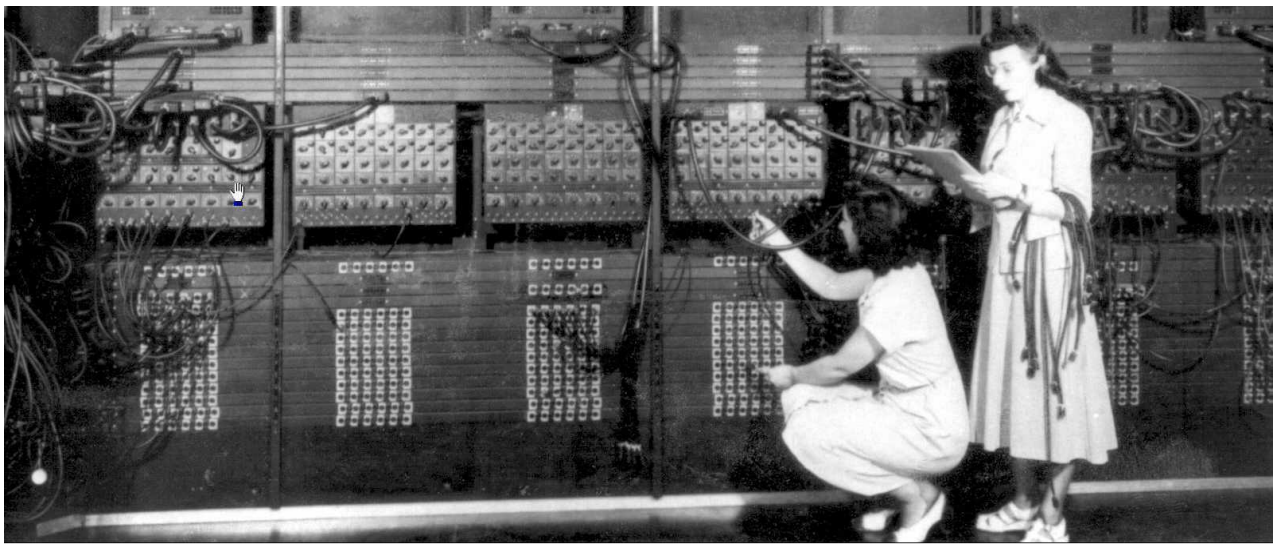
Day1 Python 基础快速入门

一、编程语言介绍

什么是编程语言?

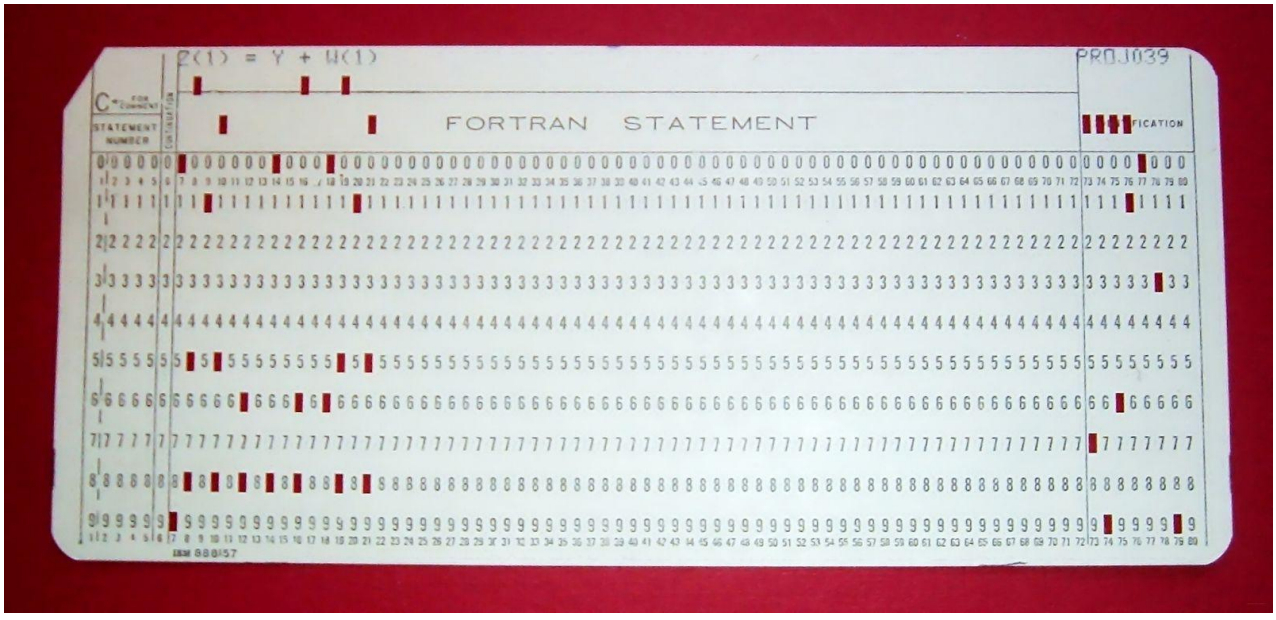
科学家可以通过晶体管，控制电路开关，让电路实现2进制的表示与运算， 计算机只认识0101。

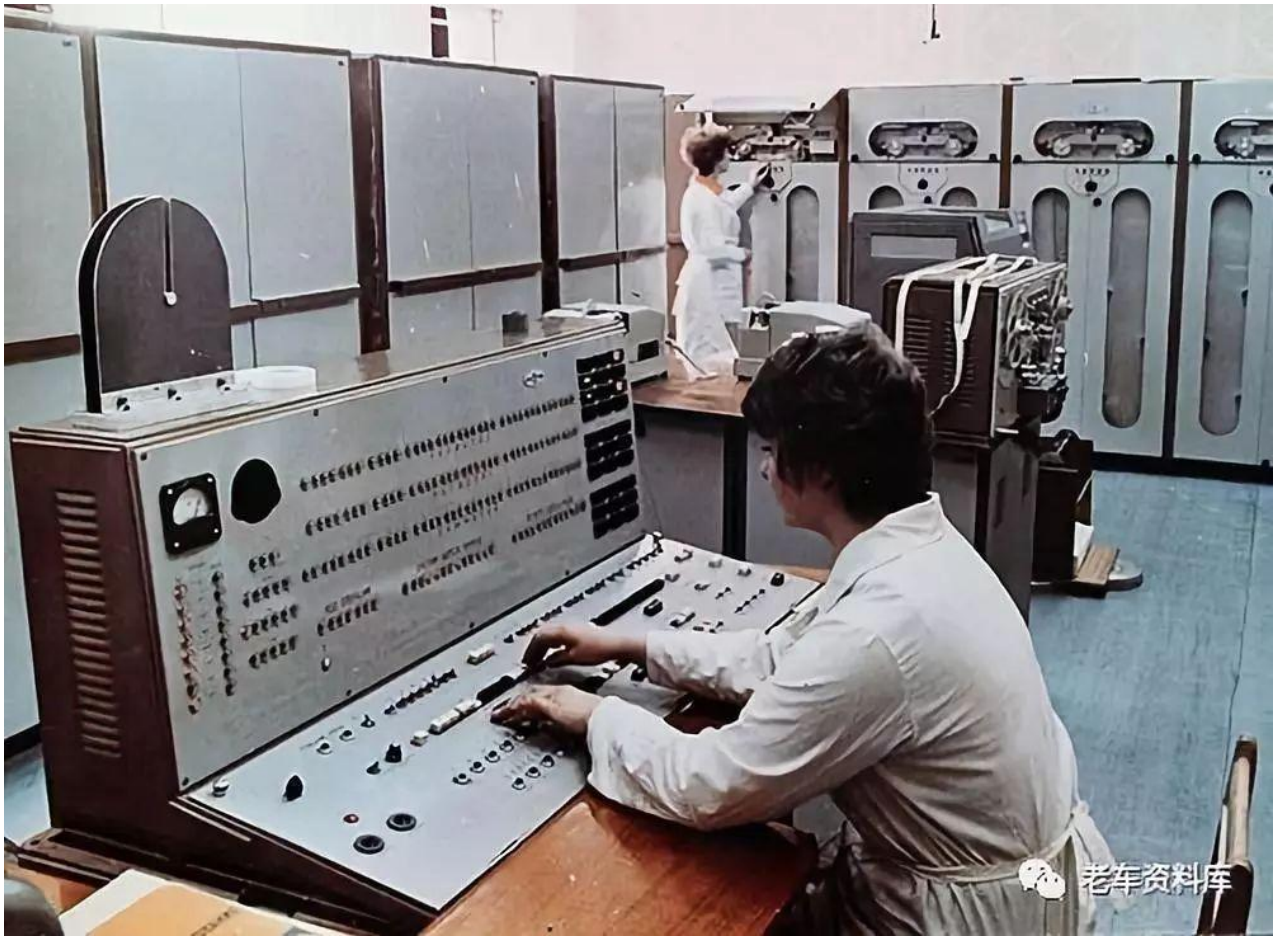
古时候在计算机刚诞生时，人们就是靠操纵一堆电路开关，实现运算。‘



这个效率太低了。。。 干半天也算不出几个数

于是发明了穿孔卡片。。。


Z(1) = Y + W(1)
PROJ039
FORTRAN STATEMENT
STATEMENT NUMBER
IBM 888157



第一个编程语言的诞生

用纸带效率太低了。。。。这时候，天才格雷斯·霍珀（Grace Hopper）姐姐，想出了一个很棒的点子。

1959年，开发世界上第一个广泛应用的编程语言COBOL

她最大的贡献是发明了世界上第一个编译器 (Compiler)，名字叫做 A-0。当时是没有任何组合语言及程序语言存在的，所有的程序设计人员都要把程序翻译成机器码，01101010110 这样的形式，在纸上打孔，再送到机器里去读。

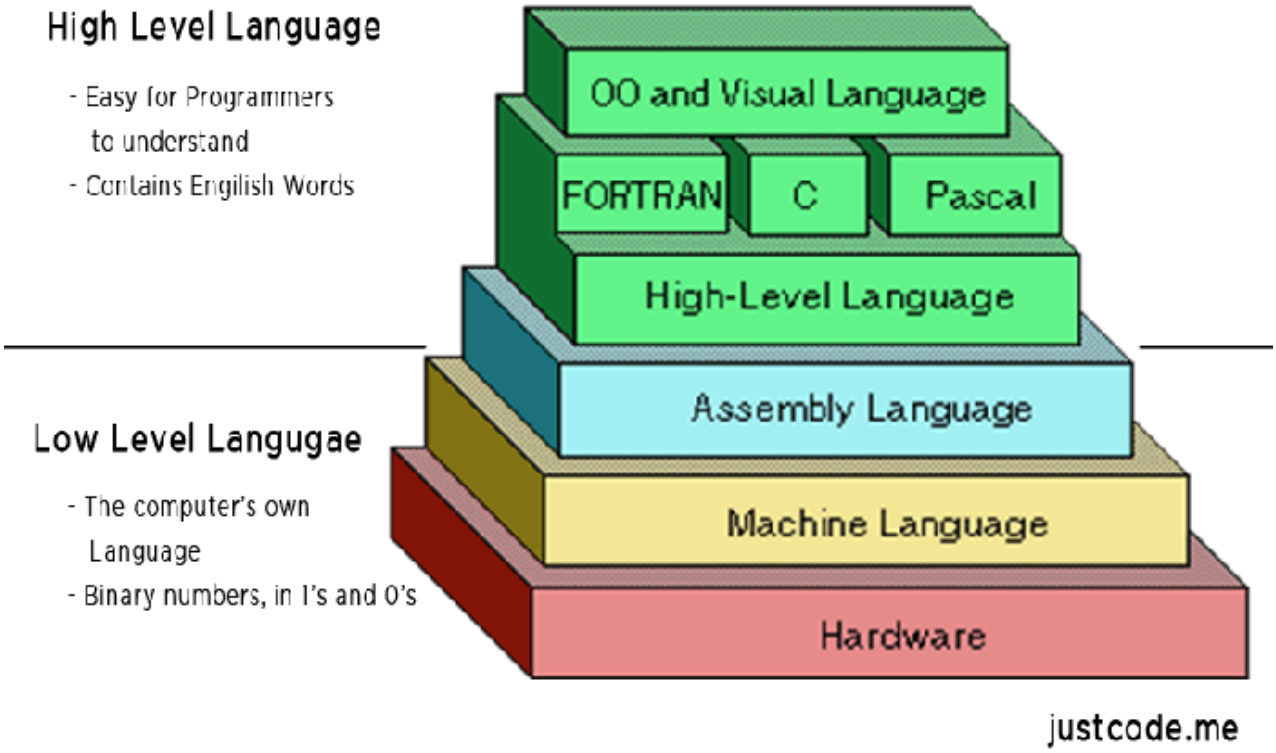
Grace 产生了一种想法，她想设计一种程序，让人可以用类似英文的语法，把想做的事写下来，然后用这个程序把英文翻译成机器的语法，交给机器去执行。这个想法就是今日的 Compiler (编译器)。

A-0 的原理是：编译程序把穿孔卡加载到计算机中。然后编写的程序将被送入计算机。计算机会吐出另一组包含机器代码的卡片。第二组卡片将被装入计算机，计算机就可以执行这段新的程序了。


SHORPY


COBOL
0717 F00092 LF1823
0718 BF1940 Q00723
0719 F00087 LF1823
0720 BF1940 Q00728
0721 R00803 U00740
0722 K00000 C00803
0723 000000 U00738
0724 B01940 AH1829
0740
+
126 INSTRUCTIONS
0805

编程语言分类

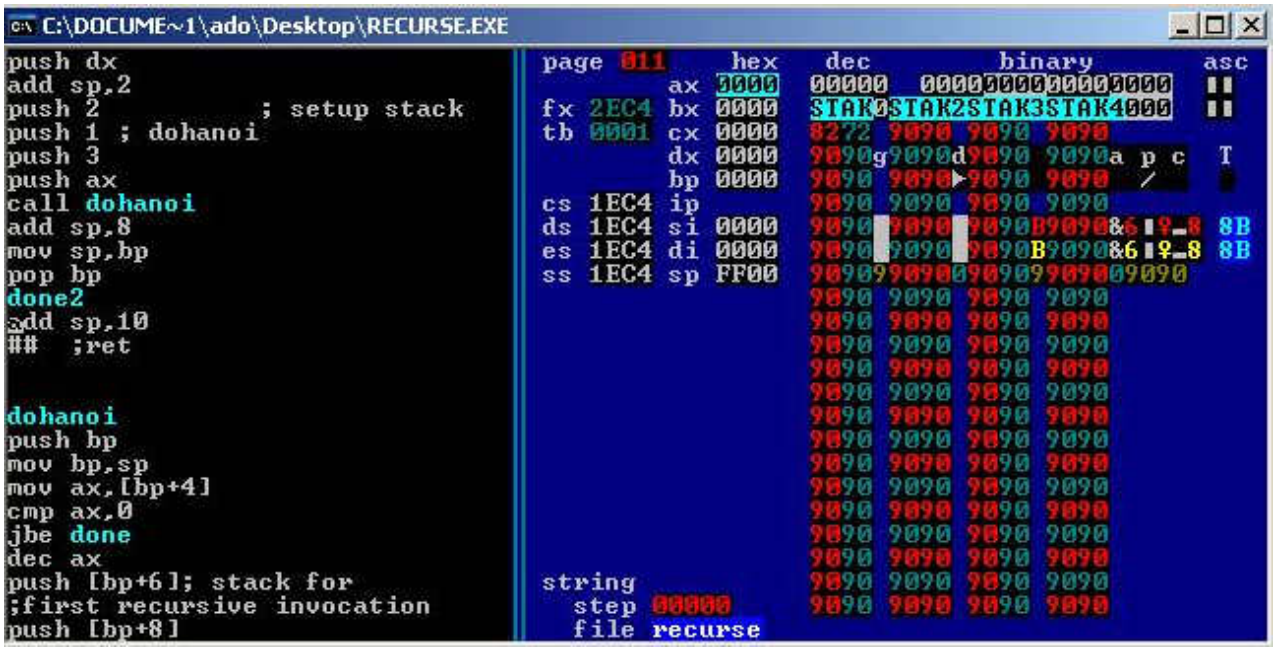


低级语言-汇编

我们知道，CPU 只负责计算，本身不具备智能。你输入一条指令（instruction），它就运行一次，然后停下来，等待下一条指令。

这些指令都是二进制的，称为操作码（opcode），比如加法指令就是`00000011`。编译器的作用，就是将高级语言写好的程序，翻译成一条条操作码。

对于人类来说，二进制程序是不可读的，根本看不出来机器干了什么。为了解决可读性的问题，以及偶尔的编辑需求，就诞生了汇编语言。



汇编语言是二进制指令的文本形式，与指令是一一对应的关系。比如，加法指令`00000011`写成汇编语言就是 ADD。只要还原成二进制，汇编语言就可以被 CPU 直接执行，所以它是最底层的低级语言。

样的话，就多出一个步骤，要把这些文字指令翻译成二进制，这个步骤就称为 assembling，完成这个步骤的程序就叫做 assembler。它处理的文本，自然就叫做 aseembly code。标准化以后，称为 assembly language，缩写为 asm，中文译为汇编语言。

高级语言分类

按生态分类

C（1970）：操作系统、嵌入式、驱动开发

C++：图形图像、科研、通信、桌面软件、游戏、游戏服务器

C#：Windows桌面软件、.NET Web开发

Java（1994）： Java SE：跨平台的桌面应用，Android；Java EE：企业级应用，web开发、服务器后端；Java Android：用于安卓开发应用

GO（2009）：高性能高并发服务器应用、云计算

Erlang（1991）：高并发服务器应用，多用于游戏

Python（1989）：人工智能、数据分析、Web开发、爬虫、游戏开发、科学计算、自动化开发

Php(1995): web开发

Ruby(1995)：web开发，用的少了

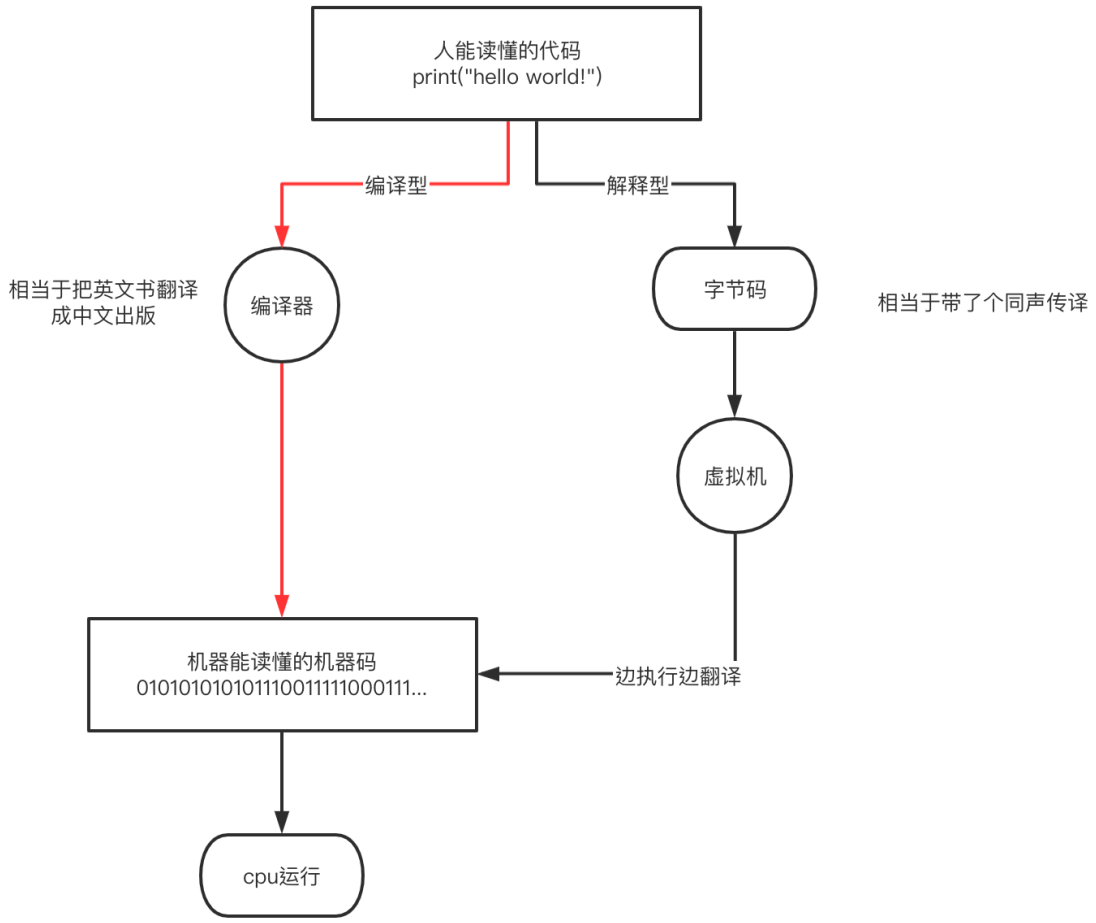
Perl(1987)：运维、文本处理，用的较少

Javascript(1995)：前端开发，在node中可以做后端

HTML/CSS(1995)：标记语言，主要是给前端工程师构建web页面使用

按编译原理不同

主要分为解释型 、 编译型



特点：

编译型语言执行速度快，缺点是跨平台略差，因为直接跟操作系统的各种接口打交道，windows,linux,mac都不一样。

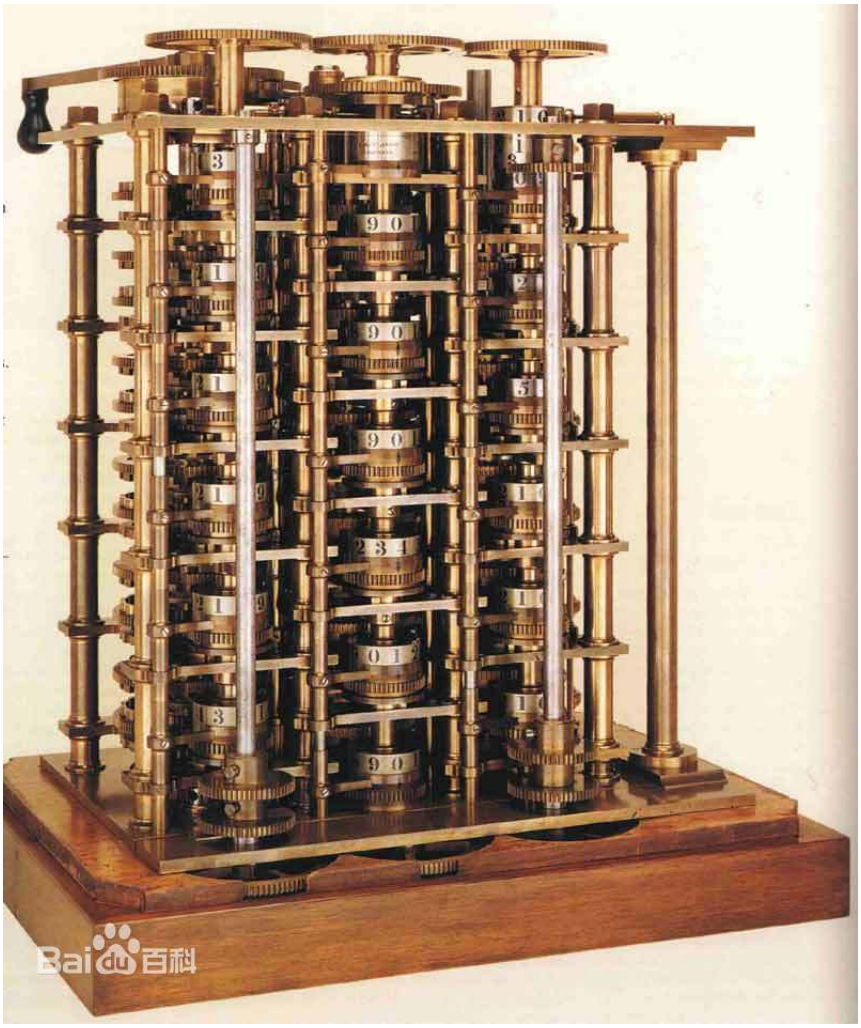
解释型跨平台好，因为解释器帮着封装了跟各操作系统交互的接口，一份代码，到处各平台使用，缺点是执行速度慢，依赖解释器运行

第一个程序员是女的

阿达·洛芙莱斯(Ada Lovelace 1815-1852)，是英国著名诗人拜伦的女儿。

出生于伦敦的阿达，在17岁时于剑桥大学第一次见到了著名的数学家、发明家兼机械工程师查尔斯·巴贝其，而这次相遇成了阿达人生的转折点。巴贝其当时正致力于发明分析机，而阿达则致力于为这台分析机编写算法。在这个过程中，阿达第一次接触到“差分机”这个概念，阿达日后的在和巴贝奇教授讨论差分机的过程中，预言了通用计算机可能。

譬如她建议建议用二进制数代替原来的十进制数，表明分析机可以接受各种各样的穿孔卡：“控制卡”、“数据卡”、“操作卡”。她还提议数字和其他符号如字母都可以“编码”成数字数据，机器可以处理它们。甚至早于现代计算机出现200年，提出了分析机的记忆能力的想法，指出分析机应该有存储位置或地址，并且有“注解或备忘”的可能性。



人类--》编译器——》0101010

三、Python介绍&发展

Python的创始人为吉多·范罗苏姆（Guido van Rossum）。1989年的圣诞节期间，Guido开始写Python语言的编译器。Python这个名字，来自Guido所挚爱的电视剧Monty Python’s Flying Circus。他希望这个新的叫做Python的语言，能符合他的理想：创造一种C和shell之间，功能全面，易学易用，可拓展的语言。

最新的TIOBE排行榜，Python赶超C++占据第3， 与Java、C一起成为全球最流行的3大编程语言。

Feb 2021	Feb 2020	Change	Programming Language	Ratings	Change
1	2	↑	C	16.34%	-0.43%
2	1	↓	Java	11.29%	-6.07%
3	**3**		**Python**	**10.86%**	**+1.52%**
4	4		C++	6.88%	+0.71%
5	5		C#	4.44%	-1.48%
6	6		Visual Basic	4.33%	-1.53%
7	7		JavaScript	2.27%	+0.21%
8	8		PHP	1.75%	-0.27%
9	9		SQL	1.72%	+0.20%
10	12	↑	Assembly language	1.65%	+0.54%
11	13	↑	R	1.56%	+0.55%
12	26	⇈	Groovy	1.50%	+1.08%
13	11	↓	Go	1.28%	+0.15%
14	15	↑	Ruby	1.23%	+0.39%
15	10	⇊	Swift	1.13%	-0.33%
16	16		MATLAB	1.06%	+0.27%
17	18	↑	Delphi/Object Pascal	1.02%	+0.27%
18	22	⇈	Classic Visual Basic	1.01%	+0.40%
19	19		Perl	0.93%	+0.23%
20	20		Objective-C	0.89%	+0.20%

Python崇尚优美、清晰、简单，上手简单，非常适合做为第一门编程语言来学习。

与Java, C这两位编程界大佬长期霸屏编程语言排行榜前列不同的是，Python是近几年才异军突起、爬到了前三的位置。活脱脱一个屌丝逆袭故事。

Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2017	2012	2007	2002	1997	1992	1987
Java	1	1	1	1	14	-	-
C	2	2	2	2	1	1	1
C++	3	3	3	3	2	2	4
C#	4	4	7	14	-	-	-
Python	5	7	6	9	27	-	-
PHP	6	5	4	5	-	-	-
JavaScript	7	9	8	7	20	-	-
Visual Basic .NET	8	21	-	-	-	-	-
Perl	9	8	5	4	4	11	-
Assembly language	10	-	-	-	-	-	-
COBOL	25	31	17	6	3	13	[illegible]
Lisp	31	12	14	10	9	9	2
Prolog	33	37	26	13	18	14	3
Pascal	102	13	19	29	8	3	5

世界上的编程语言有600多种，但真正大家主流在使用的最多二三十种，不同的语言有自己的特点和擅长领域，随着计算机的不断发展，新语言在不断诞生，也同时有很多老旧的语言慢慢无人用了。有个权威的语言排名网站，可以看到主流的编程语言是哪些

Python为何能逆袭?

答案：还是因为太优秀了哈哈。

评判一个编程语言的优劣有多种维度， 一般包括“开发效率”、“学习曲线”、“生态圈”、"运行速度"等，Python在“开发效率”、“学习曲线”、“生态圈”这3个维度上可以说是稳拿第一了。

开发效率——由于语法简洁明了，又有丰富的、现成的各种模块库，开发什么功能都不用重新造轮子，直接在前辈的写好的代码基础上扩展即可，大大提高开发效率。

学习曲线——Python算是最适合小白上手的语言了。 很多人觉得计算机难学，是因为一开始就没选对语言，你一上来就搞C\C++\Java啥的，那指针呀、链表呀、面向对象呀就够你喝一壶的。 而学Python符合人性的学习曲线，由简入繁，先易后难。

生态圈——Python自1989年诞生至今30余年，在IT技术的各个领域方向都发展出强大的生态圈，在人工智能、数据分析、爬虫开发、游戏开发、自动化运维\测试、web开发、后端开发、科学运算、GUI编程等方面均全面开花，被称为编程界的瑞士军刀、万能语言。

运行速度——Python 的运行速度相比C语言确实慢很多，跟JAVA相比也要慢一些，因此这也是很多所谓的大牛不屑于使用Python的主要原因，但其实这里所指的运行速度慢在大多数情况下用户是无法直接感知到的，必须借助测试工具才能体现出来，比如你用C运一个程序花了0.01s,用Python是0.1s,这样C语言直接比Python快了10倍,算是非常夸张了，但是你是无法直接通过肉眼感知的，因为一个正常人所能感知的时间最小单位是0.15-0.4s左右，哈哈。其实在大多数情况下Python已经完全可以满足你对程序速度的要求，除非你要写对速度要求极高的搜索引擎、网络游戏等，这种情况下，当然还是建议你用C去实现的。

综上原因，导致Python必火无疑。

目前Python主要应用领域：

1. WEB开发——最火的Python web框架Django, 支持异步高并发的Tornado框架，短小精悍的flask,bottle, Django官方的标语把Django定义为the framework for perfectionist with deadlines(大意是一个为完全主义者开发的高效率web框架)
2. 网络编程——支持高并发的Twisted网络框架， py3引入的asyncio使异步编程变的非常简单
3. 爬虫——爬虫领域，Python几乎是霸主地位，Scrapy\Request\BeautifuSoap\urllib等，想爬啥就爬啥
4. 云计算——目前最火最知名的云计算框架就是OpenStack,Python现在的火，很大一部分就是因为云计算
5. 人工智能、数据分析—— Python 是目前公认的人工智能和数据分析领域的必备语言
6. 自动化运维——问问中国的每个运维人员，运维人员必须会的语言是什么？10个人相信会给你一个相同的答案，它的名字叫Python
7. 金融分析——我个人之前在金融行业，10年的时候，我们公司写的好多分析程序、高频交易软件就是用的Python,到目前,Python是金融分析、量化交易领域里用的最多的语言
8. 科学运算—— 97年开始，NASA就在大量使用Python在进行各种复杂的科学运算，随着NumPy, SciPy, Matplotlib, Enthought librarys等众多程序库的开发，使的Python越来越适合于做科学计算、绘制高质量的2D和3D图像。和科学计算领域最流行的商业软件Matlab相比，Python是一门通用的程序设计语言，比Matlab所采用的脚本语言的应用范围更广泛
9. 游戏开发——在网络游戏开发中Python也有很多应用。相比Lua or C++,Python 比 Lua 有更高阶的抽象能力，可以用更少的代码描述游戏业务逻辑，与 Lua 相比，Python 更适合作为一种 Host 语言，即程序的入口点是在 Python 那一端会比较好，然后用 C/C++ 在非常必要的时候写一些扩展。

Python 非常适合编写 1 万行以上的项目，而且能够很好地把网游项目的规模控制在 10 万行代码以内。另外据我所知，知名的游戏 < 文明 >、 就是用Python写的

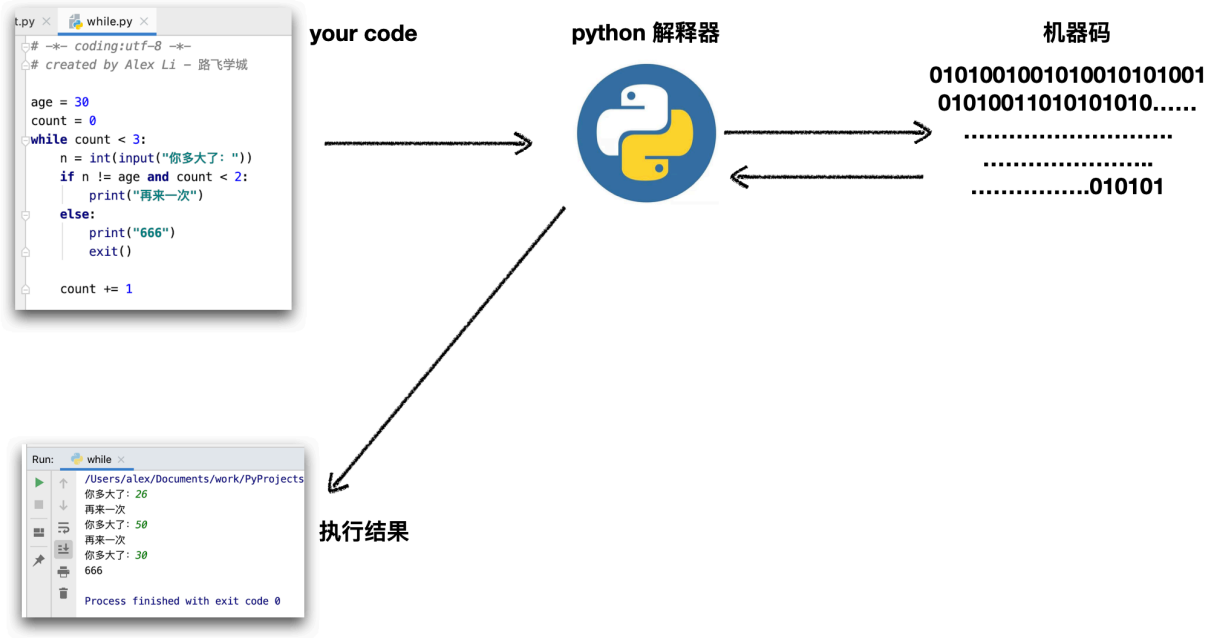
Python在一些公司的应用：

- 谷歌：Google App Engine 、code.google.com 、Google earth 、谷歌爬虫、Google广告等项目都在大量使用Python开发
- CIA: 美国中情局网站就是用Python开发的
- NASA: 美国航天局(NASA)大量使用Python进行数据分析和运算
- YouTube:世界上最大的视频网站YouTube就是用Python开发的
- Dropbox:美国最大的在线云存储网站，全部用Python实现，每天网站处理10亿个文件的上传和下载
- Instagram:美国最大的图片分享社交网站，每天超过3千万张照片被分享，全部用python开发
- Facebook:大量的基础库均通过Python实现的
- Redhat: 世界上最流行的Linux发行版本中的yum包管理工具就是用python开发的
- 豆瓣: 公司几乎所有的业务均是通过Python开发的
- 知乎: 国内最大的问答社区，通过Python开发(国外Quora)
- 春雨医生：国内知名的在线医疗网站是用Python开发的
- 大话西游、Eve，均是Python开发
- 除上面之外，还有搜狐、金山、腾讯、盛大、网易、百度、阿里、淘宝 、土豆、新浪、果壳等公司都在使用Python完成各种各样的任务。

四、Python开发环境安装

为何要装Python环境？

因为你写的代码是人类可读的，但计算机是靠电流驱动的，它只认识010100101，我们即将安装的python环境就是Python解释器，它就像个翻译官，把人类代码翻译成机器能读懂的010101二进制代码，这样才能运行哈。



Python目前已支持所有主流操作系统，在Linux,Unix,Mac系统上自带Python环境，在Windows系统上需要安装一下，超简单。

sublime...

在这先不用pycharm?

Hello World

五、变量与内存

计算机为何需要内存?

CPU运行速度快

硬盘慢

什么是变量?

变量，是用于在内存中存放程序数据的容器，怎么理解呢?

计算机的最核心功能就是“计算”， 计算需要数据源，数据源要存在内存里，比如我要把小明的姓名、身高、年龄信息存下来，后面程序会调用，怎么存呢，直接设置一个“变量名=值”， 就可以

变量名定义规则

1. 变量名只能是 字母、数字或下划线的任意组合
2. 变量名的第一个字符不能是数字
3. 以下关键字不能声明为变量名['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or', 'pass', 'print', 'raise', 'return', 'try', 'while', 'with', 'yield']

变量命名规范

名字不能瞎起。。。

出手就是专业

常用定义方式

驼峰体

```
AgeOfBlackGirl = 56
NumberOfStudents = 80
```

下划线

```
age_of_black_girl = 56
number_of_students = 80
```

你觉得哪种更清晰，哪种就是官方推荐的，我想你肯定会先第2种

定义变量不好的方式举例

- 变量名为中文、拼音
- 变量名过长
- 变量名词不达意

常量

常量即指不变的量，如pai 3.141592653..., 或在程序运行过程中不会改变的量

在Python中没有一个专门的语法代表常量，程序员约定俗成用变量名全部大写代表常量

```
MAX_USER_LIMIT = 1000
```

> 在c语言中有专门的常量定义语法， `const int count = 60;` 一旦定义为常量，更改即会报错

六、基本数据类型之数字、字符串

什么是数据类型?

我们人类可以很容易的分清数字与字符的区别，但是计算机并不能呀，计算机虽然很强大，但从某种角度上看又很傻，除非你明确的告诉它，1是数字，“汉”是文字，否则它是分不清1和‘汉’的区别的，因此，在每个编程语言里都会有一个叫数据类型的东东，其实就是对常用的各种数据类型进行了明确的划分，你想让计算机进行数值运算，你就传数字给它，你想让他处理文字，就传字符串类型给他。Python中常用的数据类型包括多种，今天我们暂只讲4种， 数字、字符串、布尔类型、列表。

数字

int（整型）

在64位系统上，整数的位数为64位，取值范围为-2的63次方～2的63次方-1，
即-9223372036854775808～9223372036854775807

long（长整型）

跟C语言不同，Python的长整数没有指定位宽，即：Python没有限制长整数数值的大小，但实际上由于机器内存有限，我们使用的长整数数值不可能无限大。

注意，自从Python2.2起，如果整数发生溢出，Python会自动将整数数据转换为长整数，所以如今在长整数数据后面不加字母L也不会导致严重后果了。

> 注意：在Python3里不再有long类型了，全都是int

```
>>> a= 2**64
>>> type(a)  #type()是查看数据类型的方法
>>> b = 2**60
>>> type(b)
```

float (浮点型)

即小数

```
>>> type(2.32)
```

字符串str

在Python中,加了引号的字符都被认为是字符串!

```
>>> name = "Alex Li" #双引号
>>> age = "22"       #只要加引号就是字符串
>>> age2 = 22          #int
>>>
>>> msg = '''My name is Alex, I am 22 years old!'''  #我擦，3个引号也可以
>>>
>>> hometown = 'ShanDong'   #单引号也可以
```

那单引号、双引号、多引号有什么区别呢？ 让我大声告诉你，单双引号木有任何区别，只有下面这种情况 你需要考虑单双的配合

```
msg = "My name is Alex , I'm 22 years old!"
```

多引号

多引号什么作用呢？作用就是多行字符串必须用多引号

```
msg = '''
今天我想写首小诗，
歌颂我的同桌，
你看他那乌黑的短发，
好像一只炸毛鸡。
'''
print(msg)
```

字符串拼接

数字可以进行加减乘除等运算，字符串呢？让我大声告诉你，也能？what ?是的，但只能进行”相加”和”相乘”运算。

```
>>> name
'Alex Li'
>>> age
'22'
>>>
>>> name + age  #相加其实就是简单拼接
'Alex Li22'
>>>
>>> name * 10 #相乘其实就是复制自己多少次，再拼接在一起
'Alex LiAlex LiAlex LiAlex LiAlex LiAlex LiAlex LiAlex LiAlex Li'
```

不用尝试相减、相除，臣妾办不到呀。

注意，**字符串的拼接只能是双方都是字符串**，不能跟数字或其它类型拼接

```
>>> name = "Alex Li"
>>> age = 23
>>>
>>> type(name),type(age)
(<type 'str'>, <type 'int'>)

>>> name + age  # 字符串跟数字相加报错
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
```

布尔型bool

布尔类型很简单，就两个值 ， 一个True(真)，一个False(假), 它的作用主要是用作逻辑判断

但其实你们并不明白对么？ let me explain, 我现在有2个值 ， a=3, b=5 , 我说a>b你说成立么? 我们当然知道不成立，但问题是计算机怎么去描述这成不成立呢？或者说a< b是成立，计算机怎么描述这是成立呢?

没错，答案就是，用布尔类型

```
>>> a=3
>>> b=5
>>>
>>> a > b #不成立就是False,即假
False
>>>
>>> a < b #成立就是True, 即真
True
```

计算机为什么要描述这种条件呢？因为接下来就可以根据条件结果来干不同的事情啦呀！比如

```
if a > b
   print(a is bigger than b )
else
   print(a is smaller than b )
```

上面是伪代码，但是不是意味着， 计算机就可以根据判断结果不同，来执行不同的动作啦?

列表

你要把你交过的所有女朋友的名字存到内存里

基本增删改查

七、常用运算符

计算机可以进行的运算有很多种，可不只加减乘除这么简单，运算按种类可分为算数运算、比较运算、逻辑运算、赋值运算、成员运算、身份运算、位运算，今天我们暂只学习算数运算、比较运算、逻辑运算、赋值运算、成员运算

7.1 算术运算符

以下假设变量：**a=10，b=20**

运算符	描述	实例
+	加 - 两个对象相加	a + b 输出结果 30
-	减 - 得到负数或是一个数减去另一个数	a - b 输出结果 -10
*	乘 - 两个数相乘或是返回一个被重复若干次的字符串	a * b 输出结果 200
/	除 - x除以y	b / a 输出结果 2
%	取模 - 返回除法的余数	b % a 输出结果 0
**	幂 - 返回x的y次幂	a**b 为10的20次方， 输出结果 100000000000000000000
//	取整除 - 返回商的整数部分	9//2 输出结果 4 , 9.0//2.0 输出结果 4.0

7.2 比较运算符

以下假设变量：**a=10，b=20**

运算符	描述	实例
==	等于 - 比较对象是否相等	(a == b) 返回 False。
!=	不等于 - 比较两个对象是否不相等	(a != b) 返回 true.
<>	不等于 - 比较两个对象是否不相等	(a <> b) 返回 true。这个运算符类似 != 。
>	大于 - 返回x是否大于y	(a > b) 返回 False。
<	小于 - 返回x是否小于y。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。注意，这些变量名的大写。	(a < b) 返回 true。
>=	大于等于 - 返回x是否大于等于y。	(a >= b) 返回 False。
<=	小于等于 - 返回x是否小于等于y。	(a <= b) 返回 true。

注意:<>在python3里已经取消了，仅在py2里有效

6.3 赋值运算符

以下假设变量：**a=10，b=20**

运算符	描述	实例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c *= a 等效于 c = c * a
/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c **= a 等效于 c = c ** a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

6.4 逻辑运算

以下假设变量：**a=10，b=20**

运算符	描述	实例
and	判断多个条件均为真时，返回真	a>10 and b > 10 结果为False
or	判断多个条件任意条件为真时，返回真	a> 10 or b > 10 结果为True
not	取反	not a > b 结果为True

注意了还可以多个or 或and 拼在一起

```
(1, 3, 10)
>>>
>>> a > 0 or b < 3
True
>>> a > 0 or b < 3 and c > 10
```

Or的优先级更高

```
>>> 10 > 5 and 8 < 5 or 7 < 9 and 4 < 7 and 8 > 8
False
>>>
>>> 10 > 5 and 8 < 5 or 7 < 9 and 4 < 7 and 8 > 8 or 4 < 5
True
```

> 注意，我画的红线，只要遇到or 就相当于把公式分成2半了

6.5 成员运算

只有in, not in 2种， 用来测试数据中是否包含了一系列的成员

运算符	描述	实例
in	如果在指定的序列中找到值返回True，否则返回False。	x 在 y序列中 , 如果x在y序列中返回True。
not in	如果在指定的序列中没有找到值返回True，否则返回False。	x 不在 y序列中 , 如果x不在y序列中返回True。

```
>>> names
['金角大王Alex', 'Jack', 'Rain', 'Rachel', 'Mack', '小强']
>>> "Jack" in names
True
>>> "Jack" not in names
False
>>> s = "我是沙河最帅的仔"
>>> "沙河" in s
True
```

- 注意，可以用来测试字符串、列表、元组、字典、集合，**但是不能测试数字类型**

八、读取用户指令

个人信息卡

字符串格式化

注释

```
name = input("Please input your name:")
age = input("Please input your age:")
hobbie = input("Your hobbie:")
job  = input("Your job:")

# 格式化方式一
person_info = f'''
-------------------------{name} info--------------------
name: {name}
age : {age}
hobbie: {hobbie}
job: {job}
_______________________________end _______________________________
'''
print(person_info)

# 格式化方式2
person_info2 = '''
-------------------------%s info--------------------
name: %s
age : %s
hobbie: %s
job: %s
_______________________________end _______________________________
''' %(name,name,age,hobbie,job)

print(person_info2)
```

九、流程控制之if...else..

单分支

```
loneliness = 75
if loneliness > 70:
    print("I am lonely , go to do massage...")
```

双分支

```
loneliness = 70
if loneliness > 70:
    print("I am lonely ,need massage...")
else:
    print("现在是贤者时间")
```

多分支

```
budget = 1000
if budget < 500:
    print("这个预算没得选...")
elif budget < 800:
    print("6分楼凤水平...")
elif budget < 1500:
    print("7分还不错")
elif budget < 2000:
    print("深圳嫩模...")
elif budget < 3000:
    print("香..")
```

缩进

> 这里必须要插入这个缩进的知识点

你会发现，上面的if代码里，每个条件的下一行都缩进了4个空格，这是为什么呢？这就是Python的一大特色，强制缩进，目的是为了让程序知道，每段代码依赖哪个条件，如果不通过缩进来区分，程序怎么会知道，当你的条件成立后，去执行哪些代码呢？

在其它的语言里，大多通过`{}`来确定代码块，比如C,C++,Java,Javascript都是这样，看一个JavaScript代码的例子

```
var age = 56
if ( age < 50){
  console.log("还能折腾")
    console.log('可以执行多行代码')
}else{
   console.log('太老了')
}
```

在有`{}`来区分代码块的情况下，缩进的作用就只剩下让代码变的整洁了。

Python是门超级简洁的语言，发明者定是觉得用`{}`太丑了，所以索性直接不用它，那怎么能区分代码块呢？答案就是强制缩进。

Python的缩进有以下几个原则:

- 顶级代码必须顶行写，即如果一行代码本身不依赖于任何条件，那它必须不能进行任何缩进
- 同一级别的代码，缩进必须一致
- 官方建议缩进用4个空格，当然你也可以用2个，如果你想被人笑话的话。

十、嵌套分支

注意if嵌套最多不超过4层，不是不行，而是层级太多会使你的程序笨拙，后续不易扩展，相当于把代码写死了。

```
loneliness = 80
money = 1000

if loneliness > 70 :
    print("寂寞难耐，要去大宝剑...")
    if money < 1000:
        print("钱太少，只能自己解决...")
```

十一、本日总结:

十二、本日作业: