

# Python 零基础轻松入门

老齐

# 1 为什么要学编程

- 是不是只有程序员才学编程？
- 学习编程的目的：
  - 当一名程序员
  - 优化自己的工作
  - 个人兴趣
  - 预防老年痴呆

## 2 编程语言简介

- 机器语言

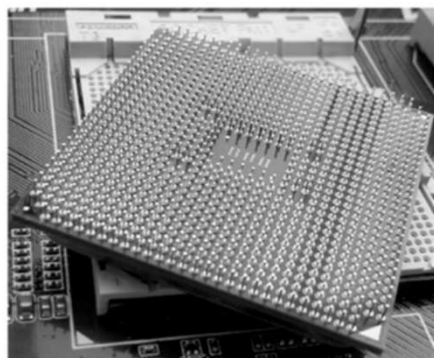


图 1-2-1 CPU 的引脚

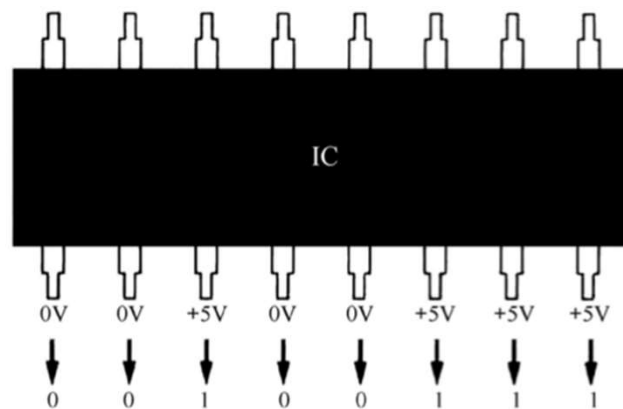


图 1-2-2 IC 的引脚和二进制

```
01001000 01100101 01101100 01101100 01101111 00100000 01010111 01101111 01110010
01101100 01100100
```

如果写成了下面这样，打印的就不是“Hello World”了。

```
01001000 01100101 01101100 01101100 01101111 00110000 01010111 01101111 01110010
01101100 01100100
```


## 2 编程语言简介

- 汇编语言

```
-----  
; Writes "Hello, World" to the console using only system calls. Runs on 64-bit Linux only.  
; To assemble and run:  
;  
; nasm -felf64 hello.asm && ld hello.o && ./a.out  
-----
```

```
global _start  
  
section .text  
_start: mov rax, 1      ; system call for write  
        mov rdi, 1      ; file handle 1 is stdout  
        mov rsi, message ; address of string to output  
        mov rdx, 13      ; number of bytes  
        syscall          ; invoke operating system to do the write  
        mov rax, 60      ; system call for exit  
        xor rdi, rdi      ; exit code 0  
        syscall          ; invoke operating system to exit  
  
section .data  
message: db "Hello, World", 10 ; note the newline at the end
```

这里是对左边指令的说明  
这是给人看的

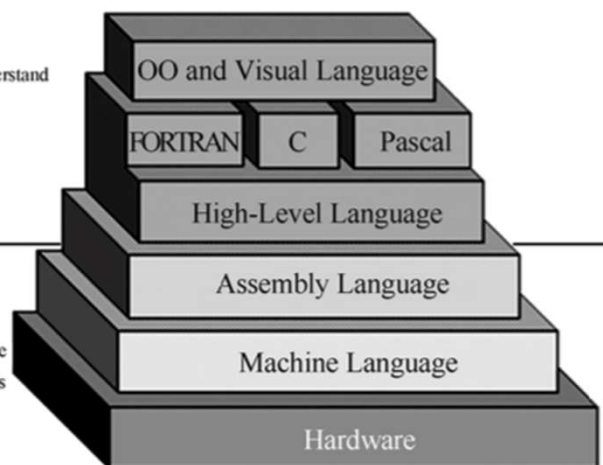


## 2 编程语言简介

### • 高级语言

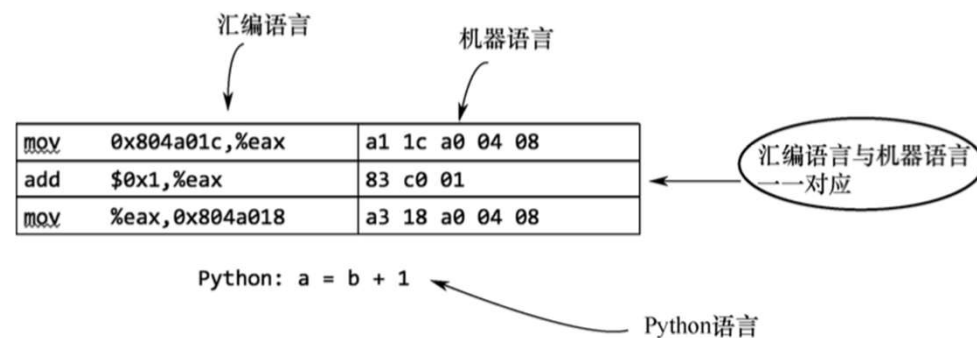
#### High level Language

- Easy for Programmers to understand
- Contains English Words



#### Low level Language

- The computer's own Language
- Binary numbers, in 1's and 0's



## 2 编程语言简介

### • 高级语言发展

- ❖ 1950 年以前，是编程语言的“史前”年代。虽然已经有了用“打孔卡”方式编程（见图 1-1-1）的记载，但并没有被广泛采用。

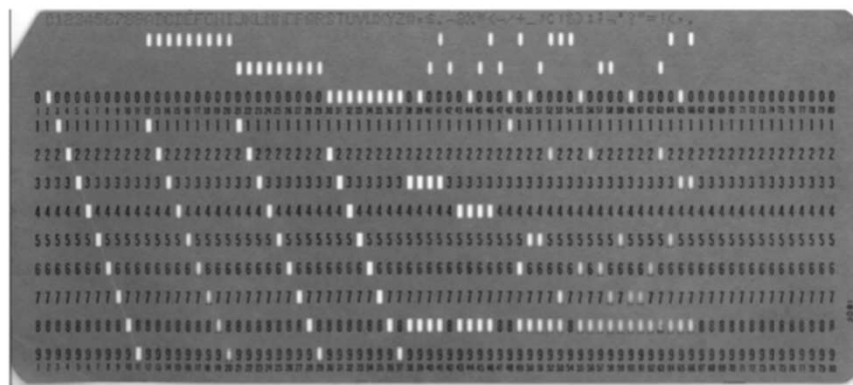


图 1-1-1 80 列、矩形孔的标准 IBM 打孔卡片（源自《维基百科》网站）

- ❖ 1957 年，Fortran 诞生，它是世界上第一个被正式采用并流传至今的高级编程语言。发明者是 John Warner Backus，此处应当献上敬意和崇拜（以下列出的各项语言发明者，亦或该语言发明团队的负责人、主要设计者，为了简便，统一称为“发明者”，并且都要献上敬意和崇拜）。
- ❖ 1958 年，LISP 诞生。发明者 John McCarthy。
- ❖ 1964 年，BASIC 诞生。发明者 John G.Kemeny 和 Thomas E.Kurtz。
- ❖ 1970 年，Pascal 诞生。发明者 Niklaus Emil Wirth。此外，他还是 Algol W、Modula、Oberon、Euler 等语言的发明者。
- ❖ 1972 年，C 诞生。发明者 Dennis Ritchie 和 Ken Thompson。

## 2 编程语言简介

### • 高级语言发展

- ❖ 1983 年，C++诞生。发明者 Bjarne Stroustrup。
- ❖ 1986 年，Objective-C 诞生。发明者 Tom Love 和 Brad Cox。
- ❖ 1987 年，Perl 诞生。发明者 Larry Wall。
- ❖ 1991 年，本书的主角 Python 诞生。发明者 Guido van Rossum。有打油诗如此赞：Python 诞生，天降大任，开源开放，简洁优雅，独步天下，人工智能，“唯我不败”。请牢记这个值得纪念的年份和“仁慈的独裁者”（BDFL）。
- ❖ 1993 年，Ruby 诞生。发明者松本行弘。
- ❖ 1995 年，Java 诞生。发明者 James Gosling。
- ❖ 1995 年，JavaScript 诞生。发明者 Brendan Eich。请注意，JavaScript 与 Java 在名字上和语法上虽然相似，但它们是两种完全不同的编程语言。
- ❖ 1995 年，PHP 诞生。发明者 Rasmus Lerdorf。
- ❖ 2001 年，C#诞生。发明者 Microsoft 公司。
- ❖ 2009 年，Go 诞生。发明者 Robert Griesemer、Rob Pike、Ken Thompson。
- ❖ 2011 年，Rust 诞生。发明者 Graydon Hoare。

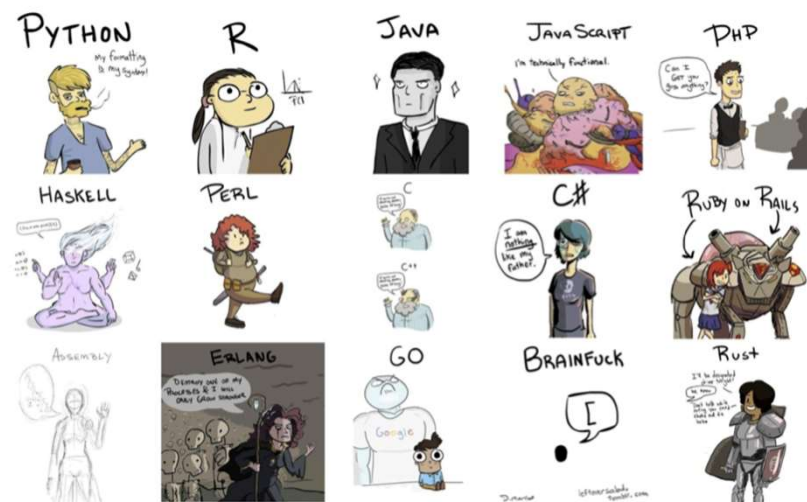


图 1-1-2 如果编程语言是人

### 3 学哪一种编程语言

- 入门的编程语言，应该具备如下特点：
  - 简单易学
  - 用途相对广泛

**TIOBE Index for August 2019**

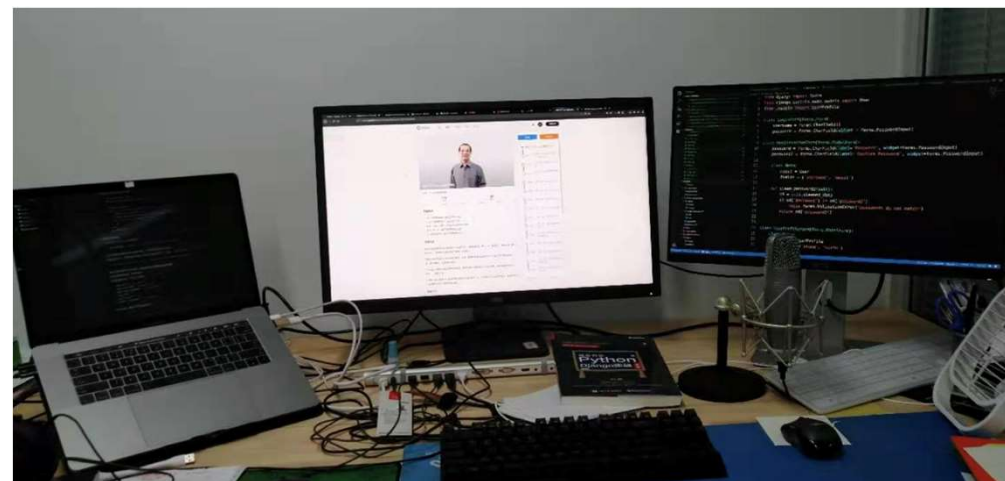
Aug 2019	Aug 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.028%	-0.85%
2	2		C	15.154%	+0.19%
3	4	▲	Python	10.020%	+3.03%
4	3	▼	C++	6.057%	-1.41%
5	6	▲	C#	3.842%	+0.30%
6	5	▼	Visual Basic .NET	3.695%	-1.07%
7	8	▲	JavaScript	2.258%	-0.15%
8	7	▼	PHP	2.075%	-0.85%
9	14	▲▲	Objective-C	1.690%	+0.33%
10	9	▼	SQL	1.625%	-0.69%



## 4 学习方法

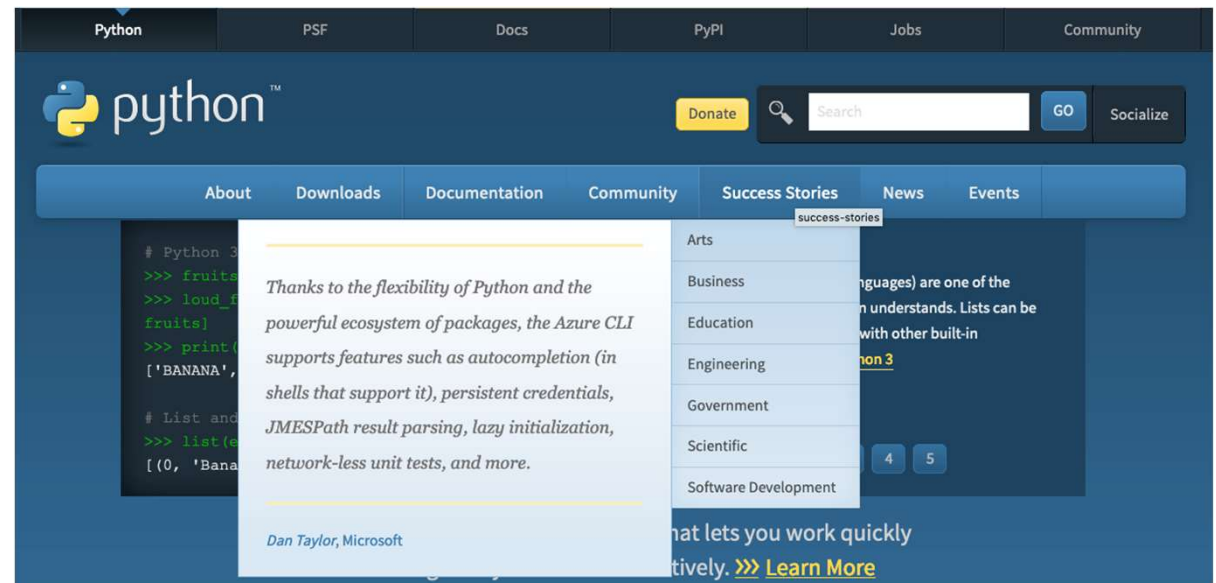
- 丢掉幻想，准备战斗
  - 时间：
    - 每天都要保证学习时间
    - 不要奢望所谓碎片时间学习
  - 思想：
    - 一定要改变“等着别人伸把手”的习惯，自动动手搜索
  - **工欲善其事，必先利其器**

凌乱而实用的办公桌



# 5 Python概要

- 发明人：吉多·范 罗苏姆(Guido van Rossum)
- 官方网站：python.org
- 目前的应用领域：
  - Web 开发
  - 网络爬虫
  - 数据分析和机器学习
  - 神经网络
- 语言特点
  - 开源
  - 简单
  - 不断进化



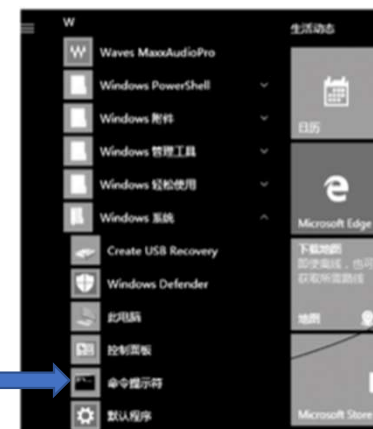
## 6 Python开发环境配置

- 熟悉自己的计算机操作系统
  - macOS
  - Linux: Ubuntu
  - Windows: 32位、64位
- Python是跨平台的
- 访问Python官网，下载安装程序。
  - Ubuntu系统，不需要安装，系统自带；如果要更改版本，可以从新下载安装
  - 其他类型系统，根据系统下载安装程序

# 6 Python开发环境配置

- Windows用户注意了
  - 要做好环境变量，才能在任何位置启动Python

命令提示符



## 6 Python开发环境配置

- Windows用户注意了
  - 在cmd中进入Python交互界面

命令提示符



```
命令提示符 - python
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation. 保留所有权利。

C:\Users\dell>python
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

# Python 零基础轻松入门

老齐

# 1 内置对象类型

- 简要理解对象
- 内置对象类型
  - 整数、浮点数
  - 字符串
  - 列表
  - 元组
  - 字典
  - 集合
- 变量与对象

## 2 整数和浮点数

- 整数类型: `int`
- 浮点数类型: `float`
- 四则运算
- 标准库中的`math`
- 计算中的“异常”



# 3 字符和字符串

- 字符
- 字符编码
- 字符串
  - 定义
  - 序列及其基本操作
  - 索引和切片
- 内置函数
  - print
  - input

## 3 字符和字符串

- 字符串的常用属性和方法
  - 查看对象的属性和方法
  - 使用帮助文档
- 字符串格式化输出

## 4 列表

- 列表基本知识
  - 定义
  - 元素特点
  - 索引和切片
  - 基本操作
- 列表的方法
- 比较列表与字符串
  - 都是序列
  - 列表是容器类对象，列表可变
  - 字符串不可变

# 5 元组

- 元组基本知识
  - 定义
  - 只有一个元素的元组
  - 索引和切片
  - 基本操作
- 比较元组和列表
  - 元组不可变
  - 元组运算速度快
  - 两者都是容器类、序列类对象

# 6 字典

- 字典基本知识
  - 定义
  - 基本操作
- 字典的方法
- 比较字典和列表
  - 字典不是序列
  - 两者都是容器类对象
  - 两者都是可变对象
  - Python3.6开始，字典也有顺序

# 7 集合

- 集合基本知识
  - 可变集合的定义
  - 不可变集合的定义
  - 集合的特点
- 可变集合的方法
- 集合的关系和运算

## 8 案例

- 案例1：编写程序，根据输入的半径，计算圆的面积。
- 案例2：编写程序，利用“凯撒密码”方案，实现对用户输入文字的加密操作。
- 案例3：编写程序，实现对输入字符串的大小写字母翻转（即大写变小写、小写变大写）操作。

## 9 作业

- 作业1：已知列表[1, 2, 3, 4, 5, 6, 7, 8, 9]，在交互模式中分别实现如下效果：
  - 得到[2, 4, 6, 8]
  - 得到[9, 7, 5, 4, 3, 1]
  - 得到[1, 2, 3, 4]
  - 得到[4, 3, 2, 1]

( 作业来源： 《Python大学实用教程》 86页第24题)



## 9 作业

- 作业2：编写程序，实现如下功能：
  - 用户输入国家名称；
  - 打印出所输入的国家名称和首都。

( 作业来源： 《Python大学实用教程》 87页第32题)

## 9 作业

- 作业3：编写程序，实现用户输入数字，显示对应的英文，如输入“250”，显示“two five zero”。

（作业来源：《Python大学实用教程》87页第34题）

# Python零基础轻松入门

老齐

# 布尔类型

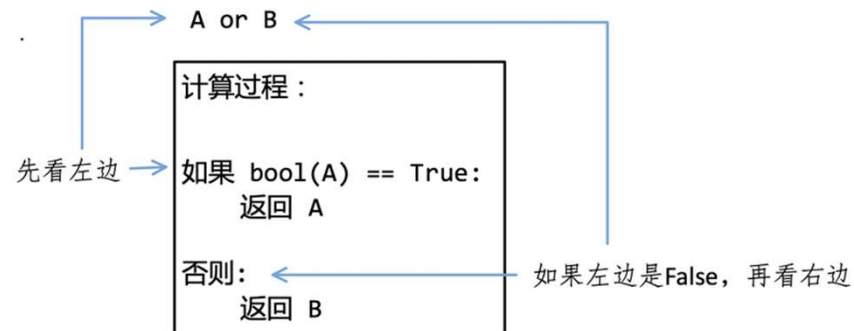
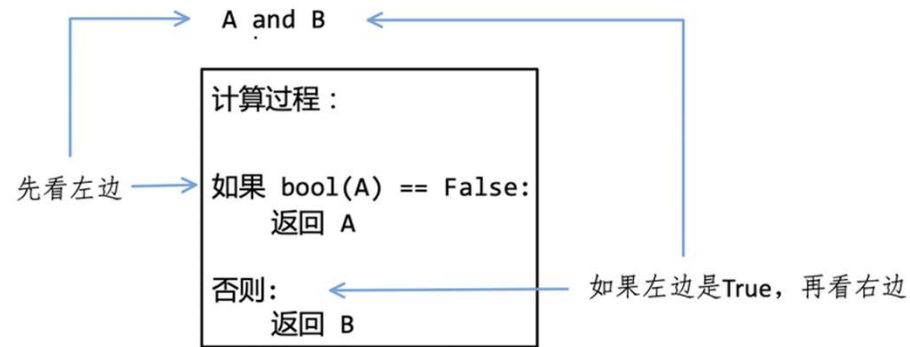
- True和False的类型: `bool`
  - True、False之间的四则运算: True: 1, False: 0
- 内置函数: `bool()`
  - `bool()`、`bool("")`、`bool(' ')`、`bool(0)`
  - `bool(False)`、`bool(True)`

# 比较运算

- 比较两个对象：
  - 符号：>, <, ==, !=, >=, <=
  - 同种类型对象比较
  - 返回值：True和False

# 逻辑运算（布尔运算）

- and
- or
- not



# 简单的语句

- import语句
  - Import module
  - Import module as new\_name
  - From module import function
  - From module import function as new\_name
  - From module import \*

# 赋值语句

- 基本形式: `variable = object`
- 其他花样:
  - `a = 1, 2, 3`
  - `a, b, c = 1, 2, 3`
  - `a, _, c = 1, 2, 3`
  - `a, *b = 1, 2, 3`
  - `a = b = 1`



# 自增运算

- $a = a + 1 \rightarrow a += 1$
- 其他形式
  - $a -= 2$
  - $a *= 2$
  - $a /= 2$

# 条件语句



# 条件语句

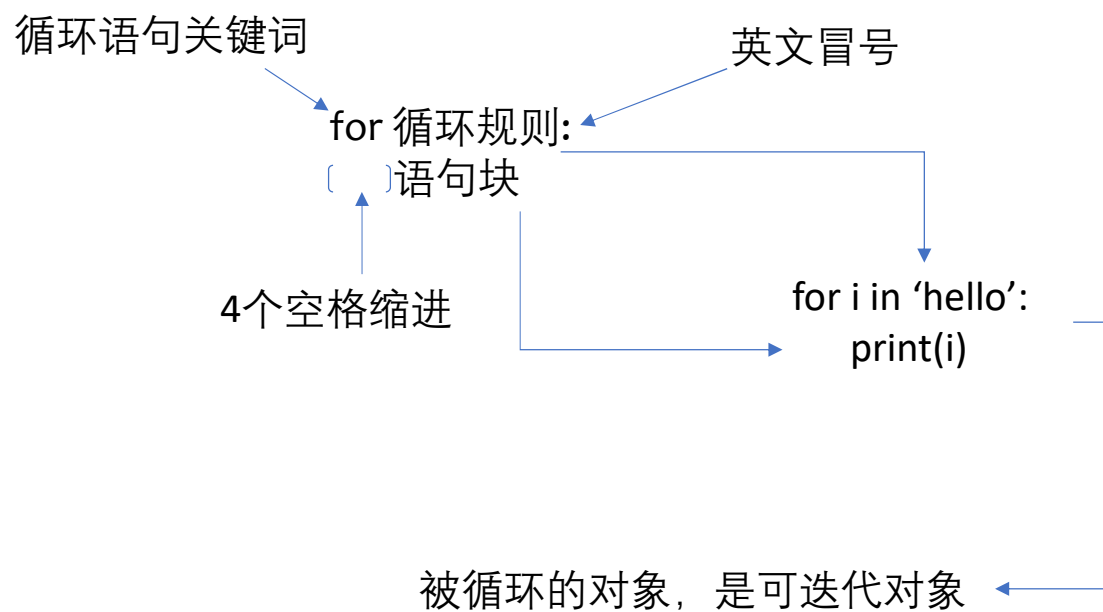
```
if (<expr>):  
    <statement>  
    <statement>  
elif (<expr>):  
    <statement>  
elif (<expr>):  
    <statement>  
else:  
    <statement>  
<following_statement>
```

```
a = 5  
if a > 3:  
    a += 1  
elif a == 3:  
    a -= 1  
else:  
    a = abs(a)  
print(a)
```

# 例题

- 编写程序，随机生成10以内的一个整数，如果该数字大于圆周率 $\pi$ ，就将其当做直径计算圆的周长和面积；否则当做半径计算圆的周长和面积。最后将计算结果输出。

# for循环语句



# for循环语句

- 例题：创建一个数据集，包含1到10的随机整数，共计100个数。并统计每个数字的次数。

# for循环语句

- 几个常用函数：
  - range()
  - zip()
  - enumerate()
- 列表解析

# for循环语句

- 例题1：创建一个列表，其中的元素是100以内的能被3整除的正整数。
- 例题2：字符串s = 'Life is short You need python'。统计这个字符串中每个单词的字母数量。



# while循环语句

- 基本格式:

```
while [condition]:  
    statements
```

# while循环语句

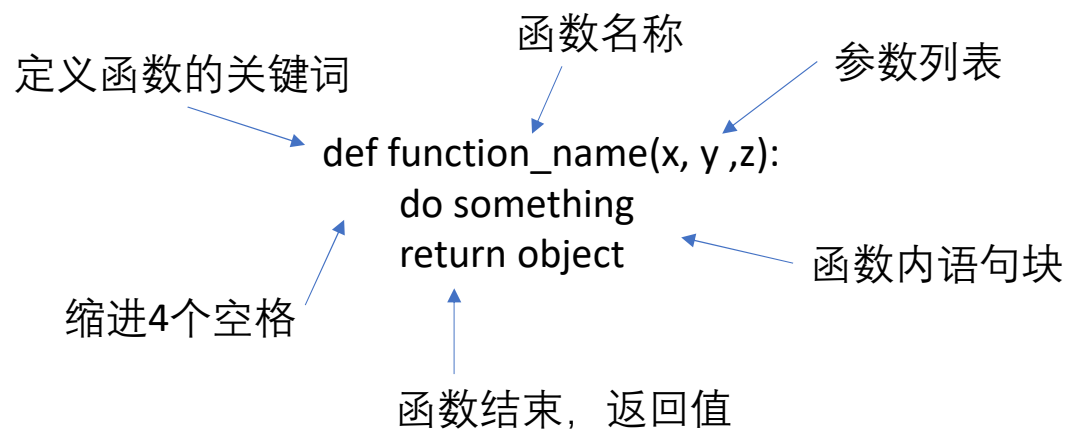
- 例题：制作一个满足如下功能的猜数游戏：
  - 计算机随机生成一个100以内的正整数；
  - 用户通过键盘输入数字，猜测计算机所生成的随机数。
  - 注意：对用户的输入次数不做限制。

# Python零基础轻松入门

老齐

# 函数基础

- 定义函数



# 函数基础

- 调用函数
  - 按照位置提供参数
  - 指明参数名称
  - 设置参数的默认值
- 返回值
  - 返回一个值
  - 返回多个值

# 练习

- 编写函数，实现斐波那契数列

$$\begin{cases} a_0 = 0 & n = 0 \\ a_1 = 1 & n = 1 \\ a_n = a_{(n-1)} + a_{(n-2)} & n \geq 2 \end{cases}$$

# 函数基础

- 参数收集
  - 一个“\*”的作用
  - 两个“\*”的作用

## 练习

- 假设有数据： `d = {'a': 39, 'b':40, 'c':99, 'd':100}`（字典的键值对还可以增加），编写函数，实现对这个字典中键值对的查询。例如向函数提供如 `a=1, b=40` 等参数，查询这些是否为此数据的值。



# 嵌套函数和装饰器

- 函数是对象
- 变量的作用域
- 装饰器的简单实现

# 练习

- 编写一个用于测试函数执行时间的装饰器函数

# 特殊函数

- lambda
- map
- filter

# Python零基础轻松入门

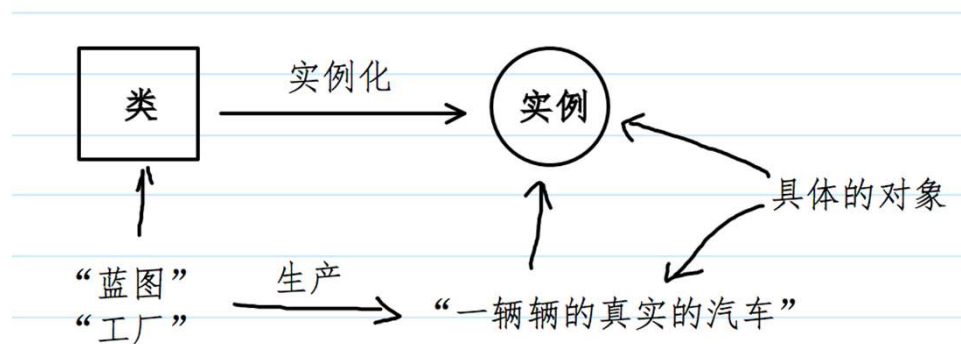
## 类

# 面向对象

- 对象（object）：表示客观世界问题空间中的某个具体事物，又表示软件系统解空间的中的基本元素
- 面向对象程序设计（Object-oriented programming, OOP）：是一种程序设计范型，也是一种程序开发的方法。

# 什么是类

- 类（class）是一种面向对象计算机编程语言的构造，是创建对象的蓝图，描述了所创建的对象共同的属性和方法



# 如何编写类

定义类的关键词      类的名称

```
class SuperMan:
```

英文半角状态下的冒号

类的文档

```
'''  
A class of superman  
'''
```

初始化方法

```
def __init__(self, name):  
    self.name = name  
    self.gender = 1 #1: male  
    self.single = False  
    self.illness = False
```

普通方法

```
def nine_negative_kungfu(self):  
    return "Ya! You have to die."
```

缩进4个空格

# 练习

- 编写一个程序，用于判断学生是否已经完成作业。如果完成，教师会给出表扬，否则要进行批评。



# 属性

- 属性：描述对象是什么
- 类属性
- 实例属性
- self的作用

# 方法

- 方法：描述对象能做什么
- 比较方法和函数
- 类方法
- 静态方法

# 练习

- 在网上购买商品，除了要支付商品总额，还要支付快递费用（显然不在“包邮区”）。

假设某网上书店与某快递公司签订了固定的快递费用，即每件（不论大小）都是5元。而对于买家，商家往往会在购买金额超过一定限额的时候免快递费。

请编写类，根据图书的单价、购买数量以及快递费，计算买家应支付的总额。

# 练习

- 创建一个类，当这个类实例化的时候，自动将数据集中的偶数和奇数分别用两个属性引用。

# 特性

- 继承
- 多态
- 封装和私有化

# 自定义对象类型

- 自定义简单的类型
- 控制属性访问
- 可调用对象

# 练习

- 创建表示温度的对象类型。

温度是表示物体冷热程度的物理量，而温度的测量只能通过物体随温度变化的某些特性间接进行，用来度量物体温度数值的标尺叫做温标。热力学温标，也称开尔文温标，对应单位是开尔文，用符号K表示。它是基本物理量之一。此外，生活中常用的还有摄氏温标和华氏温标，对应的单位就是摄氏度和华氏度。

# 练习

- 创建一种对象类型，能够将指定字符从某个字符串中选出。



# 构造方法

- 基本应用
- 单例模式

# 练习

- 对于质量单位，现实中有两套体系，一套体系是“官方”的，比如千克（公斤）、克；另外一套是“民间”的，比如斤、两。有的超市里面的价格标签常用“xx元/500克”，这算是照顾了两套体系，因为500克=1斤。

编写一个使用构造方法的类，用于实现“民间”体系向“官方”体系的转换，即以“1斤”实例化，则得到“500克”的实例。

# 迭代器和生成器

- 可迭代对象
- 迭代器
- 生成器



# Python零基础轻松入门

## —— 类

讲师：齐伟

1. 理解创建类的基本方法
2. 初步理解对象和面向对象

- 类（class）是一种面向对象计算机编程语言的构造，是创建对象的蓝图，描述了所创建的对象共同的属性和方法

```
class Superman:
    '''
    A class of superman
    '''
    def __init__(self, name):
        self.name = name
        self.gender = 1 #1: male
        self.single = False
        self.illness = False
    def nine_negative_kungfu(self):
        return "Ya! You have to die."
```

定义类的关键词

类的名称

英文半角状态下的冒号

类的文档

初始化方法

普通方法

缩进4个空格

编写一个程序，用于判断学生是否已经完成作业。如果完成，教师会给出表扬，否则要进行批评。



- 对象 (object)：表示客观世界问题空间中的某个具体事物，又表示软件系统解空间的中的基本元素
- 面向对象程序设计 (Object-oriented programming, OOP)：是一种程序设计范型，也是一种程序开发的方法。

# EDU

CSDN学院 IT实战派





# Python零基础轻松入门

## —— 属性

讲师：齐伟

1. 理解类属性和实例属性
2. 理解self的作用

- 类属性，又称静态属性
- 只有通过类才能修改
- 实例也拥有类属性，但不能修改类属性

- 实例属性，又称静态属性
- 通过实例创建
- 不同实例的实例属性不同
- 实例的`__dict__`显示当前实例的所有属性

- 类中的方法，如无特别规定，都是以self作为第一参数
- self引用当前实例

创建类，能够计算任意两个日期之间的天数、周数。

准备：pip3 install python-dateutil



# EDU

CSDN学院 IT实战派





# Python零基础轻松入门

## —— 方法

讲师：齐伟

1. 对比方法和函数
2. 理解类方法的特点
3. 理解静态方法的特点

- 名称的命名、代码块的编写方式都一样
- （实例）方法不能单独调用，只能通过实例/类调用
- 方法的第一个参数必须是self

- 使用装饰器：@classmethod
- 类方法第一个参数：cls，表示类本身

- 使用装饰器：@staticmethod
- 静态方法不与实例绑定

创建类，能够通过“年-月-日”字符串创建实例，并检验年、月、日是否合法。

# EDU

CSDN学院 IT实战派







# Python零基础轻松入门

## —— 继承

讲师：齐伟

1. 理解继承的含义
2. 掌握继承的实现方法

- 继承是对象的特性之一
- 继承是面向对象编程的一个重要概念

- 父类和子类
- 单继承
- 多继承

请编写“物理学家”的类，并将其作为“理论物理学家”和“实验物理学家”两个类的父类

# EDU

CSDN学院 IT实战派





# Python零基础轻松入门

## —— 多态与封装

讲师：齐伟

1. 理解多态的含义
2. 掌握Python实现封装的方法



- 多态是对象的特性之一
- Python语言天然具有多态特性

- 封装是对象的特性之一
- Python中通过私有化的方式实现封装

# EDU

CSDN学院 IT实战派





# Python零基础轻松入门

## —— 语音合成

讲师：齐伟

1. 如何使用SDK
2. 综合应用已学过的知识

# EDU

CSDN学院 IT实战派





# Python零基础轻松入门

## —— 网络爬虫

讲师：齐伟

# EDU

CSDN学院 IT实战派







# Python零基础轻松入门

## —— 标准库和第三方包

讲师：齐伟

1. 引入和使用标准库
2. 安装、引入和使用第三方包

- `import module`
- `from module import XXX`
- `import module as other_name`

- 安装: `pip install package_name`
- 来源: <https://pypi.org/>

# EDU

CSDN学院 IT实战派





# Python零基础轻松入门

## —— 文件读写

讲师：齐伟

1. 文件读写的一般方法
2. 读写特定类型文件

# EDU

CSDN学院 IT实战派







# Python零基础轻松入门

## —— 异常处理

讲师：齐伟

## 1. 了解异常处理的基本方法

# EDU

CSDN学院 IT实战派





# Python零基础轻松入门

## —— 定制类

讲师：齐伟

1. 理解类和类型
2. 掌握自定义对象类型的方法

# EDU

CSDN学院 IT实战派





# Python零基础轻松入门

## —— 编写和发布模块

讲师：齐伟

1. 了解模块和包的结构
2. 发布包的方法



模块：.py文件

Python搜索路径

包：有一定层次的目录结构

- “.py” 文件或子目录
- \_\_init\_\_.py

# EDU

CSDN学院 IT实战派





# Python零基础轻松入门

## —— 读写SQLite数据库

讲师：齐伟

1. 简要了解SQLite数据库
2. 用Python读写SQLite数据库

SQLite是小型的关系型数据库

不需要单独的服务、零配置

Python内置驱动模块

像文件那样操作

# EDU

CSDN学院 IT实战派





# Python零基础轻松入门

## —— 读写MySQL数据库

讲师：齐伟



1. 简要了解Python DB-API
2. 简要了解MySQL数据库
3. 用Python读写MySQL数据库

Python Database API, Python的DB-API, 为大多数的数据库实现了接口, 使用它连接各数据库后, 就可以用相同的方式操作各数据库。

Python DB-API的使用流程:

- (1) 引入API模块
- (2) 获取与数据库的连接
- (3) 执行sql语句和存储过程
- (4) 关闭数据库连接

两个针对事务的方法:

commit() 提交

rollback() 回滚

cursor的方法:

callproc(self, procname, args)

execute(self, query, args) : 执行sql语句,

executemany(self, query, args) : 执行单挑sql语句,但是重复执行参数列表里的参数

nextset(self): 移动到下一个结果集

cursor用来接收返回值的方法:

fetchall(self): 接收全部的返回结果行.

fetchmany(self, size=None): 接收size条返回结果行

fetchone(self): 返回一条结果行.

rowcount: 属性, 返回执行execute() 方法后影响的行数。

scroll(self, value, mode='relative '): 移动指针到某一行;

如果mode='relative',则表示从当前所在行移动value条,

如果 mode='absolute',则表示从结果集的第一行移动value条.

MySQL 是最流行的关系型数据库管理系统之一

MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目前属于 Oracle 公司。

MySQL 是开源的（社区版）

MySQL 支持大型的数据库。可以处理拥有上千万条记录的大型数据库。

MySQL 使用标准的 SQL 数据语言形式。

MySQL 可以运行于多个系统上，并且支持多种语言。

MySQL 是可以定制的，采用了 GPL 协议，你可以修改源码来开发自己的 MySQL 系统。

根据不同操作系统安装MySQL数据库

基本组成：

MySQL服务器。

MySQL 客户端程序，用于连接并操作Mysql服务器。

PyMySQL

mysql-connector-python

MySQL-python

.....

```
pip install mysql-connector-python
```

# EDU

CSDN学院 IT实战派

