



# 第0章 基础知识

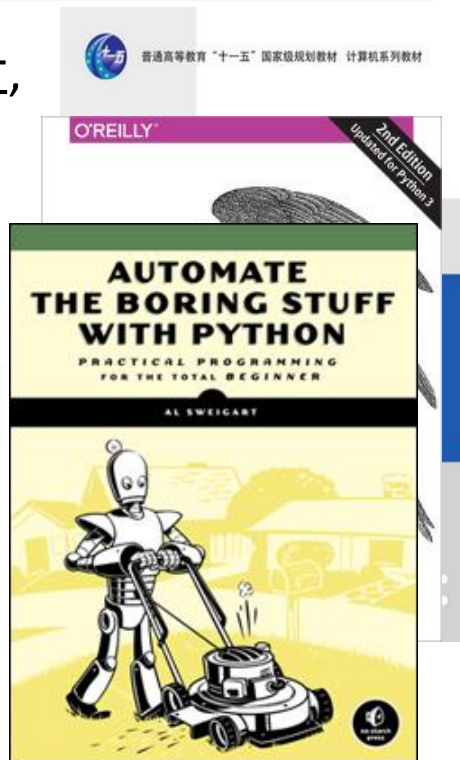
---

刘 卉

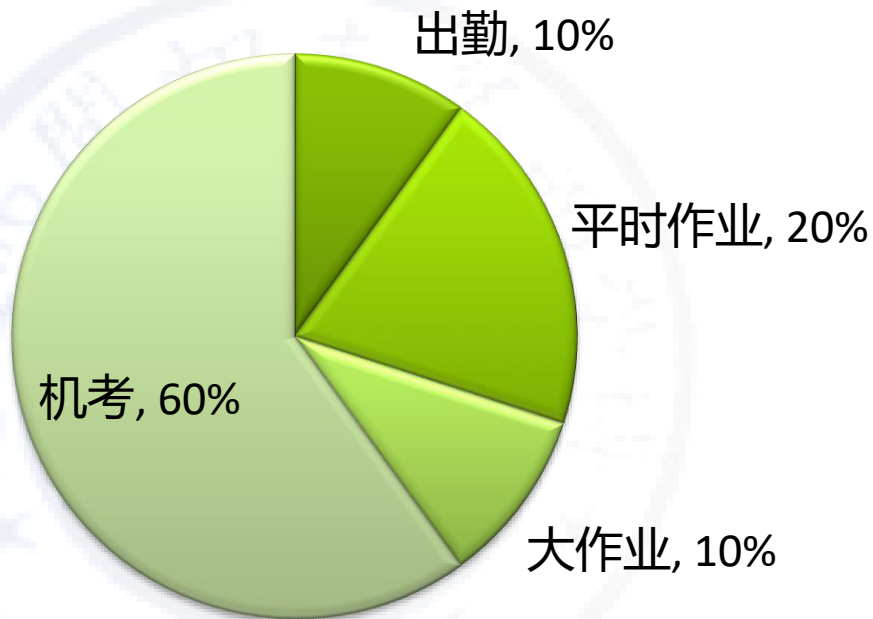
huiliu@fudan.edu.cn

# 教材及参考书

- 董付国. 《Python程序设计基础》. 清华大学出版社, 2015年8月, ISBN: 978-7-302-41058-4
- Think Python 2nd Edition by Allen B. Downey.  
<https://greenteapress.com/wp/think-python-2e/>
- <https://automatetheboringstuff.com/>



# 成绩评定规则



# 考勤和作业

## □ 考勤

- 进教室前刷一卡通 or 在教室的计算机上用学号登录.

## □ 每周作业

- 三道编程题: 每周一3:10pm发布在[Canvas](#)上.
- 提交截止期: 每周三10:00pm.
- 迟交作业: 1) 下一次上课前, 通过邮件提交给助教, 抄送给老师; 2) 评分降一级

## □ 大作业

- 大约第10周布置, 期末考试前提交.

# 授课方式

## □ 2节上课

- 课程内容推进速度很快

## □ 2节上机

- 动手实践新课内容, 及时消化吸收
- 编程完成每周作业
- "老师+助教"现场辅导

## □ 助学手段——Canvas

- 布置/提交作业, 上传/下载课件

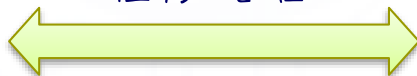
# 计算机编程概述

## 程序(设计)语言是什么？

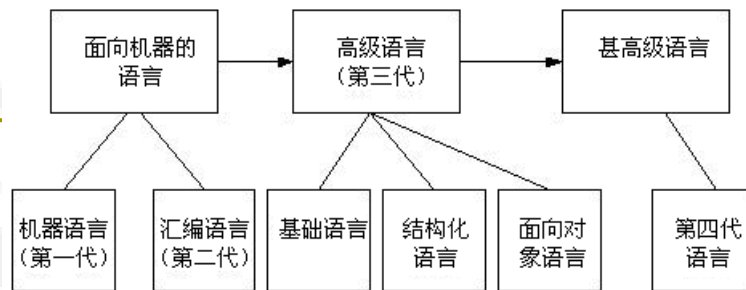
- 程序: 用于控制计算机的一系列指令
- 程序语言: 描述指令的语言  $\Rightarrow$  "程序员"与"机器"对话.
  - 语法(syntax): 哪些符号/文字的组合方式是正确的.
  - 语义(semantics): 程序的意义, 程序运行时计算机做什么.



程序语言



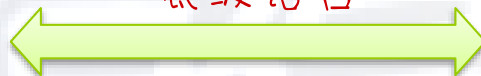
# 程序(设计)语言的发展



前辈



低级语言



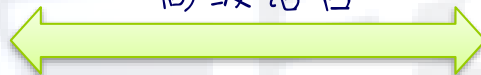
0101110, 汇编



高级码农



高级语言



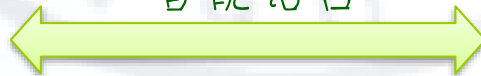
C, C++, Java, Python



任何人



智能语言



领域语言、自然语言



机器人:  
Atlas

# 程序(设计)语言的分类——解释型与编译型

- 解释器(Interpreter): 将语句翻译成机器码⇒执行
  - 修改程序方便⇒修改代码重新运行即可
  - 每次运行, 都要进行翻译, 影响运行速度
  - 必须有解释器才能运行, 跨平台
  - Python, Javascript, Perl, PhP, .....



# 编译器(Compiler): 将语句翻译成机器码, 形成目标代码文件

- 相比解释, 编译时可作更多优化;
- 修改程序后需重新编译;
- 一次编译, 执行过程中不再需要翻译语句;
- 编译后的目标代码可直接在相应的操作系统中运行, 不再需要编译器;
- C, C++, .....

## 有些语言将解释和编译结合在一起

- Java: 源代码首先通过编译器(javac)转换为Java字节码(Byte Code), 然后在目标机器上通过解释器来运行.
- Python也支持伪编译⇒将程序转换为字节码来优化程序和提高执行速度.

## 关于Python的补充说明

- 使用py2exe工具将Python程序转换为“.exe”可执行程序, 可在未安装Python解释器和相关依赖包的平台上运行.

一种解释型高级动态编程语言

1989年, Guido van Rossum设计了Python.

- Python的前身是ABC语言: 由Guido参加设计的一种教学语言, 专门为非专业程序员而设计.
- 1989年圣诞节期间, 在Amsterdam, Guido为了打发圣诞节的无聊, 决心开发一个新的脚本解释程序, 做为ABC语言的一种继承



- Python以接近自然语言的风格诠释程序设计，逐渐成为最受欢迎的编程语言之一。
- 广泛应用在系统管理、科学计算、大数据、Web应用、图形用户界面开发、游戏、.....

e.g. 引力波数据分析(gwpy)采用Python实现:

<https://github.com/gwpy/gwpy>

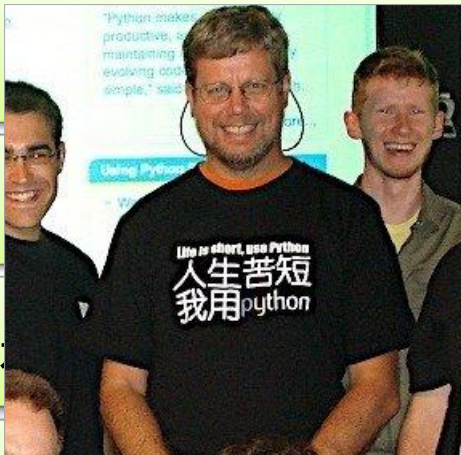
"学了Python才知道编程不止有快感, 更有美感"

Google、Instagram、豆瓣、

主要的互联网公司都在用它

"人生苦短, 快学Python"

小白迅速入门编程, Python



Python是世界上最容易学的编程语言, 从没接触过编程的人也能搞定.

网络广告摘录, 请谨慎相信☺













## □ Python语言的特点\*

- 简单、易学
- 开源、免费
- 跨平台
- 灵活
- 可嵌入
- 丰富的扩展库支持

\* 部分内容摘自百度百科

## □ 与其它语言比较\*

简单 + 强大

	C++		JavaScript
	Java/C#		PHP(Without MySQL)
	Ruby		Pascal
	Perl		Lisp
	Visual Basic		Haskell
	Python		C

\* <http://www.dataguru.cn/thread-190925-1-1.html>

# Python之禅(The Zen of Python)

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!

—— Tim Peters

- Python的设计哲学:



- Python开发者的哲学:

“用一种方法，最好是只有一种方法来做一件事”。

在设计Python语言时，如果面临多种选择，Python开发者总会拒绝花哨的语法，而选择明确的没有或者很少有歧义的语法。





# 第1章 Python基础知识

---

刘 卉

huiliu@fudan.edu.cn



# 主要内容

## 1.0 预备知识

## 1.1 Python版本

## 1.2 Python的安装和使用

## 1.4 Python基础知识

### 1.4.1 Python对象模型

### 1.4.2 Python变量

### 1.4.3 基本输入输出

### 1.4.4 数字

### 1.4.5 字符串

### 1.4.6 运算符和表达式

### 1.4.7 常用内置函数

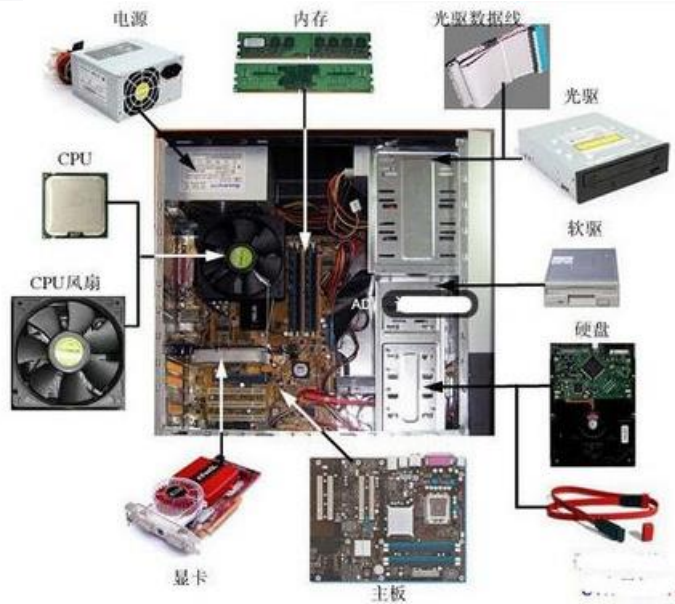
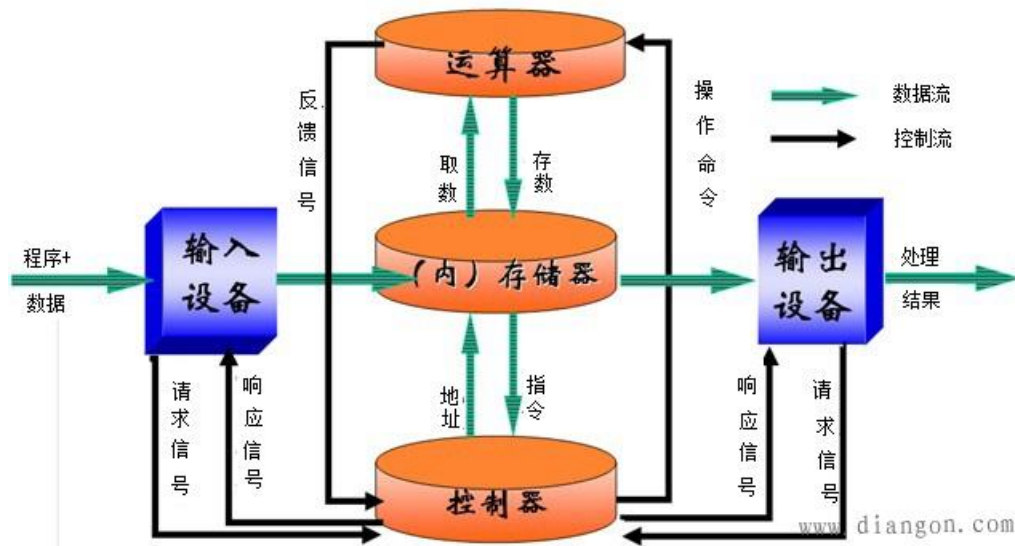
### 1.4.8 对象的删除

### 1.4.9 模块导入与使用(含1.3)

## 1.5 Python代码编写规范

# 1.0 预备知识

## 现代计算机的部件



# 数据表达——Bit, Byte(K/M/G)

- 计算机内部采用二进制表示

- 一个字节=8个比特 (1B=8b)
- 一个字节可以表示0到255的整数
  - 或者也可表示一个ASCII字符
- 二进制00101011对应十进制43

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

$$0 * 2^7 + 0 * 2^6 + 1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 = 43$$

- Kilobyte (kB):  $2^{10} = 1024$  bytes
- Megabyte (MB):  $2^{20} = 1024$  kB or  $1024 * 1024$  bytes
- Gigabyte (GB):  $2^{30} = 1024$  MB
- Terabyte (TB):  $2^{40} = 1024$  GB



# 字符在计算机内的保存方式

## ASCII编码

- 1个字节↔ 1个字符
- 对128个常用字符进行了编码: a-z、A-Z、数字0-9、标点符号、非打印字符(换行符、制表符等4个)以及控制字符(退格、响铃等).

# ASCII表

( American Standard Code for Information Interchange 美国标准信息交换代码 )

高四位	ASCII控制字符											ASCII打印字符													
	0000					0001					0010	0011		0100	0101		0110		0111						
	0					1					2	3		4	5		6		7						
	十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	Ctrl		
0000	0	0		^@	NUL	\0	空字符	16	▶	^P	DLE	数据链路转义	32		48	0	64	@	80	P	96	`	112	p	
0001	1	1	☺	^A	SOH		标题开始	17	◀	^Q	DC1	设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q	
0010	2	2	☹	^B	STX		正文开始	18	↕	^R	DC2	设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r	
0011	3	3	♥	^C	ETX		正文结束	19	!!	^S	DC3	设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s	
0100	4	4	♦	^D	EOT		传输结束	20	¶	^T	DC4	设备控制 4	36	S	52	4	68	D	84	T	100	d	116	t	
0101	5	5	♣	^E	ENQ		查询	21	§	^U	NAK	否定应答	37	%	53	5	69	E	85	U	101	e	117	u	
0110	6	6	♠	^F	ACK		肯定应答	22	—	^V	SYN	同步空闲	38	&	54	6	70	F	86	V	102	f	118	v	
0111	7	7	•	^G	BEL	\a	响铃	23	↕	^W	ETB	传输块结束	39	'	55	7	71	G	87	W	103	g	119	w	
1000	8	8	▣	^H	BS	\b	退格	24	↑	^X	CAN	取消	40	(	56	8	72	H	88	X	104	h	120	x	
1001	9	9	○	^I	HT	\t	横向制表	25	↓	^Y	EM	介质结束	41	)	57	9	73	I	89	Y	105	i	121	y	
1010	A	10	◉	^J	LF	\n	换行	26	→	^Z	SUB	替代	42	*	58	:	74	J	90	Z	106	j	122	z	
1011	B	11	♂	^K	VT	\v	纵向制表	27	←	^[	BSC	\e	溢出	43	+	59	;	75	K	91	[	107	k	123	{
1100	C	12	♀	^L	FF	\f	换页	28	└	^_	FS	文件分隔符	44	,	60	<	76	L	92	\	108	l	124		
1101	D	13	♪	^M	CR	\r	回车	29	↔	^]	GS	组分分隔符	45	-	61	=	77	M	93	]	109	m	125	}	
1110	E	14	🎵	^N	SO		移出	30	▲	^^	RS	记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111	F	15	🎵	^O	SI		移入	31	▼	^-	US	单元分隔符	47	/	63	?	79	O	95	_	111	o	127	△	

^Backspace  
代码: DEL

注：表中的ASCII字符可以用“Alt + 小键盘上的数字键”方法输入。

制作：NHL QQ:1208980380

2013/08/08



# 基本编程过程——四部曲

一想  
(Thinking)

三码  
(Coding)

二画  
(Designing)

四测  
(Testing)



# 1.1 Python版本 <https://www.python.org/>

## □ Python2

- 2000.10发布
- 存在大量使用Python2开发的扩展库

## □ Python3

- 2008.12发布
- V3.x与V2.x不完全兼容

e.g. Python3: `print('abc')`

Python2: `print 'abc'`



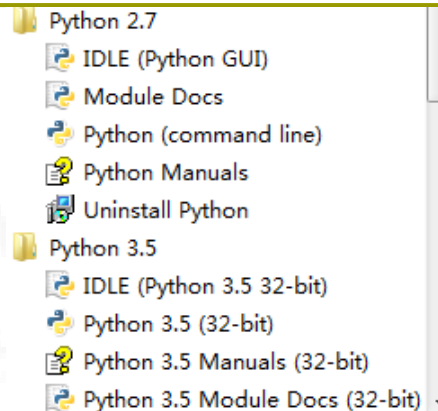
- 绝大部分扩展库同时支持Python2和Python3, 但是仍有少量扩展库仅支持Python2
- Python提供2to3的转换工具, 将支持Python2.7的代码转换为Python3
- 本课程采用Python 3——Python的现在和未来.
- 可同时安装多个版本, 通过更改系统环境变量path决定缺省使用哪个版本.
- 查看已安装版本的方法(在启动后的IDLE界面可直接看到)

```
>>> import sys
>>> sys.version
>>> sys.version_info
```

# 1.2 Python的安装

## □ Python解释器

- <https://www.python.org/downloads/> 根据所使用的操作系统(Linux/Mac OS X/Windows)下载相应的安装程序.
- Install Now: 按缺省设置进行安装.
- Customize installation: 可修改安装目录等设置.
- Python Launcher for windows(py): 帮助选择合适的Python版本.



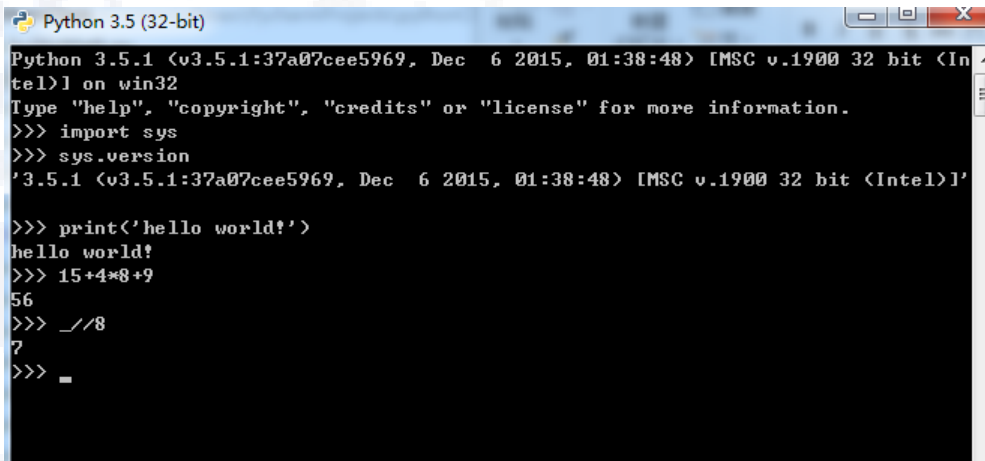
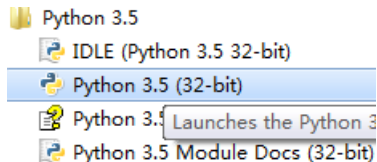
# Python的使用: Python控制台

## □ 通过菜单选择Python控制台(Console)

- Python解释器的提示符: >>>
- 可输入语句, 解释执行并输出结果
- 可充当计算器

- '\_' : 表示上次运算结果

(注: 仅在交互运行环境中起作用)

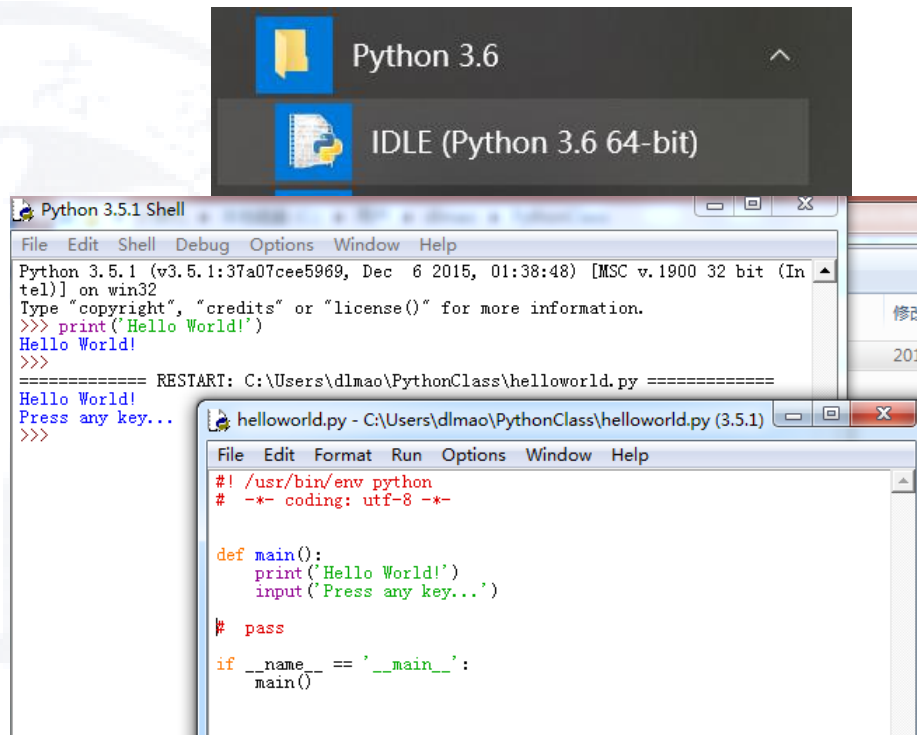


```
Python 3.5 (32-bit)
Python 3.5.1 <v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48> [MSC v.1900 32 bit <Intel>] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.version
'3.5.1 <v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48> [MSC v.1900 32 bit <Intel>]
>>> print('hello world!')
hello world!
>>> 15+4*8+9
56
>>> _/8
7
>>> _
```

# Python的使用: 集成开发环境IDE (Integrated Development Environment)



- ❑ 将编辑器, 调试器, 解释器集成在一起.
- ❑ 下载安装的Python环境中包括了IDLE.
- ❑ 其它常用的IDE: PyCharm, WingIDE, Sublime.



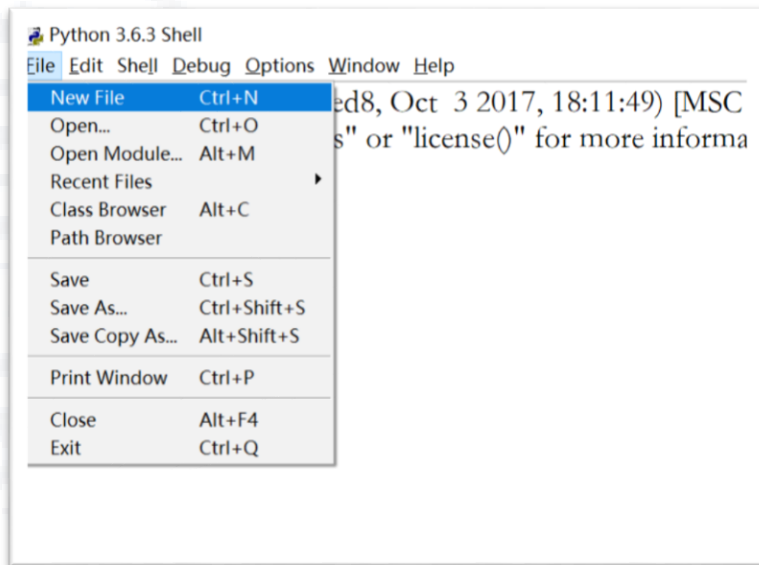
# IDLE

## □ Python Shell: 交互式解释器

- 实时运行Python语句.
- >>>: 交互式提示符
- 关闭Shell时, 输入的一切丢失.


## □ 编写自己和他人都能运行的程序

- 专门用来编程的文本编辑器



# "Hello"程序

## □ 如何创建→编辑→运行一个python程序

- 
1. File⇒New File 创建程序(.py文件)
  2. File ⇒Save 保存程序
    - 确保将程序保存在一个以后能找到的硬盘位置.
    - 为程序起个有意义的名字.
  3. Run⇒Check Module 检查语法错误
  4. Run⇒Run Module 运行程序

# IDLE快捷键

- ❑ 撤销(Ctrl+Z), 全选(Ctrl+A), 复制(Ctrl+C), 粘贴(Ctrl+V), 剪切(Ctrl+X)

快捷键	功能说明
Alt+p	浏览历史命令(上一条)
Alt+n	浏览历史命令(下一条)
Ctrl+F6	重启Shell, 之前定义的对象和导入的模块全部失效
F1	打开Python帮助文档
Alt+/	自动补全曾经出现过的单词, 如果多个单词具有相同前缀, 则在多个单词中循环选择
Ctrl+]	缩进代码块
Ctrl+[	取消代码块缩进
Alt+3	注释代码块
Alt+4	取消代码块注释



# 1.4 Python基础知识

## 1.4.1 对象(Object): Python中各种数据的抽象

- 标识id(identity): 对象一旦创建其id不变, 可看成该对象在内存中的地址.

形象比喻: 内存 - 大楼, 地址(id) - 房间号

e.g. 内置函数id(x) 返回对象x的id

a is b 判别a和b是否为同一个对象(id是否相同)



## 对象的两个属性

- 1) 类型(type): 决定对象的取值范围&支持的操作.
- 强类型语言
- 内置函数type(x): 返回对象x的类型
- 对象是类型的一个实例(instance)
- 2) 值(value): 值可变/不可变, 取决于其类型
- 不可变(immutable): 有些对象一旦创建, 其值不可变, 如: 数字、字符串、元组、.....
- 可变(mutable): 对象的值可以改变, 如: 列表、字典、.....

# Python的内置对象类型

对象类型	示例
数字	1234, 3.14, 3+4j
字符串	'Fudan', "I'm student", '''Python'''
列表	[1, 2, 3] ['a', 'b', ['c', 2]]
元组	(2, -5, 6)
字典	{1:'food',2:'taste',3: 'import'}
集合	{'a', 'b', 'c'}
文件	f = open('data.dat', 'r')
布尔型	True, False
空类型	None
编程单元类型	函数(def), 模块, 类(class)



## 1.4.2 如何描述和引用对象——变量

### Literal(字面值)

- 某个内置对象类型的值(固定不变), Python解释器自动识别其类型.
- 字符串字面值: 'Hello World'
- 整数字面值: 2016, 0o177, 0xda80, 0b10010111
- 浮点数字面值: 3.14, 10., .001, 3.14e-10, 1.0e100
- 复数字面值: 3.14j, 1+10j

## □ 表达式(expression)

- 由对象和运算符(operator)组成

e.g.  $(3*4+5)*6$       # '\*'代表乘法运算符

## □ 变量(variable): 某个对象的引用(reference)

e.g. `>>> x = 3`      # 变量x是对象3的引用

- 在程序中通过变量名描述.
- 使用前无需声明类型, 由系统自动判定, 支持的运算由类型决定.
- 变量在使用前必须定义(赋值)

# 赋值语句

LHS = RHS (Right Hand Side)

- 变量出现在LHS, 表示给该变量赋值, 即保存RHS对象的引用(地址)

e.g. `>>> a = (3*4+5) * 6` # 变量a是对象102的引用

`>>> a = a * a` # 变量a更新为 $102^2$ 的引用

- 变量出现在RHS时, 表示引用该变量所指对象的值.

e.g. `>>> b = a` # 把 $102^2$ 赋给变量y, y也是 $102^2$ 的引用

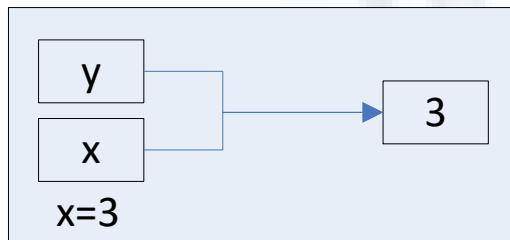
- 同时给多个变量赋值

e.g. `i, j = 1, 0`

`x, y, z = 'Mike', 4, 3.14`

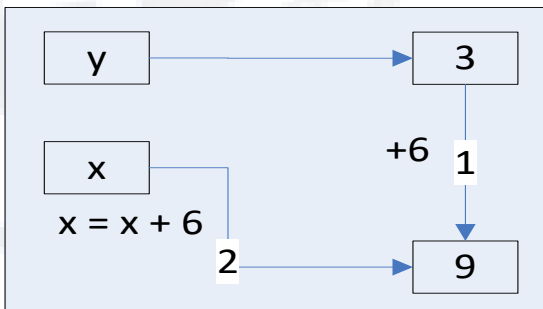
- ❑ 变量表示对象的引用→允许多个变量指向同一个对象.
- ❑ 给变量赋予新的值后, 变量指向新的对象.

```
>>> x = 3
>>> id(x)
1786684560
>>> y = x
>>> id(y)
1786684560
```



```
>>> x = x + 6
>>> id(x)
1786684752
>>> y
3
>>> id(y)
1786684560
```

对变量赋其它值  
→产生新对象、



- Python是一种动态类型语言: 变量的类型可随时变化, 变量实际保存的是对象的引用.

```
>>> x = 3      # 变量x是整数类型
>>> print(type(x))  # 函数的嵌套调用
<class 'int'>
>>> x = 'Hello world.' # x变为字符串类型
>>> print(type(x))
<class 'str'>
>>> x = [1, 2, 3]    # x变为列表类型
>>> print(type(x))
<class 'list'>
>>> isinstance(3, int)
True
>>> isinstance('Hello world', str)
True
```

内置函数isinstance(obj, class): 测试对象obj是否为指定类型class的实例

## □ Python具有自动内存管理功能

- 跟踪所有对象, 并自动删除不再有变量指向的对象.
- 可显式使用del语句删除不再需要的对象(回收所占用的内存).

e.g. `del x`

## □ 两个有用的内置函数(built-in function, BIF)

- `help([obj])`: 查看obj相关的帮助

e.g. `help()`, `help(id)`

- `dir([obj])`: 查看obj相关的属性列表

e.g. `dir()`, `dir(x)`



# 变量命名规则

- ❑ 标识符: 变量、函数、类、模块和其它对象的名字.
- ❑ 必须以字母或下划线开头, 其后的字符可以是字母、下划线或数字.
- ❑ 以下划线开头的变量在Python中有特殊含义.
- ❑ 变量名中不能有空格和标点符号.  
e.g. 99var, It'OK, first-class ✖
- ❑ 标识符对英文字母的大小写敏感.  
e.g. student和Student代表不同变量.

## ❑ Python关键字不能用作变量名

```
>>> import keyword
>>> print(keyword.kwlist)
>>> keyword.iskeyword('str')    # 判断str是否为关键字
>>> help()    # help函数的另外一种用法
help> keywords
help> quit
```

- ❑ 不建议用作变量名: 系统内置的模块名/类型名/函数名、已导入的模块名及其成员名.
  - 否则会改变其类型和含义.
  - 可通过`dir(__builtins__)`查看所有内置模块、类型和函数.

# Python 3's keywords

---

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

- ❑ You don't have to memorize this list. In most development environments, keywords are displayed in a different color; if you try to use one as a variable name, you'll know.

- 使用能反映变量含义的名字(e.g. area, keys). i, j, k仅用在循环结构中.
- 变量名使用小写字母, 由多个单词构成时, 以下划线连接  
e.g. lower\_case\_with\_underscores

## 1.4.3 基本输入输出函数

`input(prompt)`

参数, 可替换

- 首先输出prompt, 然后等待用户输入, 直到用户按回车结束, 返回用户输入的字符串(不包括最后的回车)

带默认值的参数

`print(value1, value2, ..., sep=' ', end='\n', file=sys.stdout)`

- 将多个值转换为字符串并且输出到相应的文件file, 输出项之间以sep分隔, 最后以end结束.
- sep默认为空格, end默认为换行, file默认为标准输出(显示器).

```
>>> x=input('Please input:')  
Please input: 3 ✓  
>>> x  
'3'  
>>> int(x)  
3
```

```
>>> x,y,z = 'Mike',4,3.14  
>>> print(x, y, z)  
Mike 4 3.14  
>>> print(x, y, z, sep=',')  
Mike,4,3.14  
>>> print(x,y,z,sep=',',  
end='*****')  
Mike,4,3.14*****  
>>> print('Mike', y, z)  
Mike 4 3.14
```



# 像普通程序一样运行——双击

## □ 如何避免程序闪退?

- 在程序的最后添加一行代码:

```
input('Press <Enter>')
```

- [illegible]



## 整数字面值(integer literal)

- 十进制整数: 0, -1, 9, 123
- 二进制整数: 以**0b**开头, 包括数字0和1, 如: 0b101, 0b100, 0b10010111
- 八进制整数: 以**0o**开头, 包括数字0~7, 如: 0o35, 0o11
- 十六进制整数: 以**0x**开头, 包括数字0~9以及[a~f](a, b, c, d, e, f), 如: 0x10, 0xfa, 0xabcdef (大小写均可)



# (有限精度)浮点数(float)

## □ “浮点数”字面值: 用于表示实数

- 小数表示: 3.14, 10., .001
- 科学计数法:  $15e-2$  ( $15 \times 10^{-2} = 0.15$ ),  $3.14e-10$ ,  $1.0e100$

## □ 计算机内部采用二进制表示 $\Rightarrow$ 有转换误差

e.g.  $a = 0.1 + 0.1 + 0.1$

$a == 0.3$     # 结果为 False

$\text{abs}(a - 0.3) < 1e-8$  # 浮点数差的绝对值小于一个很小的实数 $\Rightarrow$ 即可判断二者相等

# 复数(Complex)

## 复数字面值

- ⑩ 一个复数包括实部(real)和虚部(imaginary)
  - 实部: 浮点数/整数
  - 虚部: 浮点数/整数后面跟j/J
- ⑩ 合法的复数:  $3+4j$ ,  $1.1+2.2j$ ,  $3+4.2j$ ,  $3-4.2j$ ,  $2.5j+3$ ,  $4j$ ,  $0j$
- ⑩ 非法的复数:  $j$ ,  $4*j$

## □ 复数对象的属性和方法

```
>>> c.real # 查看复数实部
```

```
8.0
```

```
>>> c.imag # 查看复数虚部
```

```
10.0
```

```
>>> a.conjugate() # 返回共轭复数
```

```
(3-4j)
```

# 算术运算符

算术运算符	举 例
$+$ , $-$	$35+4$ , $2.5+3.5$
$*$	$(35+4)*5$ , $3.4*5.2+4$ #注意*不能省略
$//$ (整除) $/$ (真除)	$5//2 \Rightarrow 2$ , $-5//2 \Rightarrow -3$ $5/2 \Rightarrow 2.5$ , $-5/2 \Rightarrow -2.5$
$**$ (幂运算)	$4**2 \Rightarrow 16$ , $4**0.5 \Rightarrow 2.0$
$\%$ (取余/求模)	$10\%5 \Rightarrow 0$ , $20\%3 \Rightarrow 2$ , $-20\%3 \Rightarrow 1$ #余数 $\geq 0$ $2.3\%2.2 \Rightarrow 0.1$ #可以对实数取余
$+$ , $-$ (单目运算, 取正/负)	<pre>&gt;&gt;&gt; c = 3+4j &gt;&gt;&gt; -c (-3-4j) &gt;&gt;&gt; +c (3+4j)</pre>



□ 复数支持常用的数学运算: +, -, \*, /, \*\*

```
>>> a = 3+4j
```

```
>>> b = 5+6j
```

```
>>> c = a+b
```

```
>>> c
```

```
(8+10j)
```

```
>>> a*b # 复数乘法
```

```
(-9+38j)
```

```
>>> a/b # 复数除法
```

```
(0.6393442622950819+0.03278688524590165j)
```

# 复合赋值运算符

## □ 将赋值运算符和算术运算符结合使用

$a \theta = b$     #  $\theta$  代表算术运算符,  $a = a \theta (b)$

$+=$     $-=$     $*=$     $/=$     $//=$     $**=$     $\%=$

```
>>> a = (3*4+5)*6
```

```
>>> a
```

```
102
```

```
>>> a /= 6
```

```
>>> a
```

```
17.0
```

```
>>> a = (3*4+5)*6
```

```
>>> a //= 6
```

```
>>> a
```

```
17
```

```
>>> a -= 15
```

```
>>> a
```

```
2
```

```
>>> a **= 2
```

```
>>> a
```

```
4
```

```
>>> a *= 2+3
```

```
>>> a
```

```
20
```

```
>>> a %= 3
```

```
>>> a
```

```
2
```

# 优先级 & 结合性

$$3-4**2+10\%5+8*6//7$$

- 优先级:  $(+, -) < (*, /, //, \%) < (+, -) < (**)$
- 结合性: 从左向右

$$\text{Step1. } 4**2 \Rightarrow 3-16 \Rightarrow -13+10\%5+8*6//7$$

$$\text{Step2. } 10\%5 \Rightarrow -13+0 \Rightarrow -13+8*6//7$$

$$\text{Step3. } 8*6 \Rightarrow -13+48//7$$

$$\text{Step4. } 48//7 \Rightarrow -13+6 = -7$$

- 通过"()"改变运算顺序, 可嵌套 e.g.  $((1+2)*(3+4))**2$



# 混合运算

## □ 算术运算符支持混合运算, 即不同类型对象之间的运算

e.g.  $2+2.0$  # 浮点数+整数, 结果为4还是4.0?

## □ 类型转换规则

- 若表达式包含complex对象, 则其它数字类型转换为complex对象, 结果为complex对象, e.g.  $(1+2j)+3=4+2j$
- 若表达式包含float对象, 则其它数字类型转换为float对象, 结果为float对象, e.g.  $1.2+3=4.2$

[例1] 用户输入一个三位自然数, 计算并输出其百位、十位和个位上的数字.

- 创建一个新的python程序, 命名为three\_digits.py, 输入以下代码:

```
x = int(input('请输入一个三位数: ')) # 假如输入357
hundred = x//100 # 357//100得3, 即百位数
ten = x//10 % 10 # 357//10得35, 35%10得5, 即十位数
unit = x % 10 # 357%10得7, 即个位数
print(hundred, ten, unit)
```

- 强调: 变量的命名应反映其含义. 如: hundred, ten, unit.
- 函数的嵌套调用.
- #引导单行注释, '''.....'''引导多行注释.

# 注释

- 随着程序规模增加, 将会变得非常难懂.
- 因此, 给程序添加一些自然语言的解释, 如: 程序在做什么, 为什么这样做, 将会很有帮助.
- 从#开始一直到行尾的内容都会被忽略——不会对程序的执行产生影响.

# 布尔(bool)

## □ 整数类型的特例

- True: 1, False: 0
- 非0整数 $\rightarrow$ bool型: True; 0 $\rightarrow$ bool型: False
- 非0 $\leftrightarrow$ True, 0 $\leftrightarrow$ False

```
>>> int(True), int(False) # 内置函数int(obj): 将obj转换为整型  
(1, 0)
```

```
>>> bool(4), bool(0) # 内置函数bool(obj): 将obj转换为bool型  
(True, False)
```

# 比较运算符

## □ $<$ , $<=$ , $>$ , $>=$

- 比较数字对象值的大小, 结果为bool值: True, False
- 可连用, 连用时**没有传递性**, 仅比较相邻数

e.g.  $x < y < z$  表示  $x < y$  且  $y < z$

$x < y > z$  表示  $x < y$  且  $y > z$

```
>>> x, y, z = 12, 18, 13 # 将=右侧的值一一对应赋值给左边的变量
```

```
>>> x < y, x <= y, x > y, x >= y # 运算结果为(True, True, False, False)
```

```
>>> x < y < z, x < y > z # 运算结果为(False, True)
```

## □ ==, !=

- 比较对象的值是否相等, 结果为bool值: True, False

```
>>> x, y, z = 12, 18, 13
```

```
>>> x==y # 运算结果为False
```

```
>>> x!=y # 运算结果为True
```

## □ 优先级

- 比较运算符 < 算术运算符

e.g.  $2**2 < 2+2$  # 先计算'<'两端的表达式, 然后比较计算结果



# 逻辑运算符and, or, not

- 运算结果均为bool值

- 优先级:

  - $or < and < 比较运算符 < 算术运算符 < not$

- $x$  and  $y$ : 逻辑与

  - 只有 $x$ 和 $y$ 都为True, 结果才为True, 否则为False.

  - 短路特性:  $x$ 为True时才会计算 $y$ ; 若 $x$ 为False, 则不会计算 $y$ .

e.g.  $i \neq 0$  and  $j/i > 3$  """  $i$ 为0时,  $i \neq 0$ 为False, 则不会计算 $j/i$ , 避免除数为0的错误"""

## □ x or y: 逻辑或

- 只有x和y都为False, 结果才为False, 否则为True.
- 短路特性: 只有x为False时才会计算y; 若x为True, 则不会计算y.

e.g. `ans=='Y' or ans=='y'` *'''ans为'Y'时, ans=='Y'为True, 则不必计算ans=='y', 减少计算量.'''*

## □ not x: 逻辑非

- x为True, 则not x为False;
- x为False, 则not x为True.

e.g. `i!=0` 与 `not i==0` 等价



# 常用的内置数学函数

函 数	含 义	实 例	结 果
abs(x)	x的绝对值.若x为复数,返回模	abs(-1.2) abs(1-2j)	1.2 2.2360679.....
divmod(a,b)	返回a除以b的商和余数	divmod(5, 3)	(1, 2)
pow(x,y[,z])	返回 $x^{**}y$ . 如有z,则为 $\text{pow}(x,y)\%z$	pow(2, 10) pow(2, 10, 10)	1024 4
round(number [,ndigits])	四舍五入取整. 如有ndigits, 则 保留ndigits位小数	round(3.14159) round(3.14159, 4)	3 3.1416
max(arg1,arg2,*args)	取最大值	max(1,7,3,15,14)	15
min(arg1,arg2,*args)	取最小值	min(1,7,3,15,14)	1
sum(iterable[,start])	求和, 如有start, 表示加上start	sum((1,2,3)) sum((1,2,3),44)	6 50



# math模块

import math

math.pi: 数学常量 $\pi$

math.e: 数学常量 $e$

方 法	含 义	实 例	结 果
<code>math.fabs(x)</code>	绝对值,返回float	<code>math.fabs(-3)</code>	3.0
<code>math.ceil(x)</code>	大于等于x的最小的整数	<code>math.ceil(1.2),</code> <code>math.ceil(-1.6)</code>	2, -1
<code>math.floor(x)</code>	小于等于x的最大的整数	<code>math.floor(1.8),</code> <code>math.floor(-2.1)</code>	1, -3
<code>math.trunc(x)</code>	截取为最接近0的整数	<code>math.trunc(1.2),</code> <code>trunc(-2.8)</code>	1, -2
<code>math.factorial(x)</code>	整数 $x(≥0)$ 的阶乘	<code>math.factorial(5)</code>	120
<code>math.sqrt(x)</code>	$x(≥0)$ 的平方根	<code>math.sqrt(2)</code>	1.414213.....
<code>math.exp(x)</code>	$e^{**}x$	<code>math.exp(5)</code>	148.413159...
<code>math.log(x[,base])</code>	以base为底的对数,没有base则为以e为底的自然对数	<code>math.log(math.e**2),</code> <code>math.log(4,2)</code>	2.0, 2.0

## math模块: 三角函数和角度转换

方 法	含 义	实 例	结 果
<code>sin(x)</code>	正 弦	<code>math.sin(pi/2)</code>	1.0
<code>cos(x)</code>	余 弦	<code>math.cos(2*pi)</code>	1.0
<code>tan(x)</code>	正 切	<code>math.tan(pi/4)</code>	0.999999...
<code>asin(x)</code>	反 正 弦	<code>math.asin(1)</code>	1.570796...
<code>acos(x)</code>	反 余 弦	<code>math.acos(1)</code>	0.0
<code>atan(x)</code>	反 正 切	<code>math.atan(1)</code>	0.785398...
<code>degrees(x)</code>	弧 度 $\Rightarrow$ 角 度	<code>math.degrees(pi)</code>	180.0
<code>radians(x)</code>	角 度 $\Rightarrow$ 弧 度	<code>math.radians(90)</code>	1.570796...

[例2] 已知三角形的两边长及其夹角, 求第三边长.

公式: 
$$c = \sqrt{a^2 + b^2 - 2ab \cos(\theta)}$$

程序:

```
import math
```

```
x = input('输入两边长及夹角(度), 以逗号分隔: ')\na, b, sita = eval(x) # 序列解包, 从字符串变为数值, 为多个变量赋值\nc = math.sqrt(a**2+b**2-2*a*b*math.cos(sita*math.pi/180))\nprint('c =', c)
```

函数/方法的嵌套调用: 以`math.cos(...)`的返回值与其它变量进行数学运算, 计算结果作为`math.sqrt(...)`的参数, 最终结果由`c`引用.

## □ 熟练掌握eval()函数的使用——对多个变量的赋值

```
>>> a,b=eval('2333,3.14') >>> a,b=eval(input())
```

```
>>> a,b
```

```
(2333, 3.14)
```

```
>>> x=input()
```

```
2333,3.14 ✓
```

```
>>> x
```

```
'2333,3.14'
```

```
>>> a,b=eval(x)
```

```
>>> a,b
```

```
(2333, 3.14)
```

```
>>> a,b=eval(input())
```

```
2333,3.14 ✓
```

```
>>> a,b
```

```
(2333, 3.14)
```

```
2333 3.14 ✓
```

```
Traceback (most recent call  
last):
```

```
File "<pyshell#0>", line 1, in  
<module>
```

```
    a,b=eval(input())
```

```
File "<string>", line 1
```

```
    2333 3.14
```

```
    ^
```

```
SyntaxError: unexpected EOF  
while parsing
```

❖ 如果eval的参数来自于input函数,则要求输入时,各数据项之间以逗号分隔!!!

## 1.4.4 字符串(str)

- ❑ 字符串字面值: 不可变序列 e.g. 'Hi!'
- ❑ 单行文本: 使用一对单引号/双引号括起的字符序列

- Python3的字符默认为Unicode编码⇒可包含中文字符
- '或"定义的字符串不能跨越多行
- 一般采用'定义
- 惯例: 若字符串本身包含", 则使用'定义, 反之亦然

e.g. 'abc', "Python程序设计基础", "What's your name?", "'Thank you!'"  
she said. '

## □ 多行文本: 使用''' '''或""" """括起的字符序列

- 用于长字符串, 或在程序中表示较长的注释.

```
>>> a = '''first line \nline 2
line 3
last line.'''
>>> a
'first line \nline 2\nline 3\nlast line.'
>>> print(a)
first line
line 2
line 3
last line.
```

## □ 空串: ""或"""

`info=""` # 用来确定变量`info`的类型和初值



# 字符串转义

- 若字符串本身包含单/双引号⇒通过在引号前加上'\'表示其为字符串的一部分,而非字符串的界定符. e.g. 比较`c=""`与`c='\'`
- 控制字符也通过转义描述

特殊字符	含义	特殊字符	含义
\'	单引号	\"	双引号
\\	\	\t	制表符
\n	换行符		
\ddd	3位八进制数 对应的字符	\xhh	2位十六进制数 对应的字符

- ❑ 字符串界定符前加字母r表示原始字符串：其中的特殊字符不需要转义, 但最后一个字符不能是\.

```
>>> a = 'C: \\Program Files (x86)\\Python35\\Lib'
```

```
>>> a = r'C: \Program Files (x86)\Python35\Lib'
```

```
>>> a
```

```
'C: \\Program Files (x86)\\Python35\\Lib'
```

- ❑ 字符串相关内置函数(见下页)

函 数	含 义	实 例	结 果
bin(x)	将整数x转换为二进制字符串	bin(4)	'0b100'
oct(x)	将整数x转换为八进制字符串	oct(25)	'0o31'
hex(x)	将整数x转换为十六进制字符串	hex(75)	'0x4b'
str(obj)	把对象obj转换为字符串	str(16), str(1/2)	'16', '0.5'
int(x[, d])	把数字x截取为整数/将基数为d的数字型字符串x转换为整数, 若无d缺省为十进制	int(3.14), int('314') int('ff', 16)	3, 314 255
float(x)	将对象x(数字或字符串)转换为浮点数	float(3), float('3.14') float('12e-2')	3.0, 3.14 0.12
len(obj)	返回序列类对象obj包含的元素个数	len('Hello World'), len('')	11, 0
ord(s)	返回字符s(长度为1的字符串s)的Unicode码(ASCII码是其子集)	ord('a'),ord('A'),ord('你') ord('0'), ord('\n')	97, 65, 20320 48, 10
chr(x)	返回Unicode编码为x的字符(len=1)	chr(65),chr(10),chr(20320) chr(ord('d')-ord('a')+ord('A'))	'A', '\n', '你' 'D'



# 字符串支持的运算符: +, \*, %

□ +: 字符串合并, 生成新的字符串

```
>>> a='abc'+'123'    #生成新对象 'abc123'
```

```
>>> b='\141b\x61'+'123' # 'aba123', ord('a')=(97)10=(141)8=(61)16
```

□ \*: 字符串和整数相乘→字符串的内容重复多次, 整数小于等于0时生成空字符串

```
>>> a='*-'*10    # '*-*-*-*-*-*-*-*'
```

```
>>> a=4*'NE'    # 'NENENE'
```

# 字符串格式化%: 精简版

□ `format_string%obj`: 把对象obj按格式要求format\_string转换为字符串.

```
>>>"My name is %s, and my age is %d"%('Dong Fuguo', 38)
```

```
'My name is Dong Fuguo, and my age is 38'
```

```
>>> "%d: %c"%(65, 65)  # %d: 整数→整数字符串, %c: 整数→单个字符
```

```
'65: A'
```

```
>>> a=3.6674  #变量a: float型
```

```
>>> '%7.3f'%a  # %f: 把浮点型数值→浮点型字符串
```

```
' 3.667'  #7.3: 格式化, 字符串长度为7, 小数点后保留3位
```



HW1\_2

格式字符	含义	示例	结果
%s	转换为字符串	'Hello %s'% 'Mike'	'Hello Mike'
%[-][width]d	转换为整数字符串, 宽度为width, 负号代表左对齐	'Length: %-4d'% 45	'Length:45 '
%c	转换为字符chr(num)	'%c' % 65	'A'
%[width][.precision]f	转换为浮点数字符串, 宽度为width, 小数点后precision位	'%f' % math.pi '%.2f' % math.pi '%8.4f' % math.pi	'3.141593' '3.14' ' 3.1416'

# 字符串格式化函数format(...)

□ str.format(obj): 把对象obj转换为字符串。

```
>>>"My name is {}, and my age is {}".format('Dong Fuguo',  
38)
```

```
'My name is Dong Fuguo, and my age is 38'
```

```
>>> a=3.1415926
```

```
>>> 'PI={}'.format(a)
```

```
'PI=3.1415926'
```

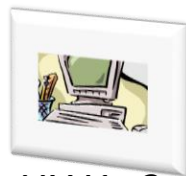
优势: 不需要指定转换格式,  
根据对象类型自动转换!

```
>>> "{}: {:c}".format(65, 65)  # {:c}: 整数→单个字符
```

```
'65: A'
```

```
>>> 'PI={:7.2f}'.format(a)  # {:7.2f}: 字符串长度为7, 小数点后保留2位
```

```
'PI=    3.14'
```



HW1\_2

# 整数 vs 整数字符串?

```
>>> i = input('Enter an integer: ')
Enter an integer: 12
>>> j = i + 2
Traceback (most recent call last):
  File "<pyshell#15>", line 1, in <module>
    j = i + 2
TypeError: must be str, not int
```

## □ 12与'12'的区别

- 对计算机而言, 它们是完全不同的两个数据!!!
- 原因: 机内表示不同

整数12: 在计算机内部, 保存为它所对应的二进制序列1100

整数字符串'12': 保存为字符'1'的ASCII码49所对应的二进制序列110001, 和字符'2'的ASCII码50所对应的二进制序列110010



## 1.4.5 运算符和表达式

### 合法表达式

- 单个任何类型的对象或常量, e.g. `a`, `10.0`, `1`, `'a'`
- 运算符连接的变量、常量、函数调用的任意组合, e.g. `a+3`, `pow(2, 10)*5`

运算符示例	功能说明
$x+y$	算术加法; 列表、元组、字符串合并
$x-y$	算术减法; 集合差集
$x*y$	乘法; 序列重复
$x/y$	除法(Python 3: 真除法)
$x//y$	求整商
$-x$	相反数
$x\%y$	余数(对实数也可进行余数运算); 字符串格式化
$x**y$	幂运算; 字符串重复
$x<y, x\leq y, x>y, x\geq y$	大小比较(可连用); 集合的包含关系比较
$x==y, x!=y$	相等(值)比较, 不等(值)比较
$x \text{ or } y$	逻辑或(只有x为假才会计算y)
$x \text{ and } y$	逻辑与(只有x为真才会计算y)
$\text{not } x$	逻辑非
$x \text{ in } y, x \text{ not in } y$	成员测试运算符
$x \text{ is } y, x \text{ is not } y$	对象实体同一性测试(地址)
$ , ^, \&, <<, >>, \sim$	位运算符
$\&,  , ^$	集合交集、并集、对称差集

# 运算符的优先级

□ 赋值运算符 < or < and < 比较运算符 < 算术运算符 < not

$=, +=, -=, *=, /=, //=, \dots$

leap = year % 4 == 0 and year % 100 != 0 or year % 400 == 0

1

5

4

3

5

4

2

5

4

■ 除优先级外, 还要考虑and和or的短路特性.

### [例3] 任意输入三个英文单词, 按字典顺序输出.

```
s = input('word1, word2, word3 = ') #输入3个单词, 以空格分隔
word1, word2, word3 = s.split() #以空格为分隔符划分字符串
if word1 > word2: #字符串按ASCII编码大小逐个比较
    word1, word2 = word2, word1 #交换值
if word1 > word3:
    word1, word3 = word3, word1
if word2 > word3:
    word2, word3 = word3, word2
print(word1, word2, word3)
```

## □ 熟练掌握split()函数的使用——对多个变量赋值的另外一种方法

- 默认以空白符作为分隔符.
- 如果要以其它符号作为分隔符, 需要指定参数.

```
>>> w1, w2, w3 = input().split()
apple pear banana ↵
>>> w1, w2, w3
('apple', 'pear', 'banana')
>>> w1, w2, w3 = input().split(',')
apple,pear,banana ↵
>>> w1, w2, w3
('apple', 'pear', 'banana')
```

## □ 单分支选择结构

if 条件表达式:     # ':'标识语句块(缩进)即将开始

缩进

语句块

# python程序依靠语句块的缩进, 体现代码之间的逻辑关系

- 语句块: 一组语句
- 如果条件表达式的运算结果为True, 则运行语句块; 为False, 则什么也不做.
- 条件表达式的运算结果只要不是0或空, Python解释器均认为与True等价;
- 与False等价: 0(或0.0、0j等), None



## 1.4.6 常用内置函数

- 内置函数(BIF): 不需要导入任何模块即可使用的函数.
- 列出所有内置函数和内置对象的命令

```
>>> dir(__builtins__)
```

- 优先考虑使用内置函数
  - BIF成熟、稳定、执行速度快.
- 对于初学者而言, 函数dir()和help()很有用

```
>>> import math  
>>> dir(math)  
>>> help(math.sqrt)
```

## □ max(iterable), min(iterable), sum(iterable)

- 计算可迭代对象中所有元素的最大值、最小值、所有元素和.
- sum()只支持数值型元素的可迭代对象
- max()和min()则要求元素之间可比较大小.

```
>>> import random          # 导入随机数模块
>>> a=random.sample(range(1,100),10) # 生成10个[1,100)随机数
>>> a
[72, 26, 80, 65, 34, 86, 19, 74, 52, 40]
>>> print(max(a), min(a), sum(a), sum(a)/len(a) )
86 19 548 54.8
```



# random模块的常用方法

函 数	含 义
random()	返回[0, 1)区间的一个随机实数
uniform(a, b)	返回[a, b]区间的一个随机实数
randint(a, b)	返回[a, b]区间的一个随机整数
choice(seq)	随机返回序列seq中的一个元素
sample(seq, n)	采样：从序列seq中随机选择n个不同元素，返回列表. 要求：seq的元素个数>n

## 1.4.7 对象的删除

Python具有自动内存管理功能

- 解释器跟踪所有对象的引用情况, 一旦发现某个对象不再有任何变量引用, 垃圾回收机制会回收该对象, 并释放内存资源.

del语句

- 显式解除变量与所指向对象之间的绑定.

```
>>> y=3
>>> z=y
>>> print(y)
```

3

```
>>> del y      #删除变量y
```

```
>>> print(y)
```

```
Traceback (most recent call
last):
```

```
File "<pyshell#52>", line
1, in <module>
```

```
    print(y)
```

```
NameError: name 'y' is not
defined
```

```
>>> print(z)
```

3

```
>>> del z
>>> print(z)
```

```
Traceback (most recent call
last):
```

```
File "<pyshell#56>", line
1, in <module>
```

```
    print(z)
```

```
NameError: name 'z' is not
defined
```



## 1.4.9 模块(modules)导入与使用

- Python默认安装仅包含部分基本或核心模块, 但用户可以安装大量的扩展模块, pip是管理模块的重要工具.
  - Pip的安装请参考<http://www.tuicool.com/articles/eiM3Er3/>
- Python启动时, 仅加载很少一部分模块(包括一些内置模块builtins), 其它模块在需要时由程序员显式加载(可能需要先安装).
  - 减小运行的压力, 仅加载真正需要的模块和功能, 且具有很强的可扩展性.
- 可使用`sys.modules.items()`显示所有预加载模块的信息.



# 模块的使用

## 1. import 模块名 [as 别名]

- 访问模块中的对象: 模块名.对象, 别名.对象

```
>>> import math
```

```
>>> math.sin(0.5)    # 求0.5的正弦
```

```
>>> import random as rnd
```

```
>>> x=rnd.random( )  # 获得[0, 1)区间的随机小数
```

## 2. from 模块名 import 对象名[ as 别名]

- 仅从模块中导入特定对象, 访问该对象时不再需要包括模块名
- 可减少查询次数, 提高执行速度

```
>>> from math import sin
```

```
>>> sin(3)    #math.sin(3)
```

```
0.1411200080598672
```

```
>>> from math import sin as f    # 别名
```

```
>>> f(3)
```

```
0.1411200080598672
```

## 3. from math import \*

- 从模块导入所有对象
- 多个模块有相同对象名时会造成混淆⇒谨慎使用



# 1.5 Python代码编写规范

## 缩进

- ⑩ python程序依靠语句块的缩进体现代码之间的逻辑关系.
- ⑩ 语句块的开始
  - 选择结构/循环结构/函数定义/类定义等, 行尾出现“:” 表示缩进的开始.
- ⑩ 缩进结束表示代码块的结束(手动控制).
- ⑩ 同一级别的代码块, 其缩进量必须相同.
- ⑩ 以4个空格为基本缩进单位, 不要采用制表符来缩进.

- IDLE中, 代码块的缩进/反缩进

菜单Format→Indent Region(Ctrl+)]/Dedent Region(Ctrl+[)

## □ 添加空格与空行以提高代码可读性

- 运算符两侧使用空格分开 e.g. 1 + 2
- 在逗号后面添加一个空格 e.g. 1, 2, ...
- 不同功能的代码块之间、不同的函数定义之间添加一个空行.



## □ 换行

- 如果一行语句太长, 可在行尾加上\换成多行, 更好的方法是使用括号来包含多行内容.

e.g. `if a > b and a >c and a > 5 and b > 5 and c > 5 \`  
`and b > c:`  
 `print('blah')`  
`if (a > b and a >c and a > 5 and b > 5 and c > 5`  
`and b > c) :`  
 `print('blah')`

## □ 适当添加注释

- Comments are most useful when they document non-obvious features of the code.

如下注释是冗余的、无用的:

```
v = 5    # assign 5 to v
```

而这条注释就包含了代码中不包含的有用信息:

```
v = 5    # velocity in meters/second.
```

- 在IDLE开发环境中, 快速注释/解除注释大段内容

菜单Format→Comment Out Region (ALT+3)/Uncomment Region (ALT+4)

## □ 每个import语句只导入一个模块

- 首先导入Python标准库模块, 如: os, sys, re;
- 然后导入第三方扩展库, 如: numpy, scipy;
- 最后导入自己定义和开发的本地模块.