Python快速入门

嵩 天

不容错过的Python语言概述

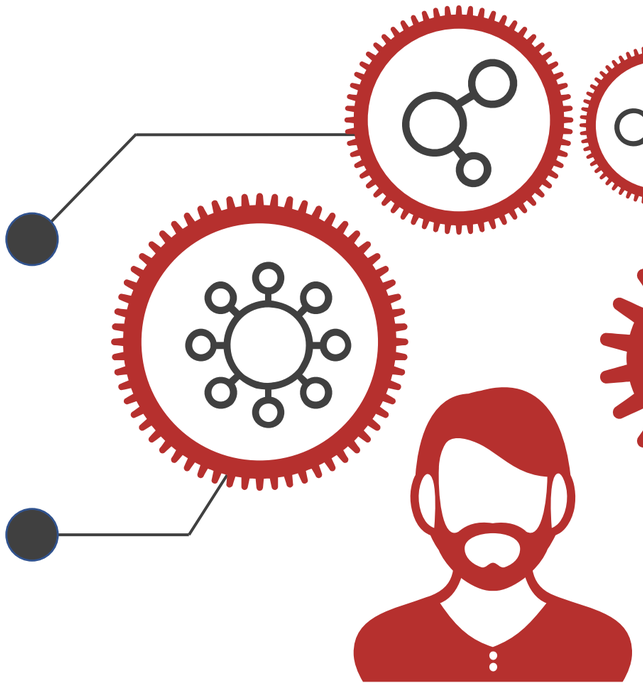
嵩 天

Python快速入门

单元开篇

Python123

单元开篇



(2) 编译与解释

(3) Python语言特点

(1) 演进中的编程语言

(4) 计算生态与超级语言

不容错过的Python语言概述

Python快速入门

演进中的编程语言

Python123

编程语言种类

Basic, C, C++, C#, CSS, Fortran, Go, HTML, Java,

JavaScript, Lisp, Lua, VC++, Object C, Pascal, Perl, PHP,

PostScript, Python, Ruby, Scala, SQL, Swift, VBA,

VB.NET, Verilog, VHDL, Visual Basic...

编程语言，也是一个江湖！

计算机技术演进过程

计算机系统结构时代

1946-1981

网络和视窗时代

1981-2008

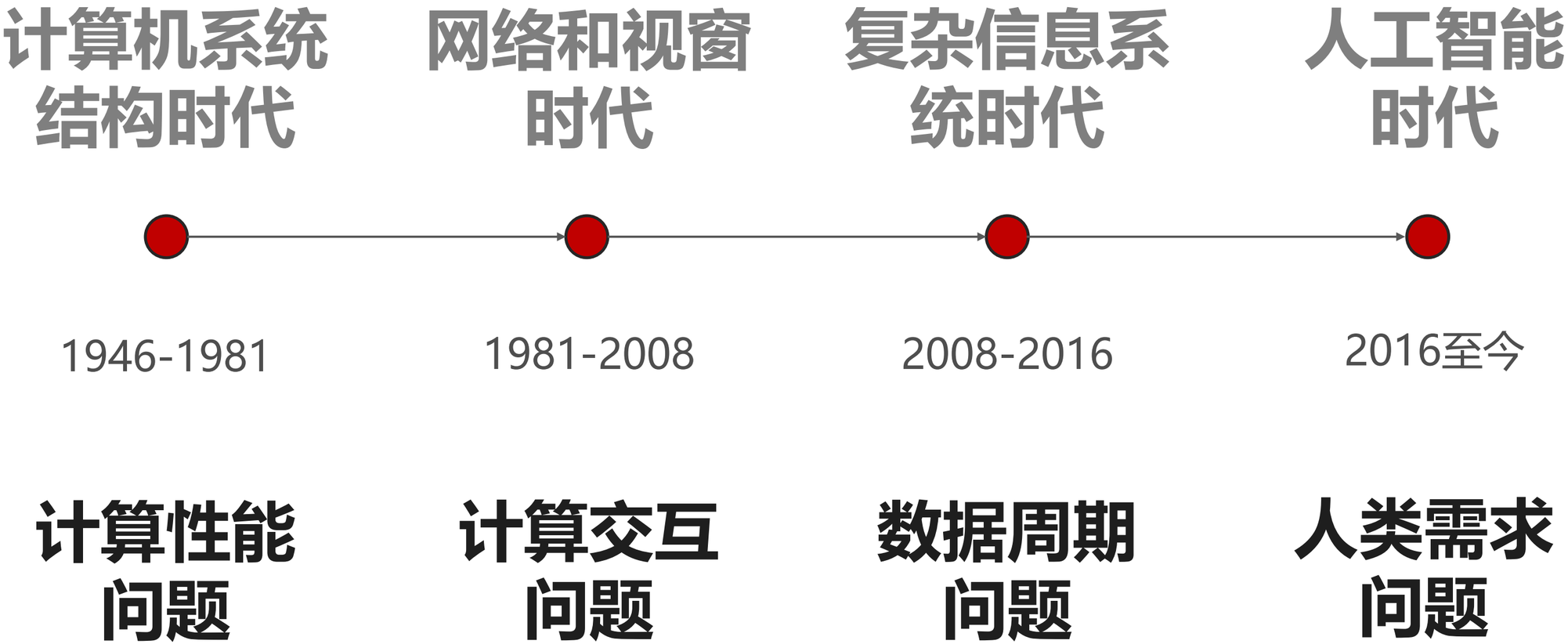
复杂信息系统时代

2008-2016

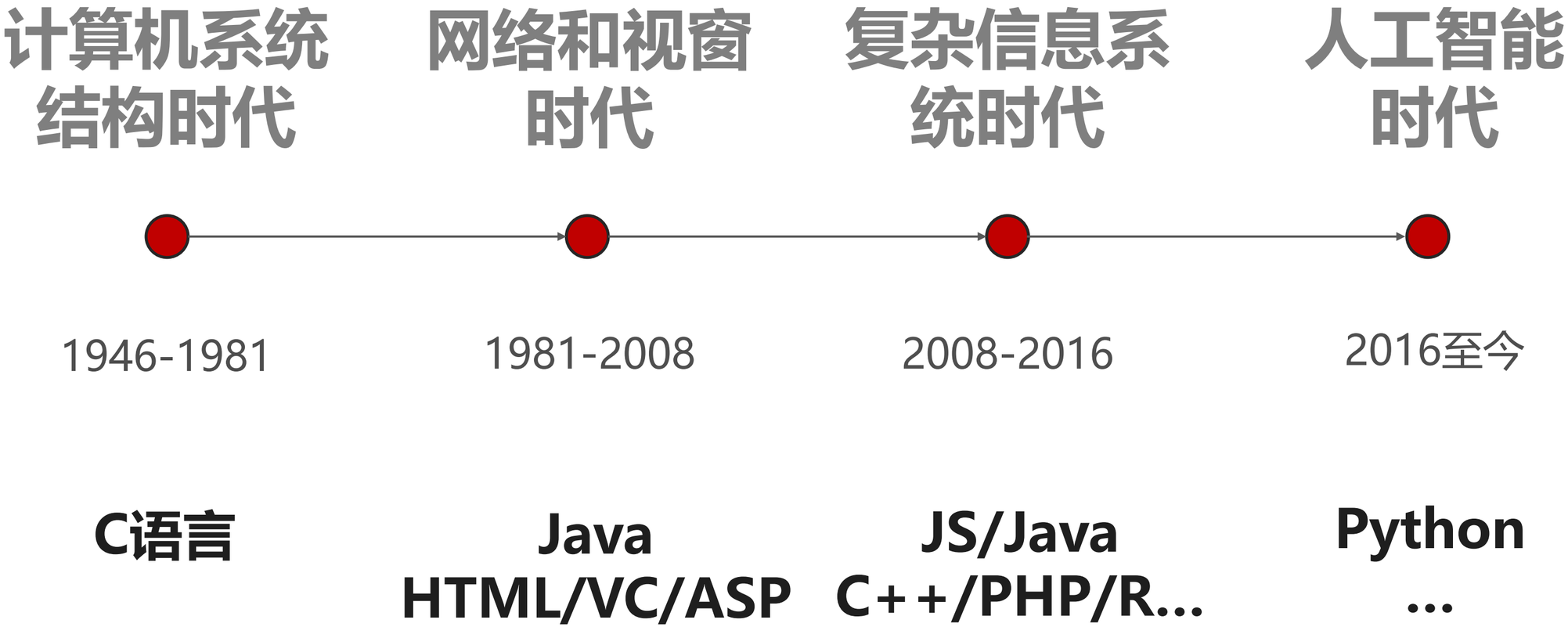
人工智能时代

2016至今

计算机技术演进阶段所解决的问题



计算机技术演进与主流编程语言变化



各编程语言的历史使命

编程语言	学习内容	语言本质	解决问题	适用岗位
C	指针、内存、数据类型	理解计算机系统结构	性能	系统底层开发
Java	对象、跨平台、运行时	理解主客体关系	跨平台	网络后台及APP开发
C++	对象、多态、继承	理解主客体关系	大规模程序	应用类特定方向
VB/VC	对象、按钮、文本框	理解交互逻辑	桌面交互	不确定
Python	编程逻辑、第三方库	理解问题求解	各类问题	大数据/人工智能等

各编程语言的历史使命

各编程语言使命不同，Python是计算时代演进的选择！

再看计算条件...

- 计算机性能已经不是解决一般问题的瓶颈
- 大数据、云计算、物联网、信息安全、人工智能等需求爆发
- 计算领域核心矛盾转化为产量、效率和方法的不足
- 在2018年及以后，面对应用导向计算需求，该使用什么编程语言?

Python语言

1. Python是通用语言
2. Python是脚本语言
3. Python是胶水语言
4. Python是跨平台语言
5. Python是多模型语言

Guido van Rossum

Python语言创立者

Python快速入门

编译和解释

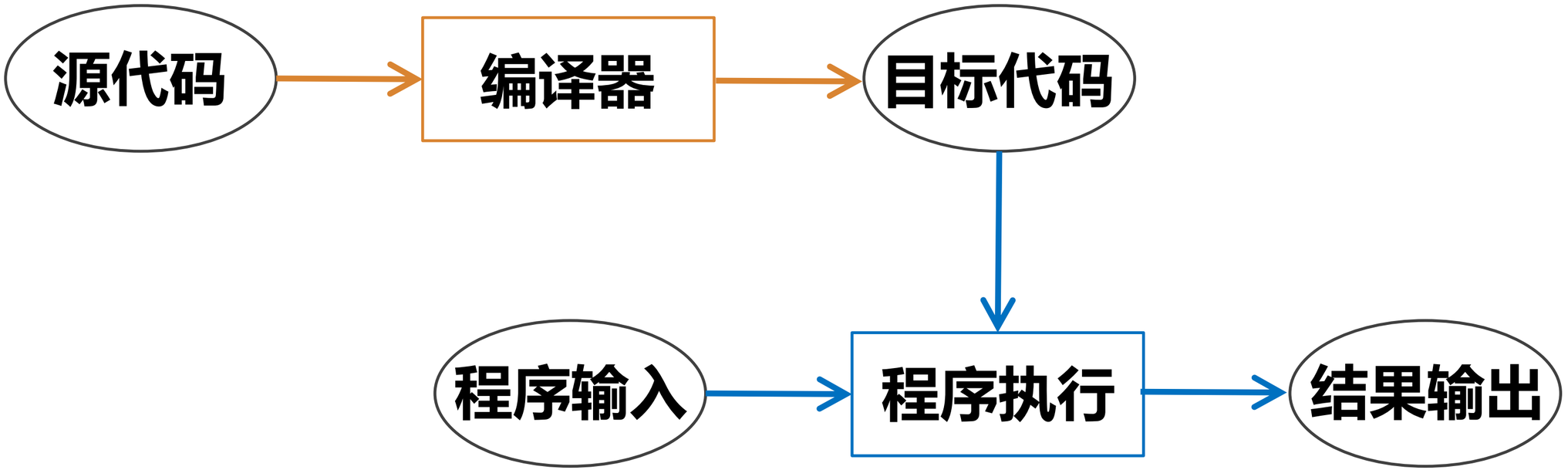
编程语言的执行方式

1 编译执行

将源代码一次性转换成目标代码的过程

编程语言的执行方式

1 编译执行



编程语言的执行方式

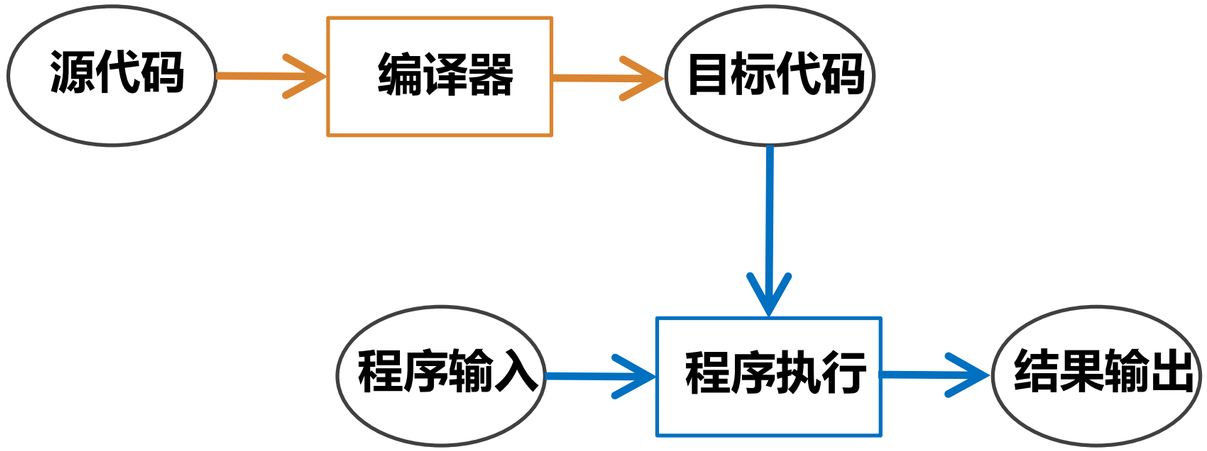
2 解释执行

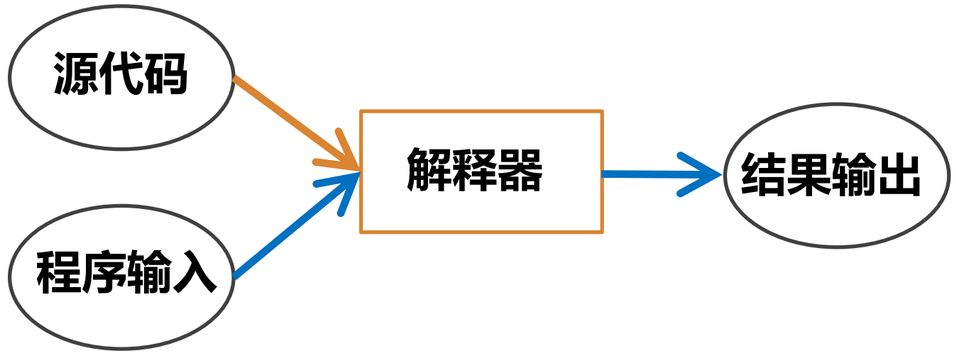
将源代码逐条转换成目标代码同时逐条运行的过程

编译和解释

1 编译执行

2 解释执行





编译和解释

1. 编译执行
2. 解释执行

- 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

1. **静态语言**：编译执行
2. **脚本语言**：解释执行

- 静态语言：C、C++、Java、Verilog ...
- 脚本语言：PHP、JavaScript、HTML、Python ...

静态语言的优点

1. 编译过程没有时间限制，优化更充分，执行速度更快
2. 编译后程序可以在同系列系统中直接执行，不需要执行环境

脚本语言的优点

1. 执行过程需要源代码，程序维护更灵活
2. 通过在不同系统中配置执行环境，可以实现源代码的跨平台执行

静态语言和脚本语言

编译和解释是两种方式的区别，在完成计算需求方面结果一致

Python快速入门

Python语言特点

Python语言

1. **Python是通用语言**
2. Python是脚本语言
3. Python是胶水语言
4. Python是跨平台语言
5. Python是多模型语言

Guido van Rossum

Python语言创立者

Python语言特点：语法简洁

语法简洁

```
# 计算并输出1到100整数之和
s = 0
for i in range(1, 101):
    s += i
print(s)
```

```
# 计算并输出n的阶乘
def fact(n):
    if n == 1:
        return 1
    return n*fact(n-1)
print(fact(10))
```

Python语言特点：语法简洁

语法简洁

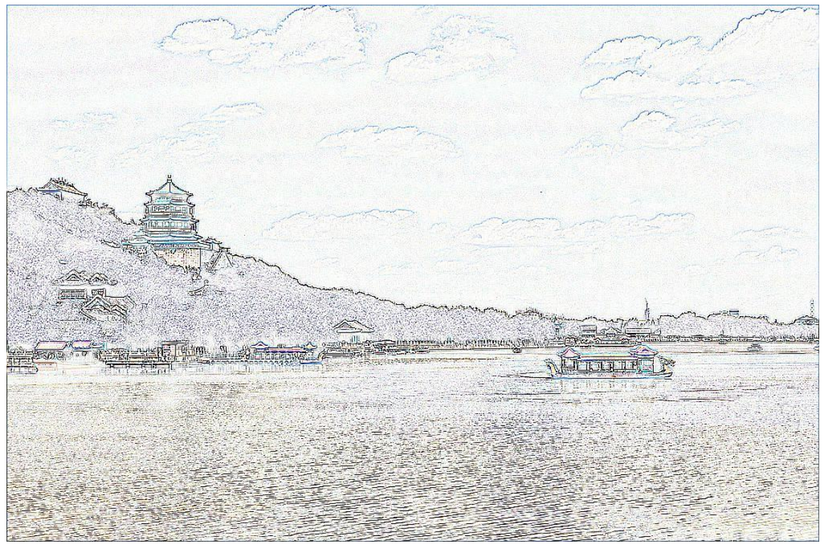
1. 没有分号、没有函数约束、没有大括号、没有begin/end
2. 没有类型声明、没有变量定义、没有指针

相同功能是C语言代码量的10%及以下

Python语言特点：开源生态

开源生态

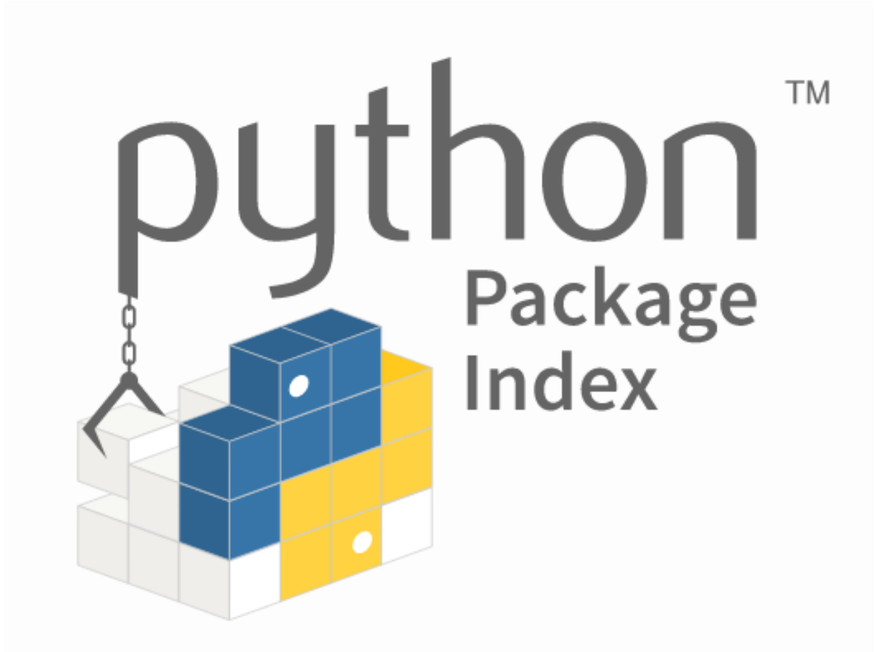
```
# 图像的轮廓获取
from PIL import Image
from PIL import ImageFilter
im = Image.open('Beijing.jpg')
om = im.filter(ImageFilter.CONTOUR)
om.save('BeijingContour.jpg')
```



Python语言特点：开源生态

开源生态

1. 超过14万个第三方库免费可用
2. 避免重复造轮子的先进理念



底层可封装C/C++等语言的代码，站在巨人肩膀上编程

Python语言特点

语法简洁　　**开源生态**

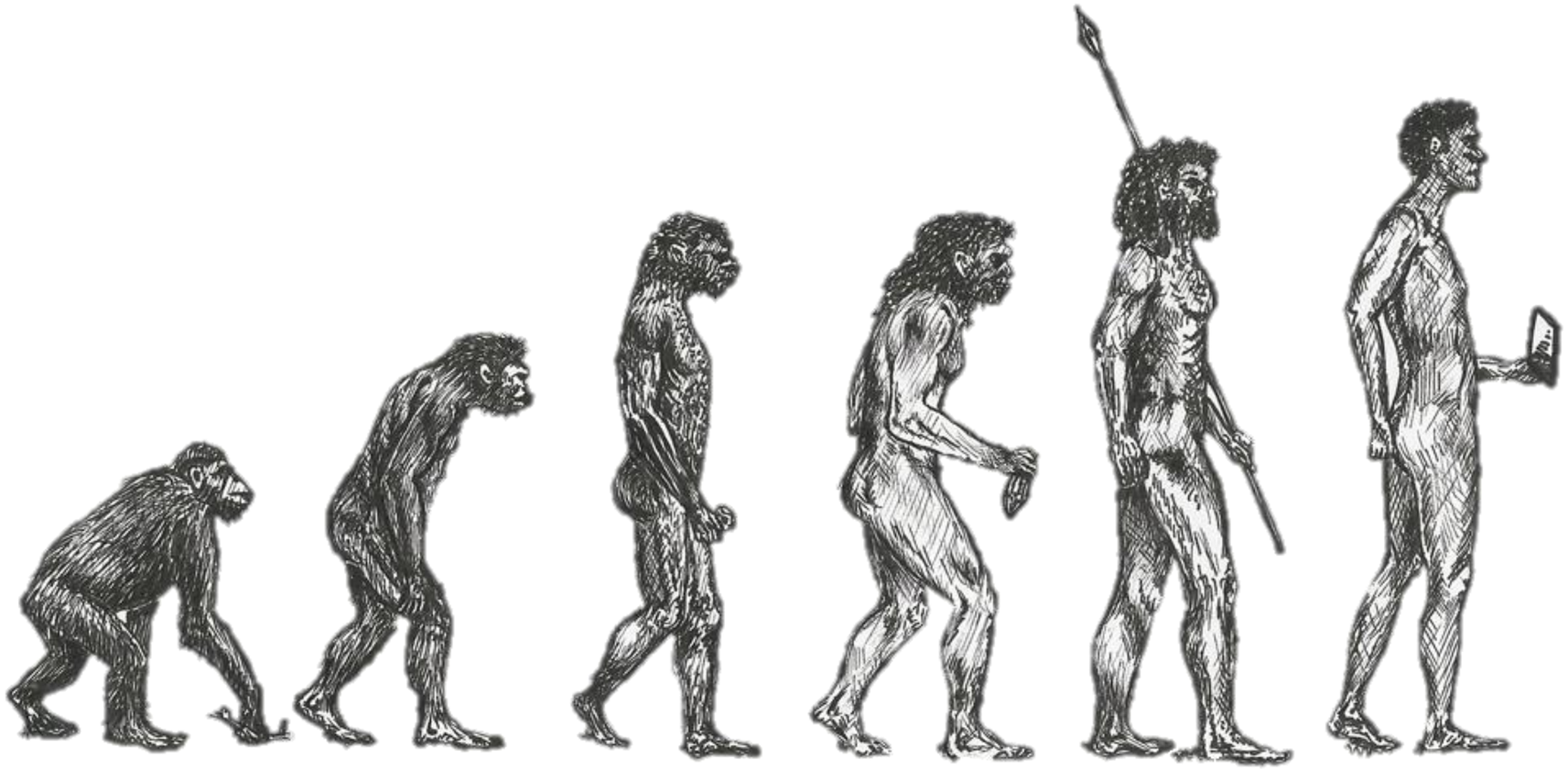
1. 通过简化语法抽象逻辑，语法简洁能提高10倍左右的编程产量

2. 通过功能模块复用集成，开源生态能再提高10倍左右的编程产量

Python能带来10倍以上的编程产量提升

Python语言特点

工具决定思维：关注工具变革的力量！



Python语言特点

>10x

Python快速入门

计算生态与超级语言

Python123

计算生态：自由软件时代

从开源运动说起...

Richard Stallman启动GNU项目

GNU通用许可协议诞生

1983

1989

计算生态：开源生态逐步建立

从开源运动说起...

Linus Torvalds发布了Linux内核

网景浏览器开源，产生了Mozilla

1991

1998

计算生态：计算生态深入演化

1983, Richard Stallman
大教堂模式

V.S.

1991, Linus Torvalds
集市模式

计算生态：定义

开源思想深入演化和发展，形成了计算生态

计算生态以开源项目为组织形式，充分利用“共识原则”和“社会利他”组织人员，在竞争发展、相互依存和迅速更迭中完成信息技术的更新换代，形成了技术的自我演化路径。

计算生态：三个特点

没有顶层设计、以功能为单位、具备三个特点

1. 竞争发展
2. 相互依存
3. 迅速更迭

Python计算生态

1. 以开源项目为形式，提供了超过14万个第三方库
2. 开源项目的建设经过野蛮生长和自然选择，高质量库众多
3. 第三方库之间相互关联使用，依存发展或逐级封装
4. Python社区庞大，新技术更迭迅速，已经成为计算生态主流

Python计算生态

API != 计算生态

Python计算生态

理解、运用和构建计算生态

编程语言的演进

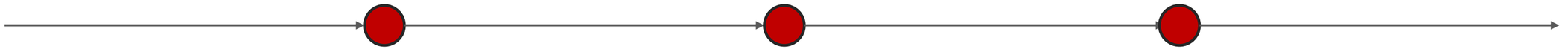
编程语言的种类...

机器语言	汇编语言	高级语言
11010010 00111011	add 2, 3, result	result = 2 + 3
CPU相关	CPU相关	CPU无关

编程语言的演进

编程语言的种类...

汇编语言	高级语言	超级语言
add 2, 3, result	result = 2 + 3	result = sum(2, 3)
CPU相关	CPU无关	计算生态

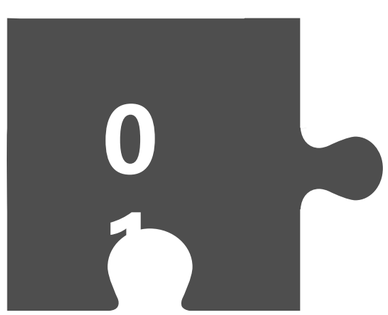


Python快速入门

单元小结

Python123

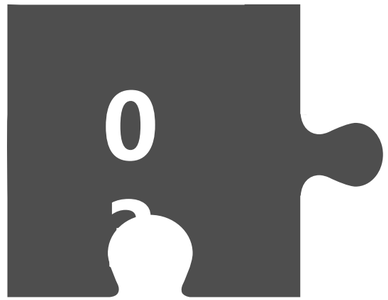
单元小结



计算机技术发展与编程语言演进

Why Python?

编译与解释



Python语言特点

What Python?

计算生态与超级语言

Thinking Python


Python 123
Thank you