



# 计算机

## 计算机是根据指令操作数据的设备

### - 功能性

对数据的操作，表现为数据计算、输入输出处理和结果存储等

### - 可编程性

根据一系列指令自动地、可预测地、准确地完成操作者的意图

### - 计算机特点

运算速度快

计算精确度高

具有记忆和逻辑判断能力

有自动控制能力

# 计算机常用的数制

■ 二进制，十进制，八进制，十六进制

二进制	十进制	八进制	十六进制	二进制	十进制	八进制	十六进制
0000	0	0	0	1000	8	10	8
0001	1	1	1	1001	9	11	9
0010	2	2	2	1010	10	12	A
0011	3	3	3	1011	11	13	B
0100	4	4	4	1100	12	14	C
0101	5	5	5	1101	13	15	D
0110	6	6	6	1110	14	16	E
0111	7	7	7	1111	15	17	F

# 文本编码---ASCII码

- 字符编码(Character Code)是用二进制编码来表示字母、数字以及专门符号
- 普遍采用是ASCII(American Standard Code for Information Interchange)码, 例如:

ASCII值	字符	ASCII值	字符	ASCII值	字符
32	(space)	64	@	96	,
33	!	65	A	97	a
34	"	66	B	98	b



# 编码的要素

■ 码字:

a

■ 编码:

97

■ 位和字节表示:

01100001

$$1*2^6+1*2^5+1*2^0=64+32+1=97$$



# Unicode编码

- **Unicode**码是计算机科学领域里的一项业界标准，包括字符集、编码方案等。
- **Unicode**是为了解决传统的字符编码方案的局限而产生的，它为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求。
- ‘\u4e16\u754c\u60a8\u597d’ 代表汉字  
世界你好，  
“\u”表示Unicode码。



# UTF-8编码

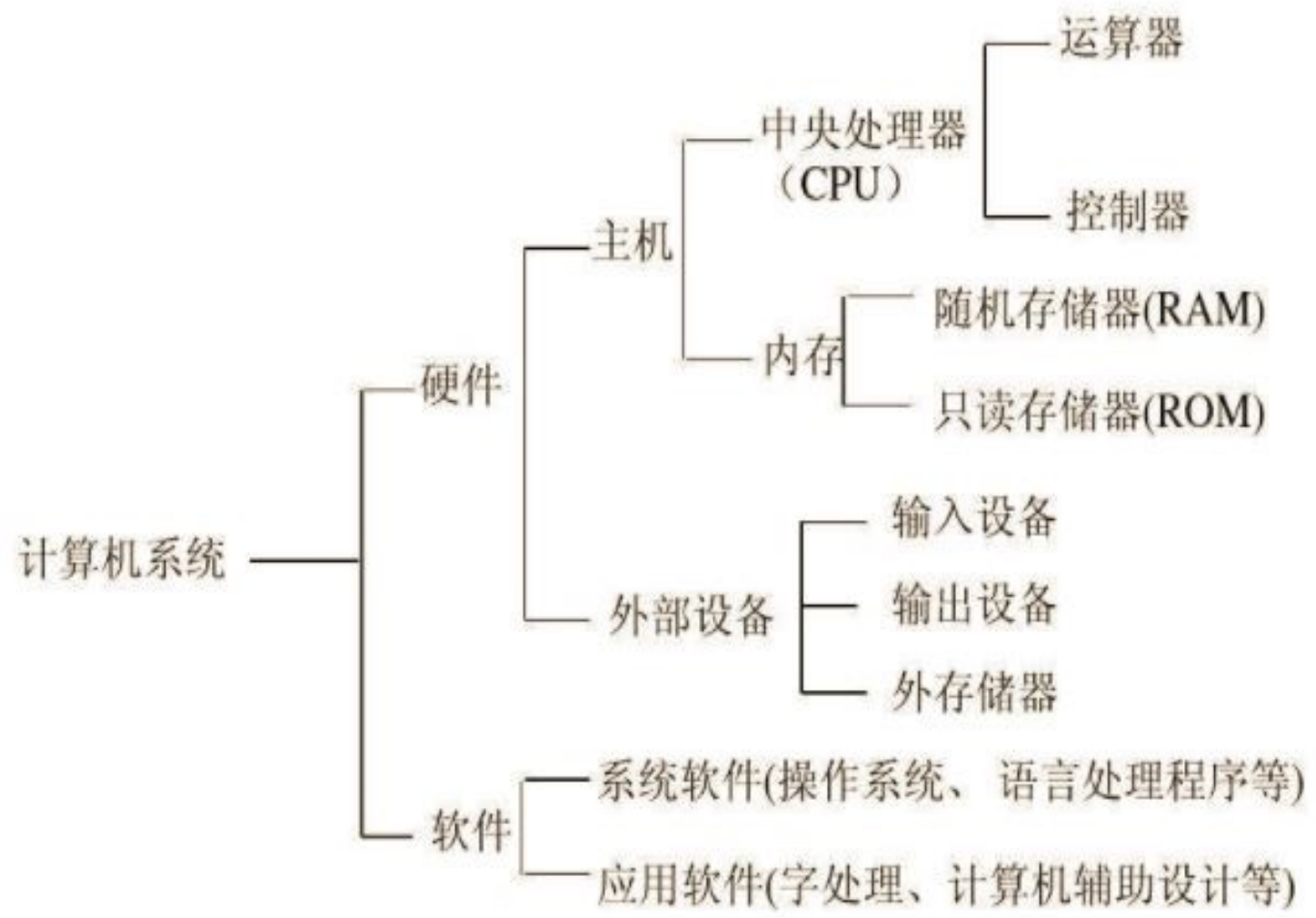
- **Unicode**码并不是存储器中的编码，使用时是把**Unicode**编码转换字节或位，**UTF-8**编码就起这个作用。

每一个**ASCII**字符都有一个**UTF-8**编码，  
该**UTF-8**编码正好与8位的**ASCII**编码一样

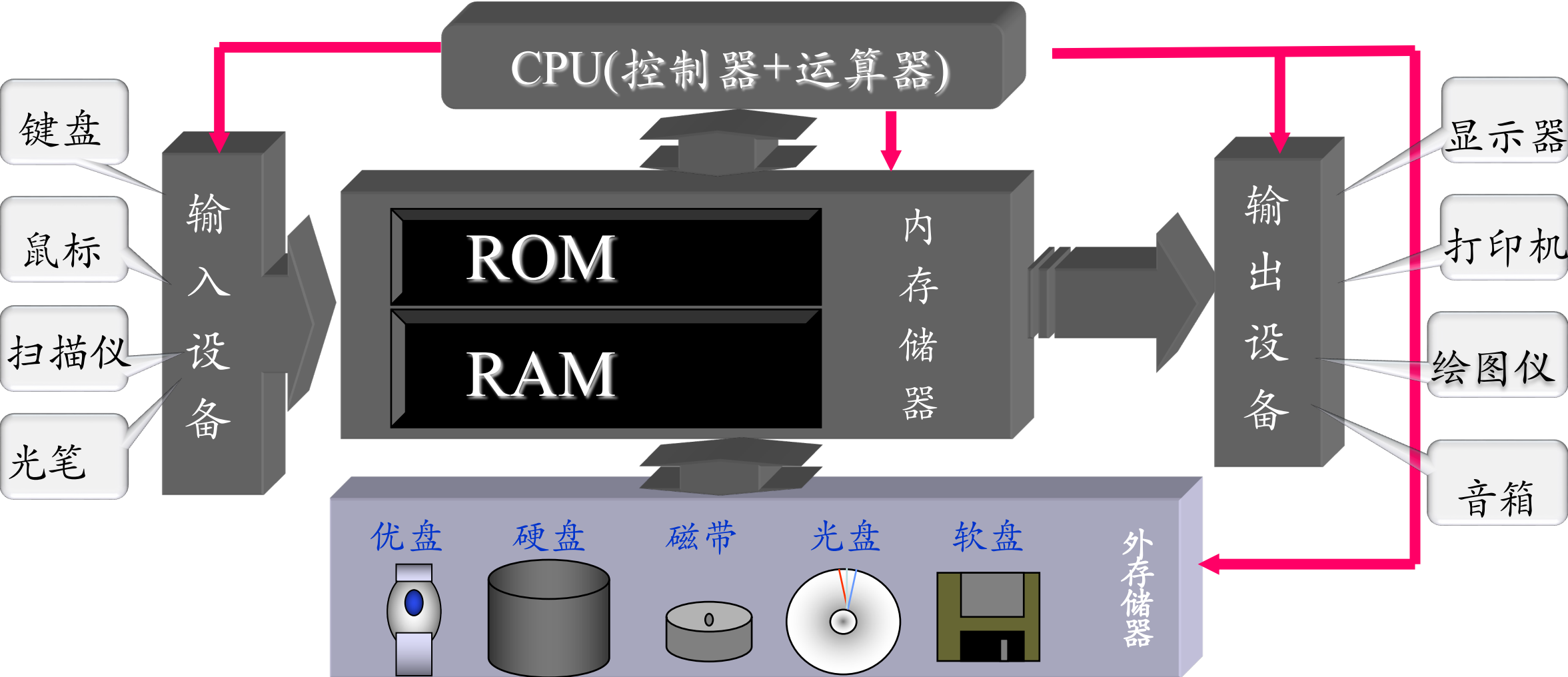
编写**Python3**程序，默认的是**UTF-8**编码



# 计算机系统组成图



# 现代计算机—硬件构架







# 存储单位

单位		实际字节数	近似表示方法
B(Byte)	字节	1	1
KB(K Byte)	千字节	$2^{10}$	$10^3$
MB(M Byte)	兆字节	$2^{20}$	$10^6$
GB(G Byte)	千兆字节(吉)	$2^{30}$	$10^9$
TB(T Byte)	兆兆字节(太拉)	$2^{40}$	$10^{12}$



# 操作系统

- 操作系统（**Operating System**，简称**OS**）是管理和控制计算机硬件与软件资源的计算机程序，是直接运行在“裸机”上的最基本的系统软件
- **Unix/Linux**
- **Windows**
- **Mac OS**



# 程序设计

**程序设计是计算机可编程性的体现**

- **程序设计，亦称编程，深度应用计算机的主要手段**

## 程序设计语言

**程序设计语言是一种用于交互(交流)的人造语言**

- **程序设计语言，亦称编程语言，程序设计的具体实现方式**
- **编程语言相比自然语言更简单、更严谨、更精确**
- **编程语言主要用于人类和计算机之间的交互**



# 程序设计语言

**编程语言种类很多，但生命力强劲的却不多**

- **编程语言有超过上千种，绝大部分都不再被使用**
- **C语言诞生于1972年，它是第一个被广泛使用的编程语言**
- **Python语言诞生于1990年，它是最流行最好用的编程语言**













# 程序设计语言发展历程

1. 第1代：机器语言——二进制代码指令构成，不同CPU具有不同指令系统
2. 第2代：汇编语言——机器指令的符号化，可直接访问系统接口
3. 第3代：高级语言——面向用户的、基本上独立于计算机硬件的语言
  - C/C++语言
  - Java语言
  - Python语言：简单、易学、实用、使用者众多
4. 第4代：非过程化语言——如数据库查询和应用程序生成器，只需说明“做什么”，不需描述算法细节



# TIOBE编程语言排行榜

Sep 2021	Sep 2020	Change	Programming Language	Ratings	Change
1	1		 C	11.83%	-4.12%
2	3	▲	 Python	11.67%	+1.20%
3	2	▼	 Java	11.12%	-2.37%
4	4		 C++	7.13%	+0.01%
5	5		 C#	5.78%	+1.20%
6	6		 Visual Basic	4.62%	+0.50%
7	7		 JavaScript	2.55%	+0.01%
8	14	▲▲	 Assembly language	2.42%	+1.12%
9	8	▼	 PHP	1.85%	-0.64%
10	10		 SQL	1.80%	+0.04%

# 编程语言的执行方式

## 计算机执行源程序的两种方式：编译和解释

- **源代码：采用某种编程语言编写的计算机程序，人类可读**

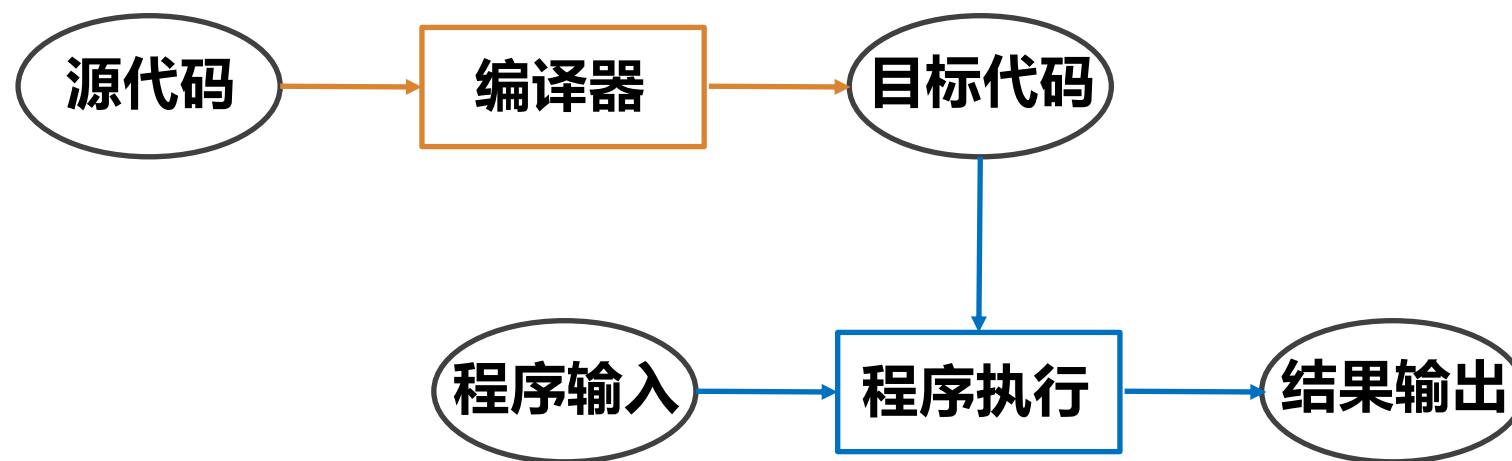
例如：`result = 2 + 3`

- **目标代码：计算机可直接执行，人类不可读（专家除外）**

例如：`11010010 00111011`

# 编译

将源代码一次性转换成目标代码的过程

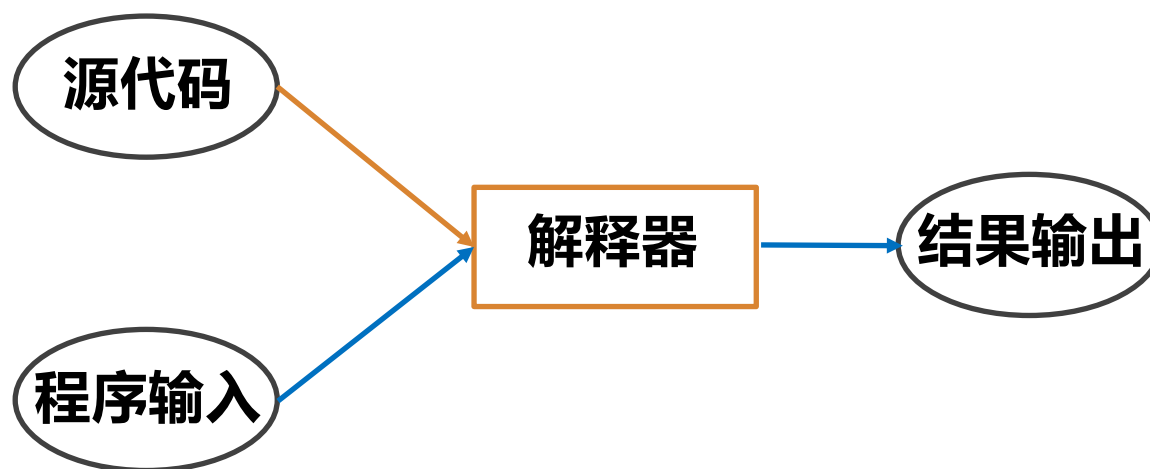


执行编译过程的程序叫作编译器



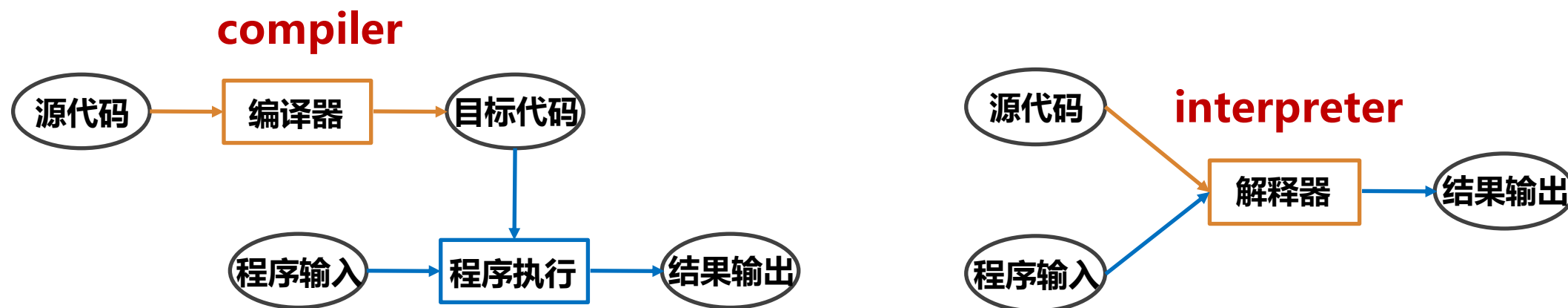
# 解释

**将源代码逐条转换成目标代码同时逐条运行的过程**



**执行解释过程的程序叫作解释器**

# 编译和解释



- 编译：一次性翻译，之后不再需要源代码（类似英文翻译）
- 解释：每次程序运行时随翻译随执行（类似实时的同声传译）



# 静态语言和脚本语言

根据执行方式不同，编程语言分为两类

- **静态语言：使用编译执行的编程语言**

**C/C++语言、Java语言**

- **脚本语言：使用解释执行的编程语言**

**Python语言、JavaScript语言、PHP语言**



# 静态语言和脚本语言

**执行方式不同，优势各有不同**

- **静态语言：编译器一次性生成目标代码，优化更充分  
程序运行速度更快**
- **脚本语言：执行程序时需要源代码，维护更灵活  
源代码在维护灵活、跨多个操作系统平台**



# 什么是Python?

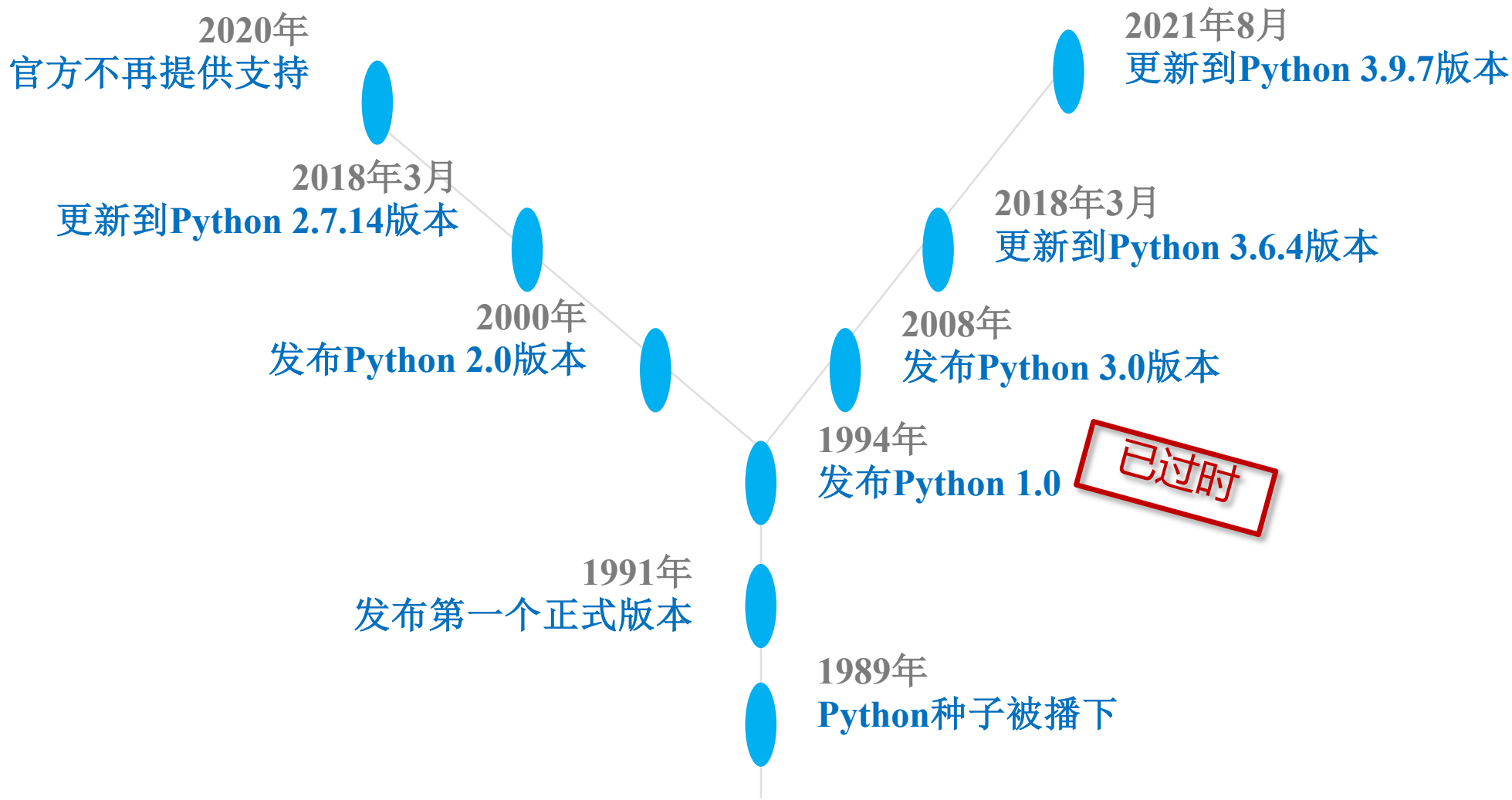
# Python

- Python是一种**面向对象**的**解释型**计算机程序设计语言，由荷兰人Guido van Rossum于1989年发明，第一个公开发行人版发行于1991年。
- Python的设计哲学是“优雅”、“明确”、“简单”
- Python是自由软件之一，免费、开源。
- Python已经被移植到许多平台上。这些平台包括 Unix/Linux、Windows、Mac OS。





- Python是一门跨平台、开源、免费的解释型高级动态编程语言。
- Python支持命令式编程（How to do）、函数式编程（What to do），完全支持面向对象程序设计，拥有大量扩展库。
- 胶水语言：可以把多种不同语言编写的程序融合到一起实现无缝拼接，更好地发挥不同语言和工具的优势，满足不同应用领域的需求。







应该选择哪个版本呢？



## 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**。
- 在选择Python版本的时候，一定要先考虑清楚自己学习Python的目的是什么，打算做哪方面的开发，该领域或方向有哪些扩展库可用，这些扩展库最高支持哪个版本的Python。这些问题全部确定以后，再最终确定选择哪个版本。
- Python 2.x系列已于2020年**全面放弃维护和更新**。
- **本课程使用Python3.x版本**



# Python的两种编程方式

## 交互式和文件式

- **交互式：对每个输入语句即时运行结果，适合语法练习**
- **文件式：批量执行一组语句并运行结果，编程的主要方式**



## 1. 使用交互式解释器

```
>>> import sys    #导入系统模块sys
```

```
>>> sys.version
```

又如：输出Python3.x的默认编码。

```
>>> sys.getdefaultencoding()  
'utf-8'
```

```
>>> import sys  
>>> sys.version  
'3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)]'  
>>> sys.getdefaultencoding()  
'utf-8'
```

A geometric diagram consisting of a central point connected to several vertices on a horizontal line, forming a series of triangles. This diagram is mirrored across the horizontal line.

# 程序的基本编写方法



# IPO

## 程序的基本编写方法

- **I: Input 输入，程序的输入**
- **P: Process 处理，程序的主要逻辑**
- **O: Output 输出，程序的输出**



# 理解IPO

## 输入——Input

- 程序的输入

文件输入、网络输入、控制台输入、交互界面输入、内部参数输入等

- 输入是一个程序的开始



# 理解IPO

## 输出——Output

- **程序的输出**

控制台输出、图形输出、文件输出、网络输出、操作系统内部变量输出等

- **输出是程序展示运算结果的方式**





# 理解IPO

## 处理——Process

- 处理是程序对输入数据进行计算产生输出结果的过程
- 处理方法统称为算法，它是程序最重要的部分
- 算法是一个程序的灵魂



# 问题的计算部分

**一个待解决问题中，可以用程序辅助完成的部分**

- 计算机只能解决计算问题，即问题的计算部分**
- 一个问题可能有多种角度理解，产生不同的计算部分**
- 问题的计算部分一般都有输入、处理和输出过程**



# 编程解决问题的步骤

## 6个步骤 (1-3)

- **分析问题：** 分析问题的计算部分，**想清楚**
- **划分边界：** 划分问题的功能边界，**规划IPO**
- **设计算法：** 设计问题的求解算法，**关注算法**



# 使用计算机解决问题

## 6个步骤 (4-6)

- **编写程序：** 编写问题的计算程序， **编程序**
- **调试测试：** 调试程序使正确运行， **运行调试**
- **升级维护：** 适应问题的升级维护， **更新完善**



# 求解计算问题的精简步骤

## 3个精简步骤

- **确定IPO：明确计算部分及功能边界**
- **编写程序：将计算求解的设计变成现实**
- **调试程序：确保程序按照正确逻辑能够正确运行**

## Python编程模式（命令式编程）

- 问题解决：把列表中的所有数字都加5，得到新列表。（命令式编程）

```
>>> x = list(range(10))
```

创建列表

```
>>> x
```

空列表

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> y = []
```

```
>>> for num in x:
```

循环，遍历x中的每个元素

```
    y.append(num+5)
```

列表方法，在尾部追加元素

```
>>> y
```

```
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

```
>>> [num+5 for num in x]
```

列表推导式

```
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

## Python编程模式（函数式编程）

- 问题解决：把列表中的所有数字都加5，得到新列表。（函数式编程）

```
>>> x = list(range(10))
```

```
>>> x
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> def add5(num):  
    return num+5
```

定义函数，接收一个数字  
，加5后返回

```
>>> list(map(add5, x))
```

```
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

把函数add5映射到x中的每个元素

```
>>> list(map(lambda num: num+5, x))
```

```
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

lambda表达式，等价于函数add5



# Python基本语法

- Python程序的基本对象是常量，变量、函数和关键字等，然后组合成为列表、元组、集合和字典等组合数据，进而形成表达式、语句和语句块，最后组成完整程序。即：
- 常量、变量、函数和关键字等 → 列表、元组、集合和字典等 → 表达式 → 语句 → 语句块 → 程序。





## 05 标识符和变量

- 标识符（名称）：用于标识变量、函数、列表、元组、集合、字典和对象等名称的符号。
- 命名规则：
  - (1) 首字符必须是汉字、字母表中字母或下划线'\_'。
  - (2) 其余部分可以是汉字、字母、数字和下划线。
  - (3) 区分大小写
  - (4) 不能使用关键字。
  - (5) 前后不能均为下划线。因为前后均为下划线的标识符通常为系统变量。
- 正确的标识符：学号、姓名、Age、age、\_var、book\_name等。
- 错误的标识符：6var、性 别、bir th、var^、sno#、\$press等。

## 关键字

- 关键字（保留字）：Python中具有特定含义的名称，而且区分大小写。不难看出，关键字不能再作它用。
- Python标准库中提供的keyword模块，可以输出当前版本的所有关键字。即：
- `>>> import keyword`
- `>>> print(keyword.kwlist)`
- `['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']`
- 提示：Python中的符号区分大小写！



## 常量和变量

- 常量就是不能改变的量，比如常用的数学常数3.14159就是一个常量
- 变量就是程序为了方便地引用内存中的值而为其取的名称。
- Python变量名是大小写敏感的
- `>>>a=7`      “=” 是赋值号
- `>>>a`
- `7`
- 7是一个对象，可以通过变量a引用这个对象



## id函数

- Python变量有一个非常重要的性质：变量是将名字和对象关联起来。赋值操作并不会实际复制值，它只是为数据对象取个相关的名字。名字是对象的引用而不是对象本身
- id是Python的内置函数，显示对象的地址

# id函数用法

```
>>> id(a)
```

```
1566532000
```

```
>>> id(7)
```

```
1566532000
```



1566532000

```
>>> a=5
```

```
>>>a
```

```
5
```

```
>>>id(a)
```

```
1566531936
```

```
>>>id(5)
```

```
1566531936
```

```
>>> id(7)
```

```
1566532000
```



15665320000



1566531936



## 06 输入及输出函数

- 输入函数: `input()`
- `input`从键盘输入一个字符串。 ‘9’ 表示是一个字符串, 它的ASCII码值是57
- ```
>>>a=input()  
9  
  
>>>a  
'9'
```



## 输入数字

- 用 `int()` 函数输入数字

- `>>>a=int(input())`

9

`>>>a`

9

## 一行输入多个值

```
>>>m, n=input("请输入多个值: ").split()
```

请输入多个值: 3 5

```
>>>m
```

```
'3'
```

```
>>>n
```

```
'5'
```

**input(“请输入多个值: ”)函数中的参数是  
输出提示字符串**



## 输出函数: print()

- print是输出函数，参数是输出值

- `>>> print(3)`     #输出1个数字

- 3

- `>>> print(3, 7)`   #输出2个数字

- 3 7

- `>>> b, c=3, 4`       #输出1个数字，两个变量

- `print(b, c, 5)`

- 3 4 5

- 井号“#”常被用作单行注释符号，在代码中使用“#”时，它右边的任何数据都会被忽略，当做是注释



## 不换行输出

- 每行输出一个值

```
print(3)
```

```
print(4)
```

```
print(5)
```

- 由于print()函数默认是以换行符结束的，所以在默认的情况下会自动换行。

- 用end参数，一行输出三个值，

```
print(3, end=' ')
```

```
print(4, end=' ')
```

```
print(5, end=' ')
```