用快捷键F5即可运行该程序

The screenshot displays two windows from the Python 3.6.4 development environment. The top window, titled 'Python 3.6.4 Shell', shows the interactive prompt with the following text: 'Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32. Type "copyright", "credits" or "license()" for more information.' It shows three successful executions of the command `>>> print("hello,world")`, each resulting in the output 'hello,world'. The bottom window, titled 'hello.py - C:/Users/chenfox/AppData/Local/Programs/Python/Python36...', shows a text editor with the code `print("hello,world")` and a cursor at the end of the line.

```
Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("hello,world")
hello,world
>>>
== RESTART: C:/Users/chenfox/AppData/Local/Programs/Python/Python36-32/1.py ==
hello,world
>>>
== RESTART: C:/Users/chenfox/AppData/Local/Programs/Python/Python36-32/1.py ==
hello,world
>>>
== RESTART: C:/Users/chenfox/AppData/Local/Programs/Python/Python36-32/hello.py
hello,world
>>>

hello.py - C:/Users/chenfox/AppData/Local/Programs/Python/Python36...
File Edit Format Run Options Window Help
print("hello,world")
```



Python开发环境配置

■ 启动

- ✓ 当程序比较大的时候，可以将程序划分成多个模块编写，每个模块用一个文件保存
- ✓ 模块之间可以通过导入互相调用（import），模块也可以导入库中的其他模块
- ✓ Python是以模块进行重用的，模块中可以包括类、函数、变量等



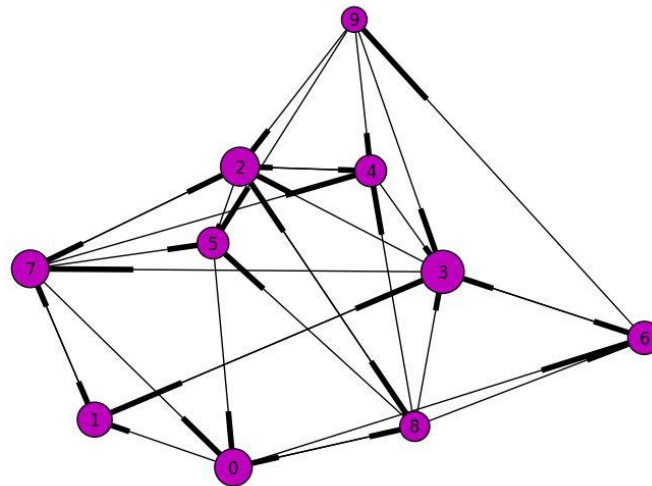
Python开发环境配置

■ 启动

- ✓ 方法3：将Python集成到Eclipse、PyCharm等面向较大规模项目开发的集成开发环境中



Python中类型的概念





Python中类型的概念

- 类型的作用
- 数据从不同角度看有不同的含义
- 这样一个数据：10, 011, 101，该怎样解释呢？
 - ✓ 1个二进制数字或者1个十进制数字
 - ✓ 一段文本或者用, 分割的3个数字
- 程序设计语言不允许存在语法歧义，因此，需要明确说明数据的含义，这就是“类型”的作用

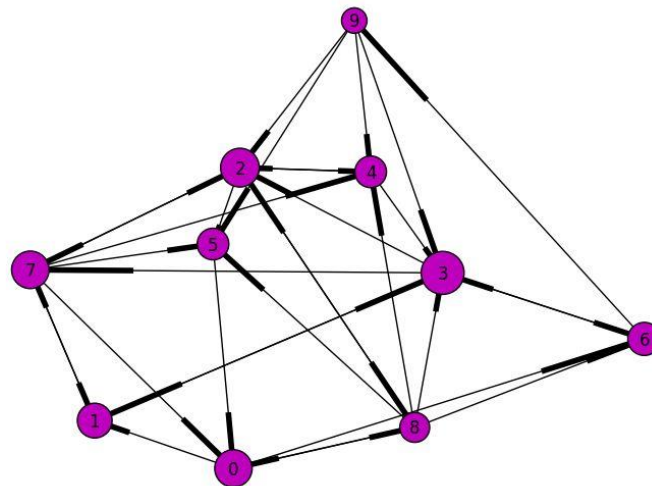


Python中类型的概念

- 类型是编程语言对数据的一种划分
- 主要介绍6种Python语言中的类型：
 - ✓ 数字类型、字符串类型
 - ✓ 元组类型、列表类型
 - ✓ 文件类型、字典类型



数字类型





数字类型

■ Python语言包括三种数字类型

- ✓ 整数类型 (int) : 与数学中整数的概念一致
- ✓ 浮点数类型 (float) : 与数学中实数的概念一致
- ✓ 复数类型 (complex)



数字类型

■ 整数类型

- ✓ 与数学中的整数概念一致，理论上没有取值范围限制，实际受限于运行Python的计算机内存

■ `pow(x, y)` 函数：计算 x^y

- ✓ 打开IDLE
- ✓ 程序1： `pow(2, 10)` ， `pow(2, 15)`
- ✓ 程序2： `pow(2, 1000)`
- ✓ 程序3： `pow(2, pow(2, 15))`



数字类型

■ 整数类型的4种进制表示

- ✓ 1010, 99, -217 (默认情况)
- ✓ 0x9a, -0X89 (0x, 0X开头表示16进制数)
- ✓ 0b010, -0B101 (0b, 0B开头表示2进制数)
- ✓ 0o123, -00456 (0o, 00开头表示8进制数)



数字类型

■ 浮点数类型

- ✓ 带有小数点及小数的数字
- ✓ 必须带有小数，小数可以是0

- Python语言中浮点数的数值范围存在限制，小数精度也存在限制。这种限制与在不同计算机系统有关



数字类型

■ 浮点数范围

✓ >>> import sys

✓ >>> sys.float_info

```
Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> import sys
>>> sys.float_info
sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308, min=2.
2250738585072014e-308, min_exp=-1021, min_10_exp=-307, dig=15, mant_dig=53, epsi
lon=2.220446049250313e-16, radix=2, rounds=1)
>>>
```



数字类型

■ 示例

- ✓ 0.0, -77., -2.17
- ✓ 96e4, 4.3e-3, 9.6E5 （科学计数法）
- ✓ 科学计数法使用字母“e”或者“E”作为幂的符号，以10为基数



数字类型

■ 复数类型

- ✓ 与数学中的复数概念一致, $z = a + bj$,
a是实数部分, b是虚数部分, a和b都是浮点类型, 虚数部分用j或者J标识
- ✓ $12.3+4j$, $-5.6+7j$
- ✓ $z = 1.23e-4+5.6e+89j$
- ✓ 对于复数z, 可以用`z.real` 获得实数部分,
`z.imag`获得虚数部分
- ✓ $a=1+j?$



数字类型

■ 数字类型的关系

✓ 三种类型存在一种逐渐“扩展”的关系：

整数 → 浮点数 → 复数

（整数是浮点数特例，浮点数是复数特例）

✓ 不同数字类型之间可以进行混合运算，运算后生成结果为最宽类型

$123 + 4.0 = 127.0$

（整数 + 浮点数 = 浮点数）



数字类型

■ 数字类型的转换

✓ 数值运算操作符可以隐式地转换输出结果的数字类型

■ 函数：int(), float(), complex()

✓ `int(4.5) = 4` （直接去掉小数部分）

✓ `float(4) = 4.0` （增加小数部分）

✓ `complex(4) = 4 + 0j`

✓ 复数不能直接转换为其他类型



数字类型

数字类型的判断

- 函数：`type(x)`，返回`x`的类型，适用于所有类型的判断
 - ✓ `type(4.5)`
 - ✓ `type(z)`



数字类型

■ 内置的数值运算操作符

操作符	描述
$x + y$	x与y之和
$x - y$	x与y之差
$x * y$	x与y之积
x / y	x与y之商
$x // y$	x与y之整数商，即：不大于x与y之商的最大整数
$x \% y$	x与y之商的余数，也称为模运算
$-x$	x的负值，即： $x * (-1)$
$+x$	x本身
$x ** y$	x的y次幂，即： x^y



数字类型

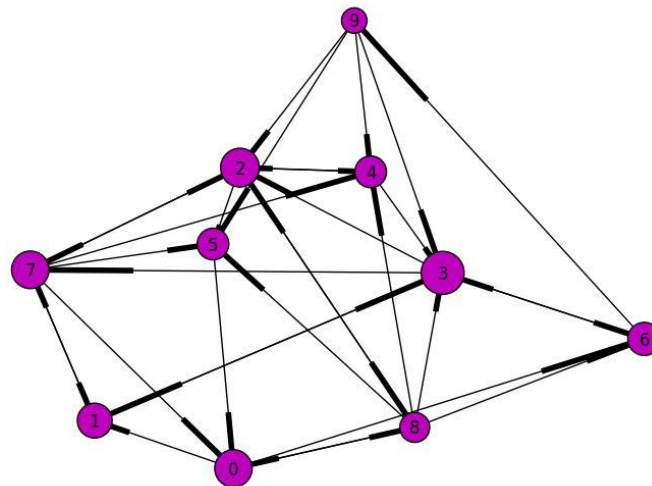
■ 内置的数值运算函数

- ✓ Python解释器提供了一些内置函数，在这些内置函数之中，有6个函数与数值运算相关

函数	描述
<code>abs(x)</code>	x的绝对值
<code>divmod(x, y)</code>	$(x//y, x\%y)$ ，输出为二元组形式（也称为元组类型）
<code>pow(x, y[, z])</code>	$(x**y)\%z$ ， <code>[..]</code> 表示该参数可以省略，即： <code>pow(x,y)</code> ，它与 <code>x**y</code> 相同
<code>round(x[, ndigits])</code>	对x四舍五入，保留ndigits位小数。 <code>round(x)</code> 返回四舍五入的整数值
<code>max(x₁, x₂, ..., x_n)</code>	x ₁ , x ₂ , ..., x _n 的最大值，n没有限定
<code>min(x₁, x₂, ..., x_n)</code>	x ₁ , x ₂ , ..., x _n 的最小值，n没有限定



字符串类型





字符串类型

- 字符串 (str) 是用双引号""或者单引号''括起来的一个或多个字符
- 字符串可以保存在变量中，也可以单独存在
- 可以用type()函数测试一个字符串的类型
 - ✓ type("1")
 - ✓ type(1)



字符串类型

- 字符串是一个字符序列：字符串最左端位置标记为0，依次增加。字符串中的编号叫做“索引”

H	e	l	l	o		J	o	h	n
0	1	2	3	4	5	6	7	8	9



字符串类型

- 单个索引辅助访问字符串中的特定位置
- 格式为<string>[<索引>]
 - ✓ `str1="hello, world"`
 - ✓ `print(str1[1])`
- Python中字符串索引从0开始，一个长度为L的字符串最后一个字符的位置是L-1
- Python同时允许使用负数从字符串右边末尾向左边进行反向索引，最右侧索引值是-1



字符串类型

- 可以通过两个索引值确定一个位置范围，返回这个范围的子串，字符串的切片
 - ✓ 格式：<string>[<start>:<end>]
- start和end都是整数型数值，这个子序列从索引start开始直到索引end结束，但不包括end位置



字符串类型

- 字符串之间可以通过+或*进行连接
 - ✓ 加法操作(+)将两个字符串连接成为一个新的字符串
 - ✓ 乘法操作(*)生成一个由其本身字符串重复连接而成的字符串
 - ✓ `x in s`: 如果x是s的子串, 返回True, 否则返回False
 - ✓ `str[N:M]`: 切片, 返回子串



字符串类型

- `len()` 函数能返回一个字符串的长度
 - ✓ `str1="hello, world"`
 - ✓ `len(str1[1])`



字符串类型

- 大多数数据类型都可以通过`str()`函数转换为字符串
 - ✓ `str(123)`
- `type()`函数测试一个字符串的类型



字符串操作

■ 字符串处理方法

操作	含义
+	连接
*	重复
<string>[]	索引
<string>[:]	剪切
len(<string>)	长度
<string>.upper()	字符串中字母大写
<string>.lower()	字符串中字母小写
<string>.strip()	去两边空格及去指定字符
<string>.split()	按指定字符分割字符串为数组
<string>.join()	连接两个字符串序列
<string>.find()	搜索指定字符串
<string>.replace()	字符串替换
for <var> in <string>	字符串迭代



字符串操作

■ 字符串的操作

✓ 可以通过for和in组成的循环来遍历字符串中每个字符

✓ `for<var>in<string>:`

操作

例: `str1="hello, world"`

`for p in str1:`

`print(p)`



字符串类型

- Python语言转义符： \，在字符串中表示转义，即该字符与后面相邻的一个字符共同组成了新的含义
- 输出带有引号的字符串，可以使用转义符
- 使用\\ 输出带有转义符的字符串
 - ✓ `print("\\"你好\\")`
 - ✓ `print("\\\\")`

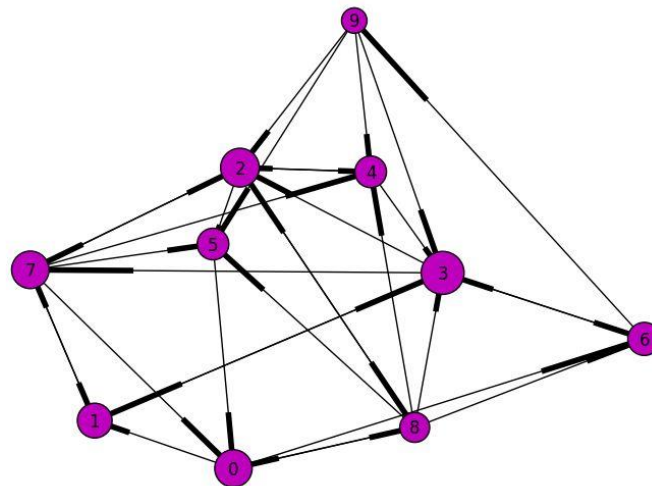


字符串操作

- 用转义符可以在字符串中表达一些不可直接打印的信息，例如：
 - ✓ 用 `\n` 表示换行
 - ✓ `\t` 表示制表符
 - ✓ `print("Hello\nWorld\n\nGoodbye 32\n")`



列表类型





列表

■ 基本概念：list

- ✓ 列表是有序的元素集合，所有元素放在一对中括号中，每个元素用逗号隔开，没有长度限制；
- ✓ 当列表元素增加或删除时，列表对象自动进行扩展或收缩内存，保证元素之间没有缝隙；

■ 列表元素可以通过索引访问单个元素

- ✓ `a=[0, 1, 2, 3, 4, 5, 6, 7, 8]`
- ✓ `a[0]`
- ✓ `a[3:]`



列表

■ 列表

- ✓ 列表中每个元素类型可以不一样
- ✓ 访问列表中元素时采用索引形式

■ 列表元素修改

- ✓ 列表大小没有限制，可以随时修改
- ✓ `a[0]=9`



列表的操作

- 针对列表有一些基本操作，这些操作与字符串操作类似

列表操作符	操作符含义
<code>< list1 > + < list2 ></code>	连接两个列表
<code>< list > * < 整数类型 ></code>	对列表进行整数次重复
<code>< list > [< 整数类型 >]</code>	索引列表中的元素
<code>len(< seq >)</code>	列表中元素个数
<code>< list >[< 整数类型 > : < 整数类型 >]</code>	取列表的一个子序列
<code>for < var > in < list > :</code>	对列表进行循环列举
<code>< expr > in < list ></code>	成员检查，判断<expr>是否在列表中



列表的操作

■ 列表相关方法

方法	方法含义
<code>< list > . append (x)</code>	将元素x增加到列表的最后
<code>< list > . sort ()</code>	将列表元素排序
<code>< list > . reverse ()</code>	将序列元素反转
<code>< list > . index ()</code>	返回第一次出现元素x的索引值
<code>< list > . insert (i, x)</code>	在位置i处插入新元素x
<code>< list > . count (x)</code>	返回元素x在列表中的数量
<code>< list > . remove (x)</code>	删除列表中第一次出现的元素x
<code>< list > . pop (i)</code>	取出列表中位置i的元素，并删除它



列表的操作

- 对于字符串，可以通过`split()`函数，将字符串拆分成一个列表，默认以空格分割，也可通过指定分隔符对字符串进行切片，并返回分割后的字符串列表（`list`）



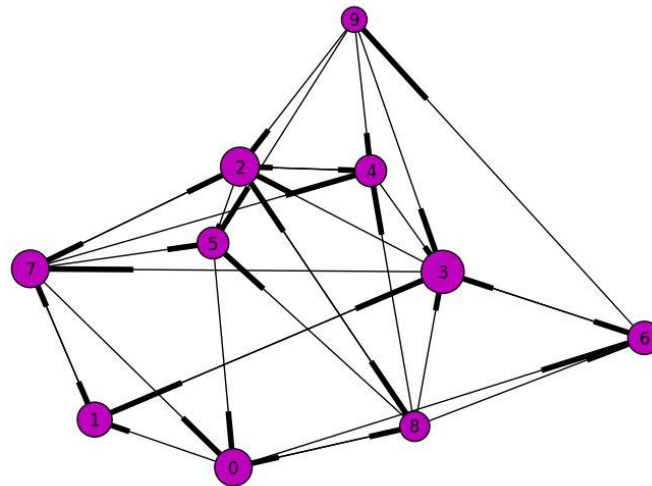
列表的操作

■ `u = "www.doiido.com.cn"`

```
1 >>> u = "www.doiido.com.cn"
2
3 #使用默认分隔符
4 >>> print u.split()
5 ['www.doiido.com.cn']
6
7 #以"."为分隔符
8 >>> print u.split('.')
9 ['www', 'doiido', 'com', 'cn']
10
11 #分割0次
12 >>> print u.split('.',0)
13 ['www.doiido.com.cn']
14
15 #分割一次
16 >>> print u.split('.',1)
17 ['www', 'doiido.com.cn']
18
19 #分割两次
20 >>> print u.split('.',2)
21 ['www', 'doiido', 'com.cn']
22
23 #分割两次，并取序列为1的项
24 >>> print u.split('.',2)[1]
25 doiido
```



元组类型





元组的概念

- 元组 (tuple) 是包含多个元素的类型，元素之间用逗号分割
 - ✓ 例如：t1 = 123, 456, "hello"
- 元组可以是空的，t2=()
- 元组包含一个元素时：t3=123,
- 元组外侧可以使用括号，也可以不使用



元组

- 元组中元素可以是不同类型
- 一个元组也可以作为另一个元组的元素，此时，作为元素的元组需要增加括号，从而避免歧义

✓ 例如：

```
t3 = 123, 456, ("hello", "world")
```



元组

- 元组中各元素存在先后关系，可以通过索引访问元组中元素。
 - ✓ 例如：t3[0]
- 元组定义后不能更改，也不能删除
 - ✓ t3[0]=789



元组

- 与字符串类型类似，可以通过索引区来访问元组中的部分元素。
 - ✓ 例如：t3[1:]
- 与字符串一样，元组之间可以用+号和*号进行运算



元组总结

- Python语言中的元组类型定义后不能修改
- 不可变的tuple有什么意义呢？
 - ✓ 因为tuple不可变，所以代码更安全。
- 如果仅考虑代码的灵活性，也可以用列表类型代替元组类型。



谢谢