

# CS001 编程零基础 Python语言入门

第一讲

# 本讲内容

计算机编程基础知识

Python下载安装及IDLE使用

程序输出与错误信息

内存与进制

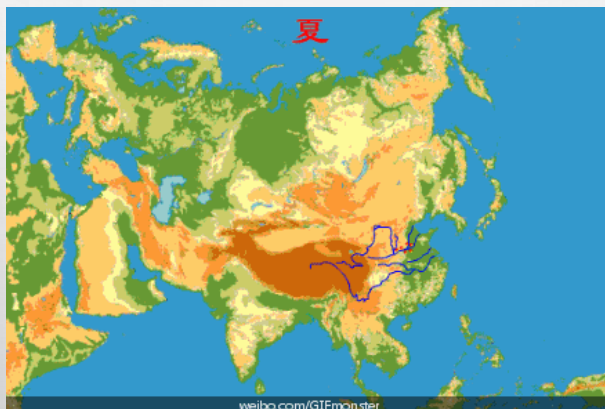
绘图

# 为什么要学编程？

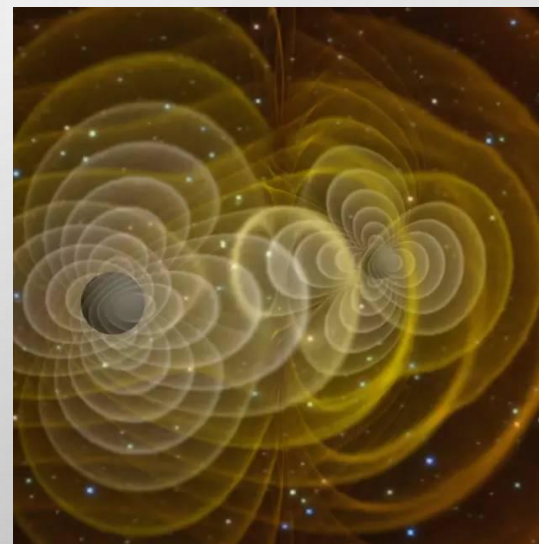
- 计算机已经渗透到社会各个领域，社会需要大量计算机人材。
- 大学学习中，各专业都需要一定的数据处理能力。
- 掌握计算机编程语言会让你在各行各业都炙手可热，这甚至和掌握一门外语一样重要！



<http://www.yjbys.com/qiuzhizhinan/show-517960.html>



图片来源  
[http://www.360doc.com/content/13/0207/12/1208701\\_264674392.shtml](http://www.360doc.com/content/13/0207/12/1208701_264674392.shtml)



引力波数据居然是用 Python 分析的

<https://www.oschina.net/news/70669/gwpy-ligo-analyze-gravitational-waves-data>

# 为什么需要编程？

计算机没有智能，只能按照预设的程序行动。

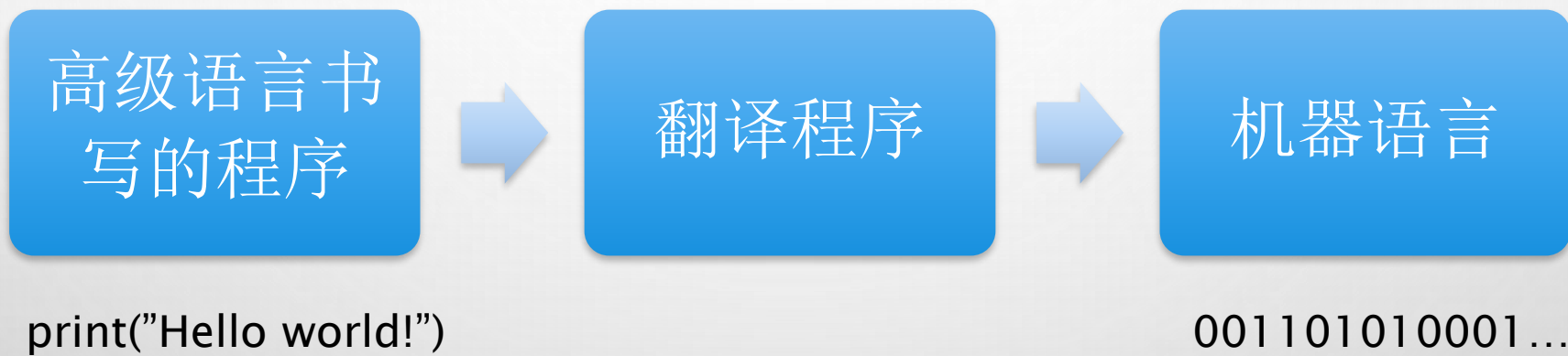
# 为什么需要编程？



要让计算机完成更复杂的任务，  
程序一般也会更复杂。



# 程序设计语言（编程语言）



# Why Python?

简单易学

编程效率  
高

数据处理  
能力强

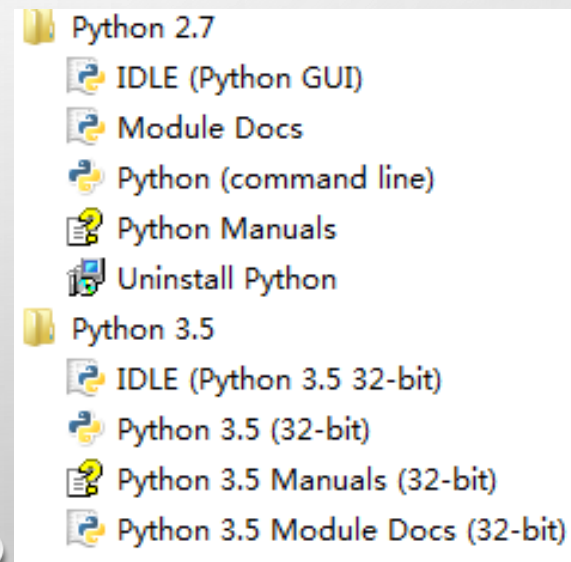
# Python3下载与安装

- Python解释器:

- [HTTPS://WWW.PYTHON.ORG/DOWNLOADS/](https://www.python.org/downloads/)

根据所使用的平台（WINDOWS / MAC OS X）下载相应的版本

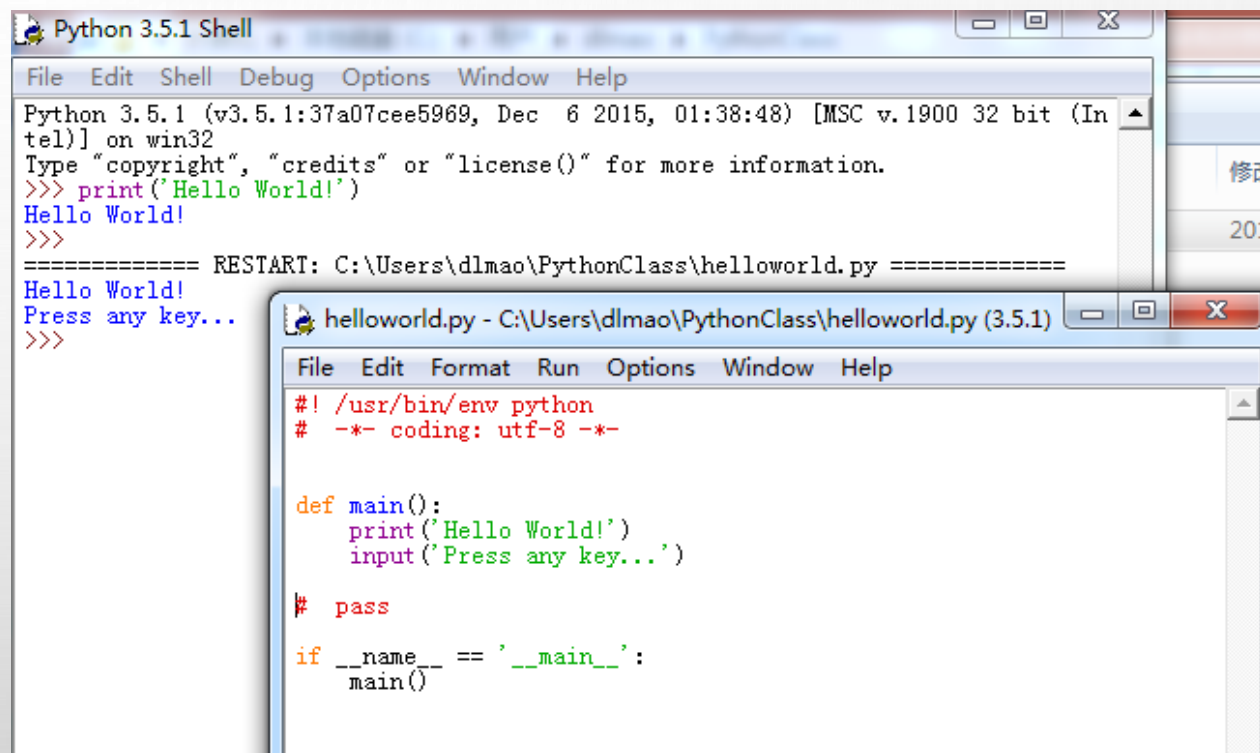
- INSTALL NOW: 按照缺省设置安装





## 集成开发环境IDLE

- 将编辑器、解释器、调试器集合在一起
- 下载安装的Python环境中包括了IDLE
- IDLE是一个通过键入文本与程序交互的途径
- >>>是Python提示符，是等待你键入信息时显示的符号。



The image shows two overlapping windows from the Python 3.5.1 IDLE environment. The background window is the 'Python 3.5.1 Shell', which displays the Python interpreter's startup message and the execution of a 'print' statement. The foreground window is a script editor titled 'helloworld.py - C:\Users\dlmao\PythonClass\helloworld.py (3.5.1)', showing the source code for a 'Hello World' program with a user input prompt.

```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print('Hello World!')
Hello World!
>>>
===== RESTART: C:\Users\dlmao\PythonClass\helloworld.py =====
Hello World!
Press any key...
>>>
```

```
helloworld.py - C:\Users\dlmao\PythonClass\helloworld.py (3.5.1)
File Edit Format Run Options Window Help
#!/usr/bin/env python
# -*- coding: utf-8 -*-

def main():
    print('Hello World!')
    input('Press any key...')

# pass

if __name__ == '__main__':
    main()
```

## 交互模式下，单个指令执行

在>>>后面键入一条指令

小窍门：组合键**Alt+P**可以  
显示历史命令中的上一条。  
下一条是**Alt+N**。

+除了表示算术加法外，也  
可以表示字符串连接。

\*除了表示算术乘法外，也  
可以表示对象重复多次。

```
>>> print('Hello')
Hello
>>> Pront('Hello')
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    Pront('Hello')
NameError: name 'Pront' is not defined
>>> 5 + 3
8
>>> 5 * 3
15
>>> 2345 * 6789
15920205
>>> 12345678987654321 * 987654321
12193263197835695789971041
>>> 'class' + 'room'
'classroom'
>>> 'ha' * 5
'hahahahaha'
>>>
```

## IDLE中的颜色

暗红色：注释。#开头。机器不执行。

橙色：关键字。特定含义的词。

紫色：内置函数等。

绿色：字符串。一对引号（半角）括起来的部分。

蓝色：系统显示结果，或定义的函数。

红色：错误信息。

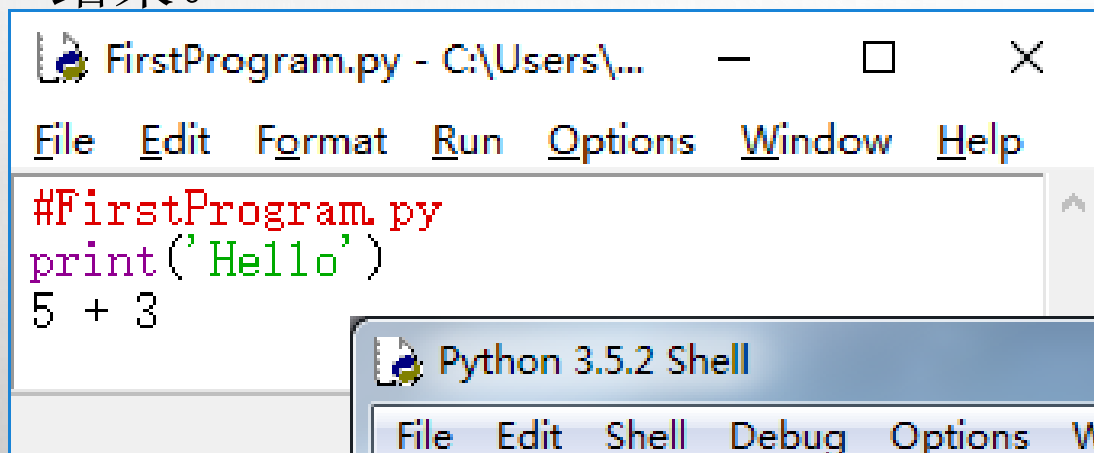
```
>>> print('Hello')
Hello
>>> pront('Hello')
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    pront('Hello')
NameError: name 'pront' is not defined
>>>
```

## 建立程序文件并运行

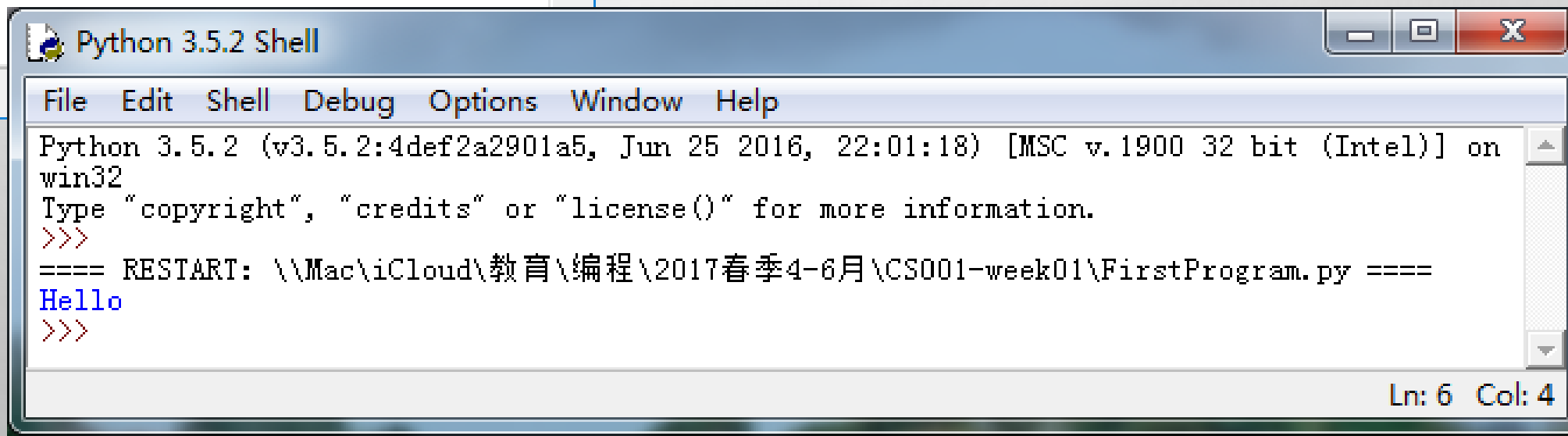
- File→New File: 新建程序文件
- 键入代码
- File→Save或File→Save As: 保存程序文件为FirstProgram.py。  
扩展名.py告诉计算机这是一个Python程序。
- Run→Run Module: 运行程序
- Restart部分表示程序开始运行。

# 程序输出

- 与交互模式下单个指令执行不同，程序文件中只有使用**print**才能看到输出结果。



```
File Edit Format Run Options Window Help
#FirstProgram.py
print('Hello')
5 + 3
```



```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: \\Mac\iCloud\教育\编程\2017春季4-6月\CS001-week01\FirstProgram.py ====
Hello
>>>
```

Ln: 6 Col: 4



## print 函数

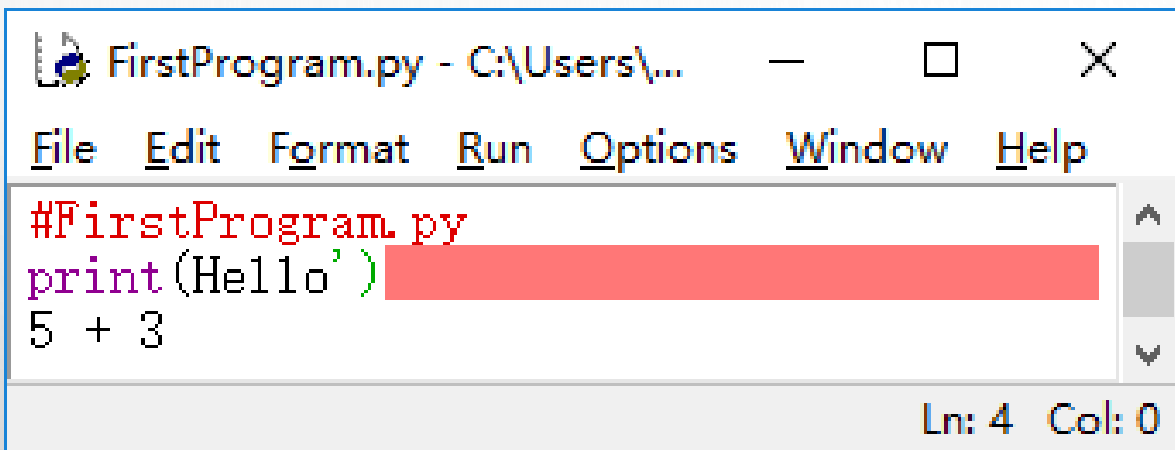
```
>>> print("3+5")
3+5
>>> print(3+5)
8
>>> print("3+5=", 3+5)
3+5= 8
>>> print("3+5=", 3+5, sep='')
3+5=8
>>> |
```

**print**可以用于输出字符串或数值。带引号和不带引号不同。

**print**可以一次输出多项内容，以逗号分隔。输出时多项内容之间自动添加空格。

如果不想要这个空格，可以利用**sep= ''**来设置内容之间的分隔符为空。空格和空不同。

## 程序出错了？



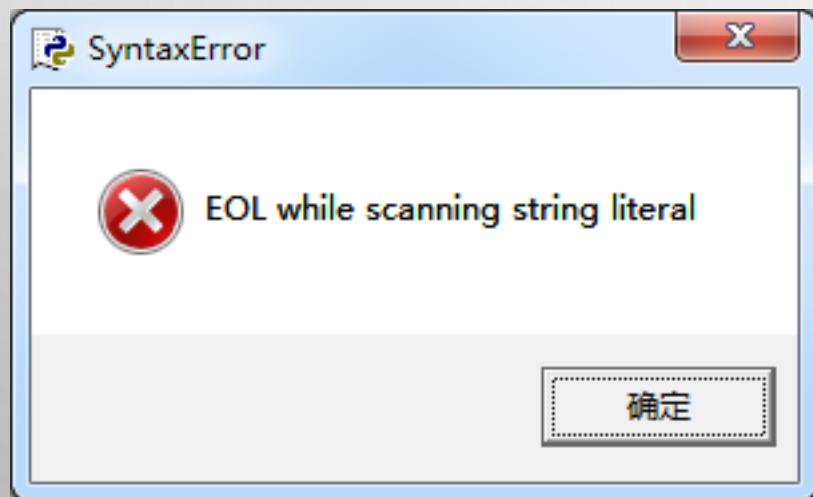
```
#FirstProgram.py
print(Hello')
5 + 3
```

Ln: 4 Col: 0

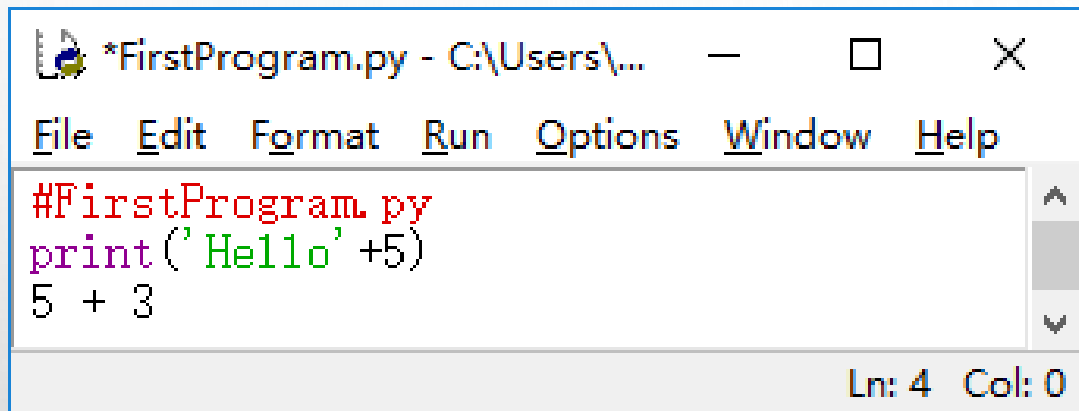
这里的错误是因为漏掉了字符串前面的引号。

IDLE会用红色突出显示它认为出错的地方。

也许问题不会恰好出现在红色显示的位置，不过应该很接近。



## 程序出错了？

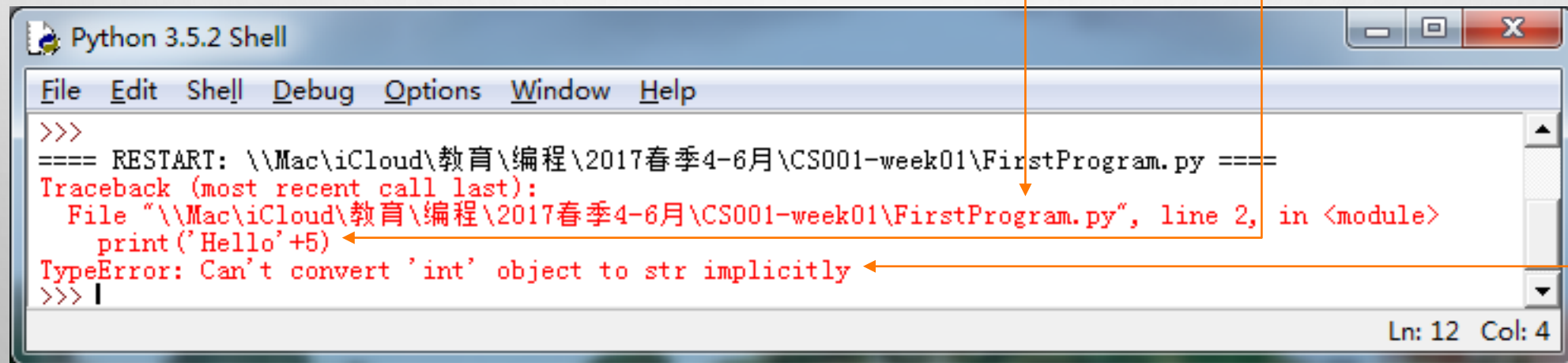


```
*FirstProgram.py - C:\Users\...  
File Edit Format Run Options Window Help  
#FirstProgram.py  
print('Hello'+5)  
5 + 3  
Ln: 4 Col: 0
```

错误发生的位置

出错的代码行

Python认为存在什么问题



```
Python 3.5.2 Shell  
File Edit Shell Debug Options Window Help  
>>>  
==== RESTART: \\Mac\iCloud\教育\编程\2017春季4-6月\CS001-week01\FirstProgram.py ====  
Traceback (most recent call last):  
  File "\\Mac\iCloud\教育\编程\2017春季4-6月\CS001-week01\FirstProgram.py", line 2, in <module>  
    print('Hello'+5)  
TypeError: Can't convert 'int' object to str implicitly  
>>> |  
Ln: 12 Col: 4
```

# Why?

'Hello' + 5不行，而'Hello' \* 5可以。为什么？

就像1光年+1年，不同类型的对象可能无法相加。  
但乘法表示翻倍，总是可以的。

## 程序三要素





# 内存

- 所有的指令和数据必须先进入内存，才能被**CPU**执行和处理。



内存速度快，但成本高，所以容量较小。断电后，内存信息全部丢失。  
外存速度慢，容量大，可以长期保存数据。

## 内存的存储单位

- 计算机内部采用二进制表示
  - 一个比特 (**bit**) 就是二进制数字的一位，有**0**或**1**两种取值可能。
  - 一个字节=**8**个比特 (**1 byte=8bits**)。字节是内存读写的最小单位。
- KILOBYTE (KB):  $2^{10} = 1024$  bytes
- MEGABYTE (MB):  $2^{20} = 1024 \text{ KB OR } 1024 * 1024 \text{ bytes}$
- GIGABYTE (GB):  $2^{30} = 1024 \text{ MB}$
- TERABYTE (TB):  $2^{40} = 1024 \text{ GB}$

目前家用计算机，  
内存大小在**GB**级别，  
硬盘大小在**TB**级别。

# 字符的存储形式

- ASCII编码：一个字节表示一个字符
  - 几个重要字符的ASCII码：
    - 0: ‘ ’ (空)
    - 32: ‘ ’ (空格)
    - 48: ‘0’
    - 65: ‘A’
    - 97: ‘a’

## 查表谁最快？

下面这些ASCII码表示的是什  
么？

72 101 108 108 111

答案：

Hello

ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符
32	(space)	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	\$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	,	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	DEL

请默记刚才表格中的编码几分钟，  
然后抢答！

下列ASCII码表示什么意思？

72 105 32 84 111 109

答案：

Hi Tom

# ASCII编码和字符之间转换

ASCII.py

- ord函数（求字符的编码）
- chr函数（求编码对应的字符）

```
#ASCII.py
print(ord('S')) #输出字符S的ASCII编码，下面类似
print(ord('a'))
print(ord('m'))
print(chr(83)) #输出83作为ASCII码对应的字符，下面类似
print(chr(97))
print(chr(109))
```

```
83
97
109
S
a
m
>>> |
```



# PYTHON支持二、八、十、十六进制的书写

二进制

- **0B**开头，例如**0B010**
- 每位数字只可能是**0**或**1**

十进制

- 按数学格式正常书写

# 进制转换



- bin函数（整数转二进制）
- int函数（转十进制整数）

```
#base.py  
print(bin(10))  
print(bin(1024))  
print(int('0b1010', 2))  
print(int('0b100000000000', 2))
```

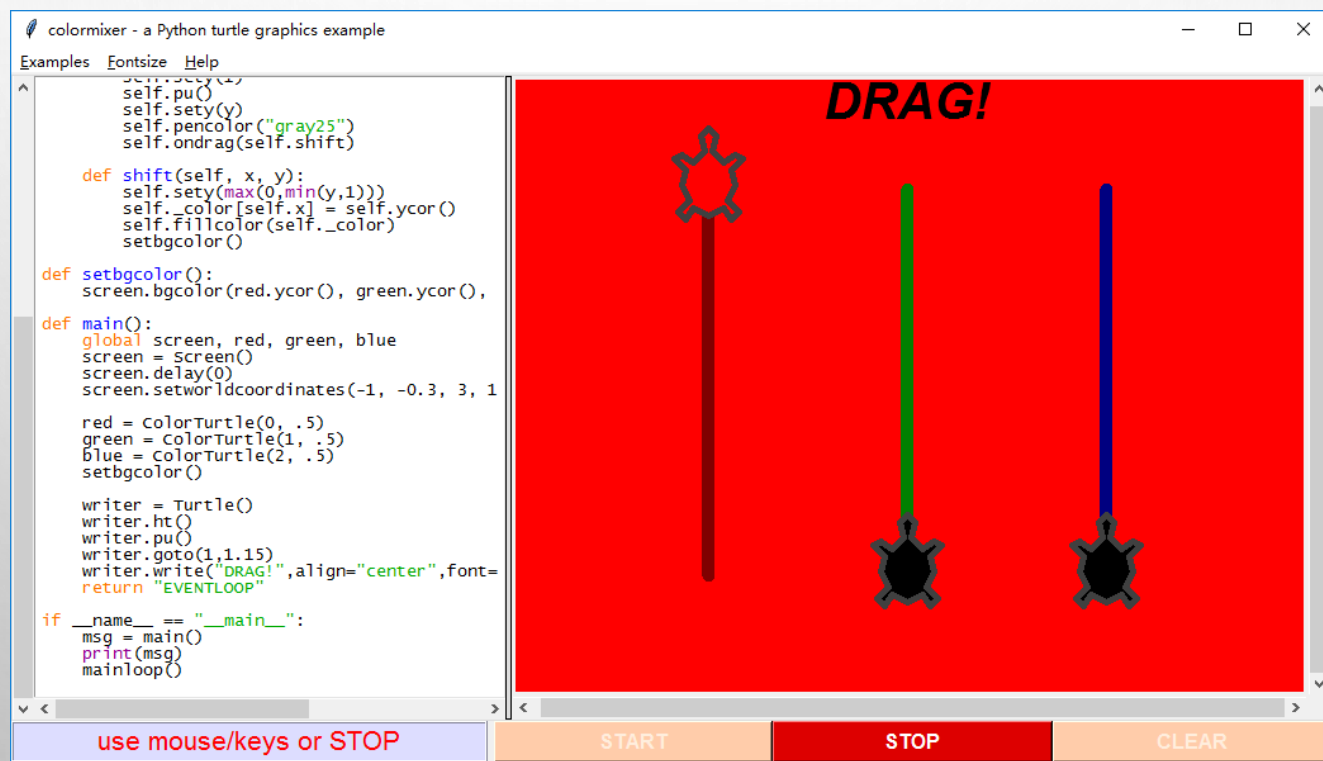
#输出整数10的二进制表示

#输出二进制数字0b1010对应十进制表示

```
0b1010  
0b100000000000  
10  
1024  
>>>
```

# turtle绘图

- Shell菜单: Help-> Turtle Demo



# 绘制几何形状

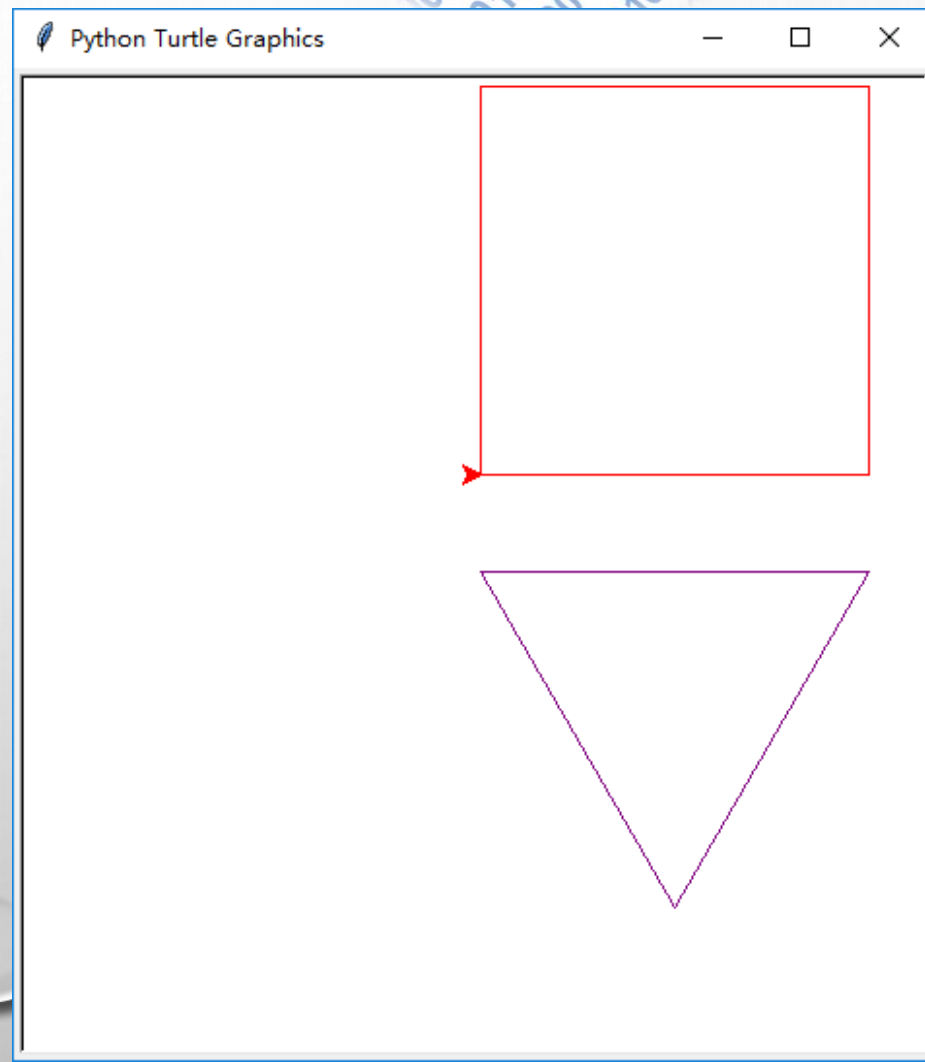
draw1.py

```
#draw1.py
#从turtle导入所有函数
from turtle import *
#定义绘图的速度，数值越大，速度越快
speed(5)
color("purple") #定义绘制时画笔的颜色
#注意：画笔初始位置在画布中央，即坐标原点(0, 0)。方向水平向右。

for i in range(3): #重复三次
    forward(200) #前进200像素
    right(120) #右转120度

up() #抬起画笔
goto(0, 50) #移动至坐标(0, 50)
down() #落笔

color("red")
for i in range(4): #重复四次
    forward(200) #前进200像素
    left(90) #左转90度
```

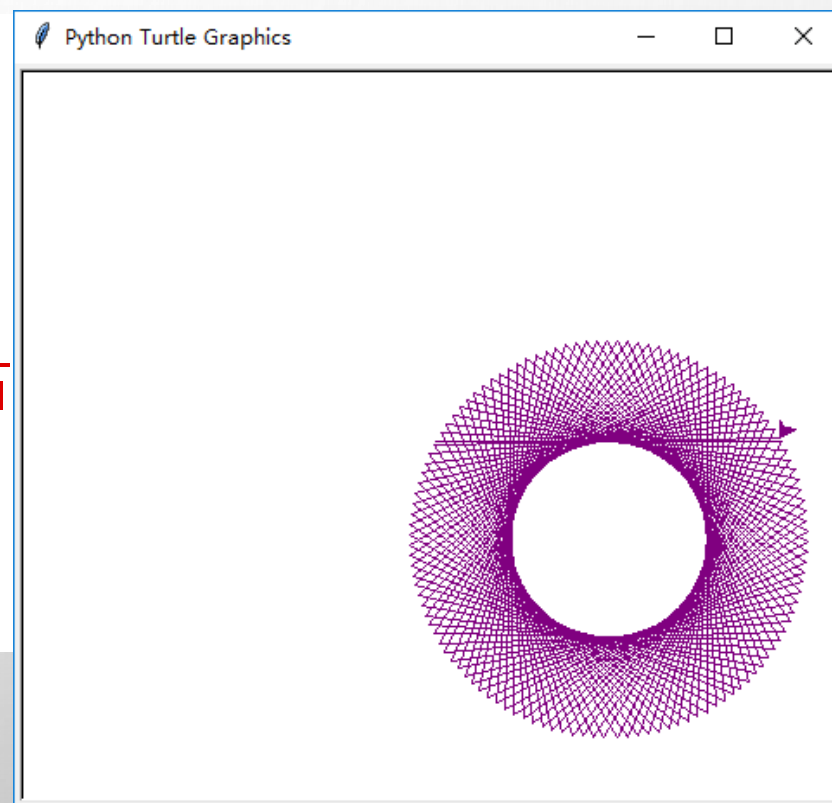


# 绘制几何形状

draw2.py

```
#draw2.py
#从turtle导入所有函数
from turtle import *
#定义绘图的速度，数值越大，速度越快
speed(10)
color("purple") #定义绘制时画笔的颜色
#注意：画笔初始位置在画布中央，即坐标原点(0,0)。方向水平向右

for i in range(120): #重复120次
    forward(200) #前进200像素
    right(121) #右转121度
```





# 复习阅读

- 课本第1章，第2章的2.1

