网络编程与自动化

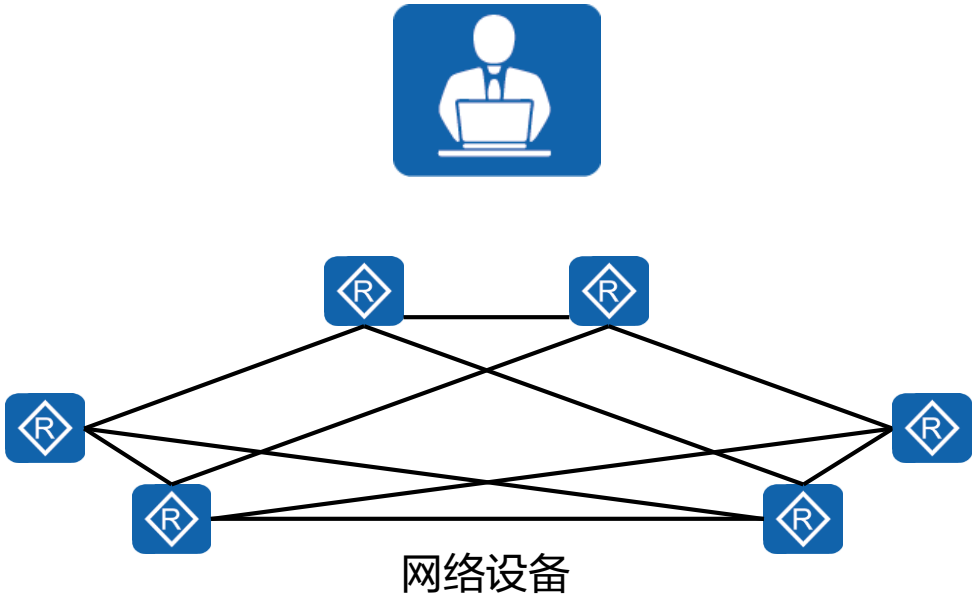
主讲人：施淼淼

目录

背景：传统网络运维困境

- 传统的网络运维工作需要网络工程师手动登录网络设备，人工查看和执行配置命令，肉眼筛选配置结果。这种严重依赖“人”的工作方式操作流程长，效率低下，而且操作过程不易审计。

设备多！操作烦琐！效率低！



网络设备

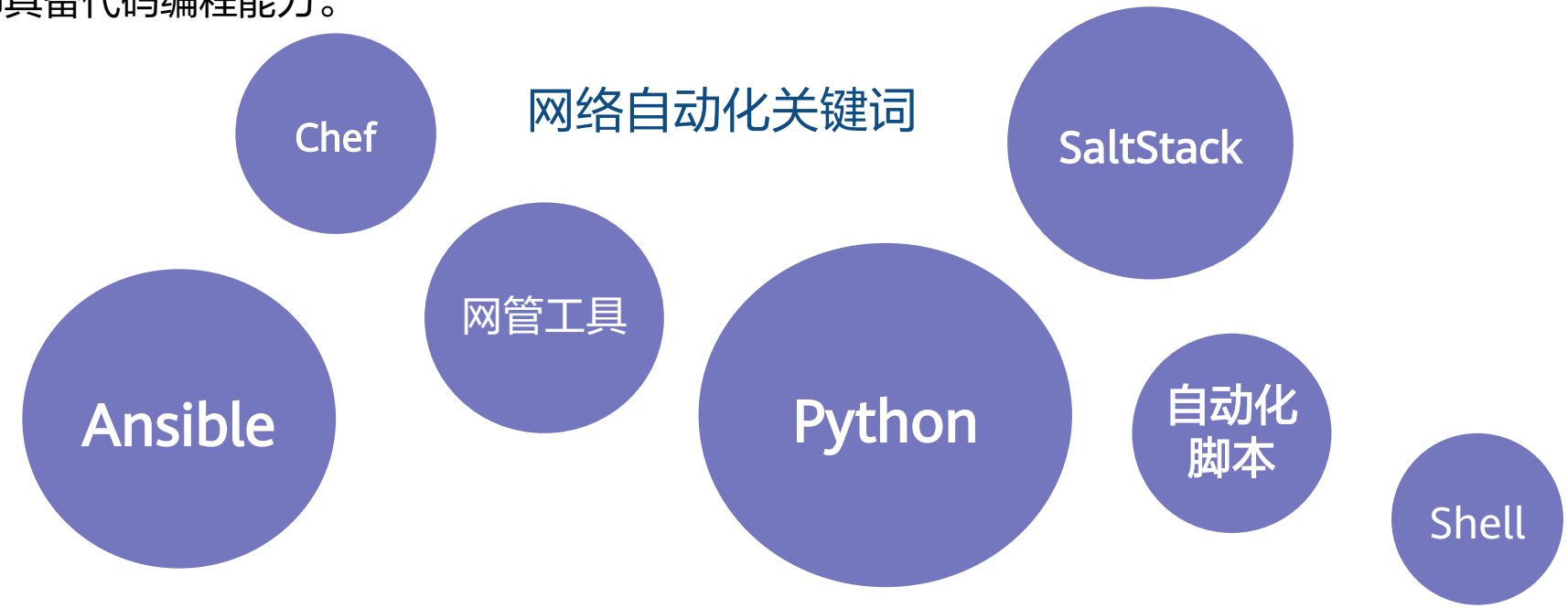
经典运维场景

在工作中你是否遇到过这样的场景：

1. 设备升级：现网有数千台网络设备，你需要周期性、批量性地对设备进行升级。
2. 配置审计：企业年度需要对设备进行配置审计。例如要求所有设备开启sTelnet功能，以太网交换机配置生成树安全功能。你需要快速地找出不符合要求的设备。
3. 配置变更：因为网络安全要求，需要每三个月修改设备账号和密码。你需要在数千台网络设备上删除原有账号并新建账号。

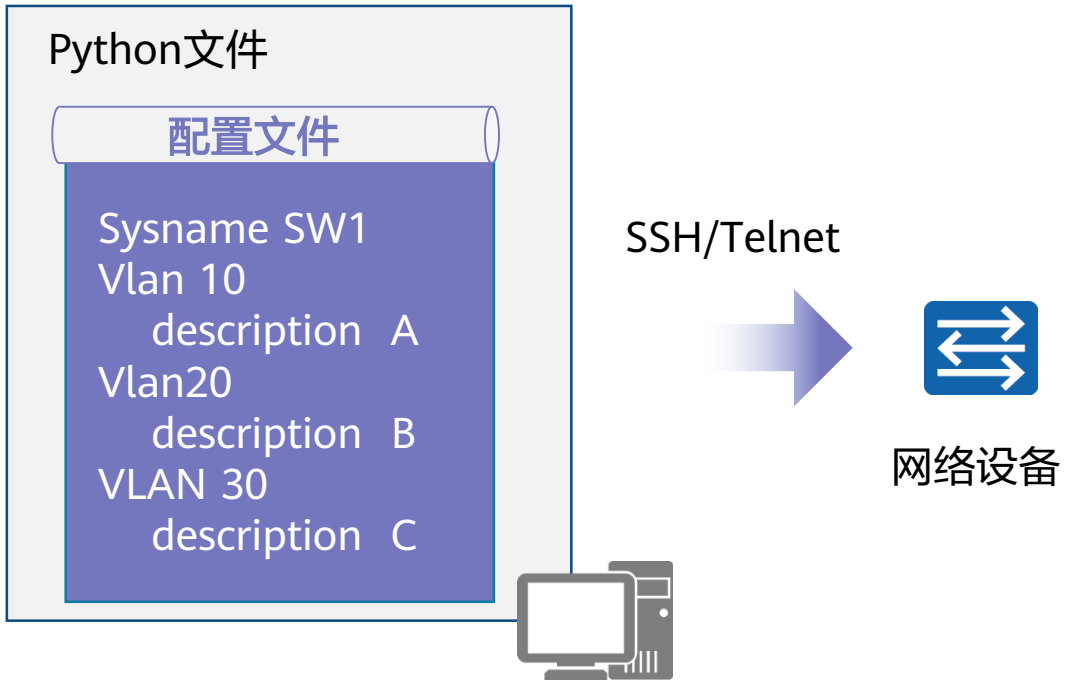
网络自动化

- 网络自动化，通过工具实现网络自动化地部署、运行和运维，逐步减少对“人”的依赖。这能够很好地解决传统网络运维的问题。
- 业界有很多实现网络自动化的开源工具，例如Ansible、SaltStack、Puppet、Chef等。从网络工程能力构建的角度考虑，更推荐工程师具备代码编程能力。



基于编程实现的网络自动化

- 近几年随着网络自动化技术的兴起，以Python为主的编程能力成为了网络工程师的新技能要求。
- Python编写的自动化脚本能够很好的执行重复、耗时、有规则的操作。



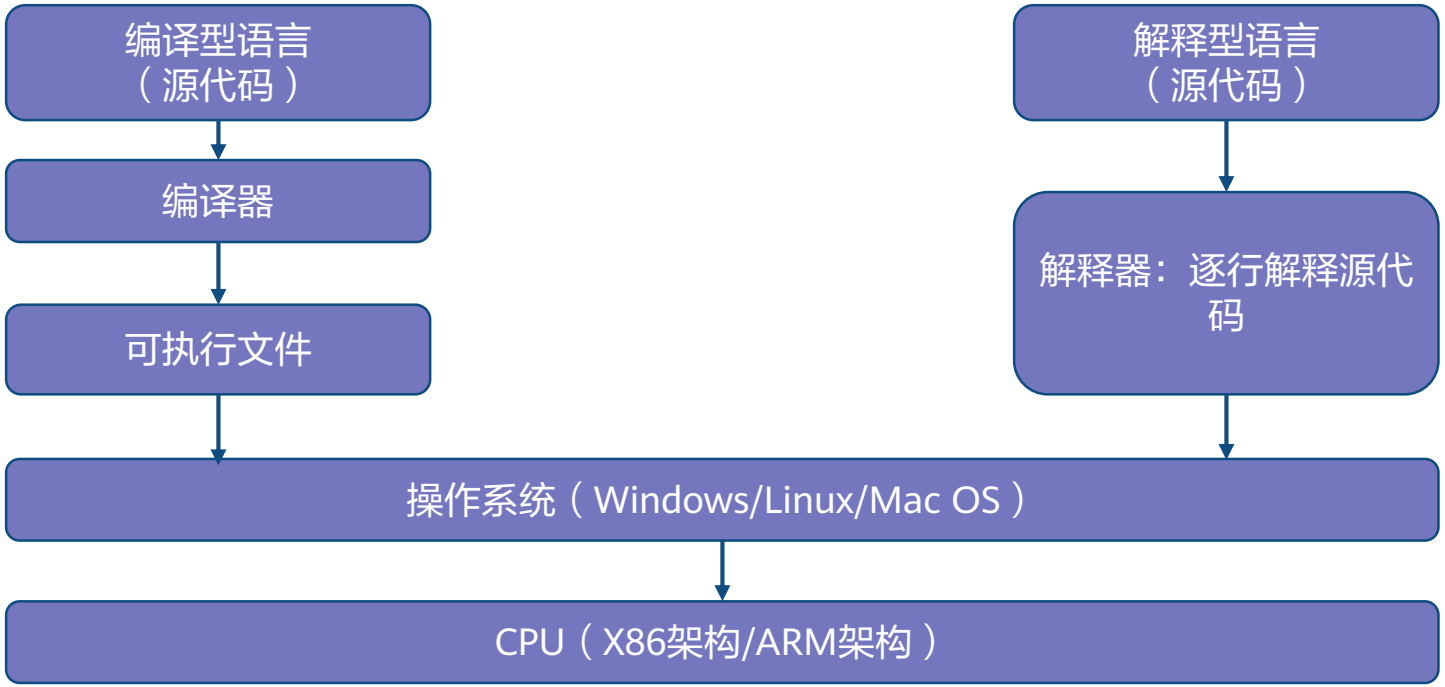
举例：Python实现设备自动化配置

- 网络自动化能做什么？最直观的一个网络自动化例子就是自动化配置设备。我们可以把这个过程分为两个步骤：编写配置文件和编写Python代码将配置文件推送到设备上。
- 首先用命令行方式写配置脚本，然后通过Telnet/SSH将它传到设备上运行。这种方式对于初学网络编程与自动化的网络工程师来说，比较容易理解。本章节主要介绍这种方式实现网络自动化。

目录

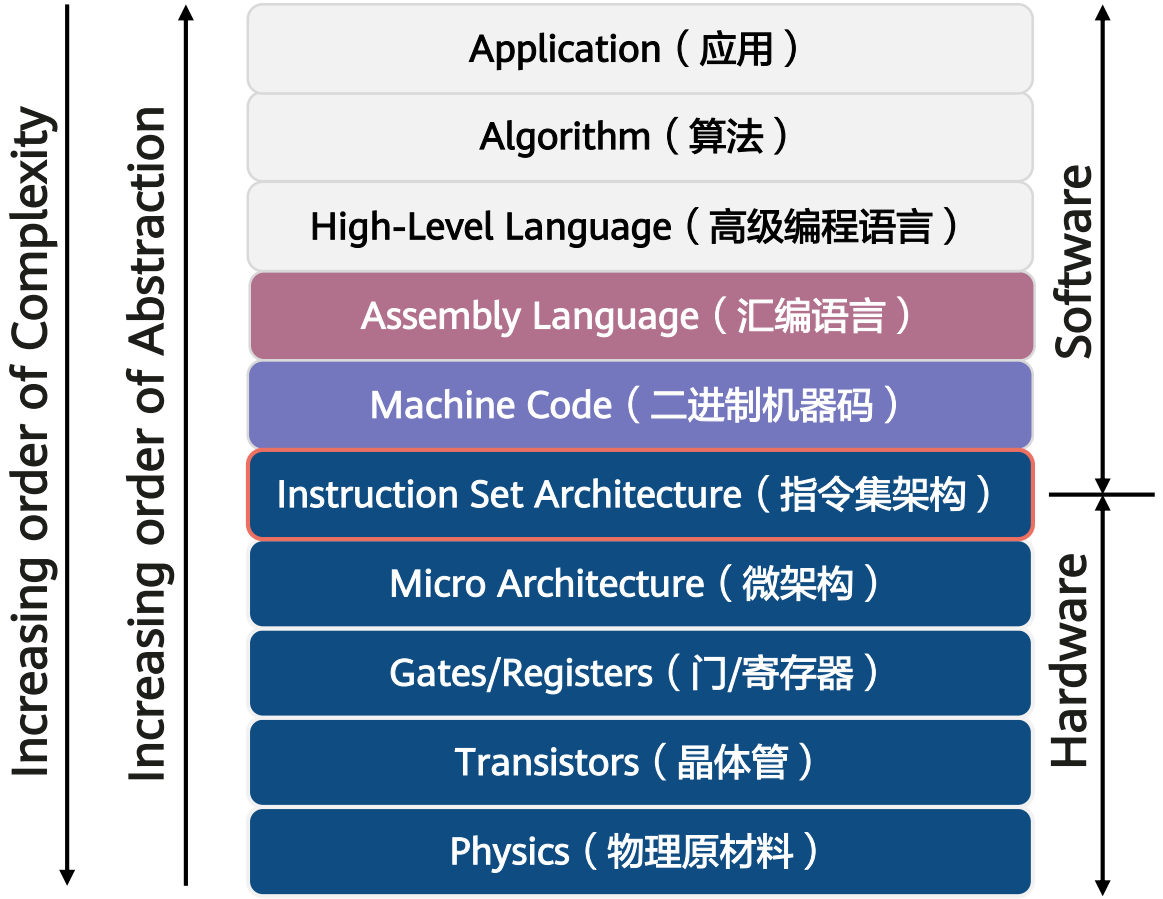
编程语言

- 编程语言（Programming Language），是一种用于编写计算机程序的语言，用于控制计算机的行为。
- 按照语言在执行之前是否需要编译区分，可以将编程语言分为需要编译的编译型语言（Compiled Language），不需要编译的解释型语言（Interpreted Language）。

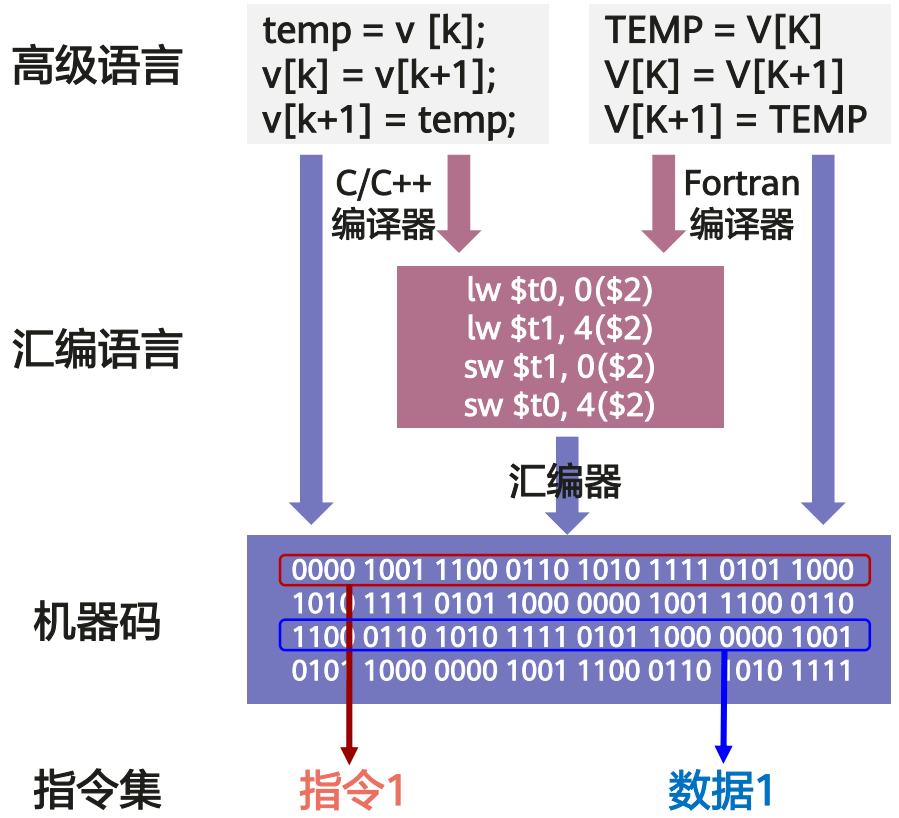


计算技术栈与程序执行过程

计算技术栈

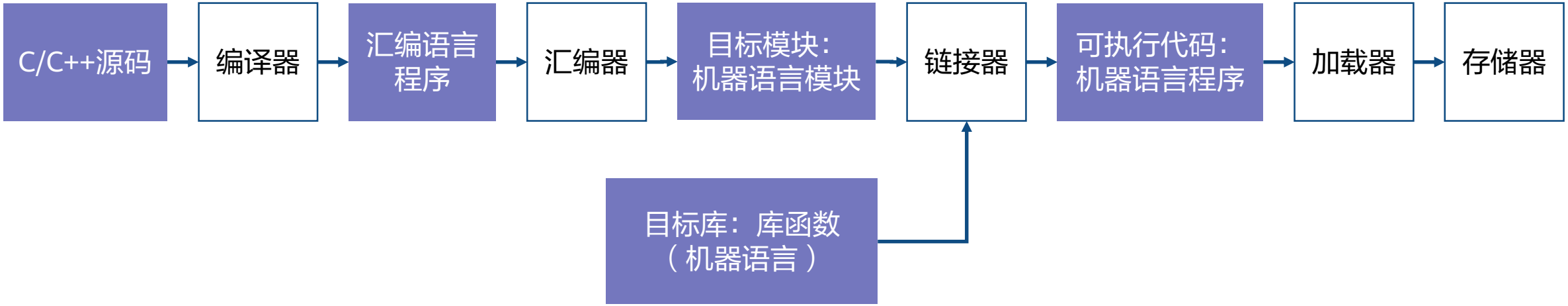


程序执行过程



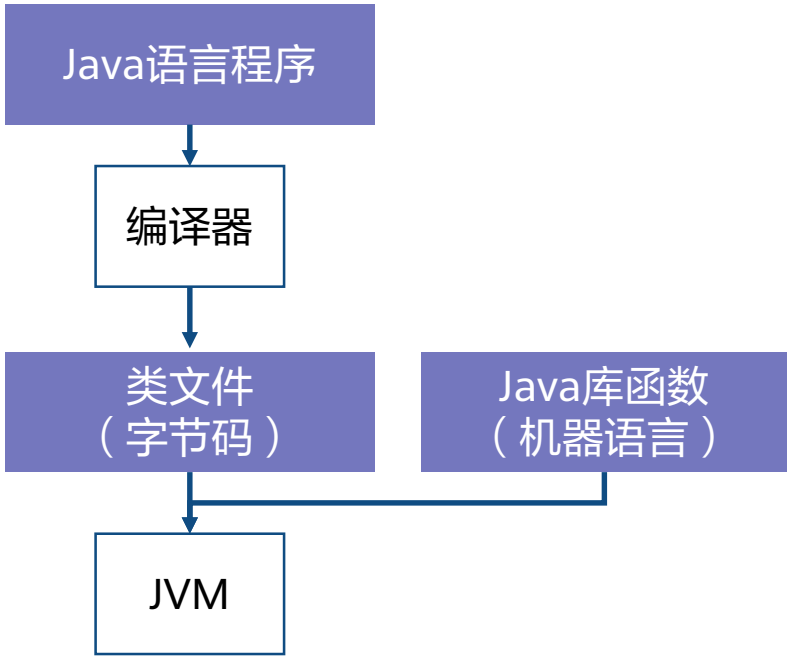
高级编程语言 - 编译型语言

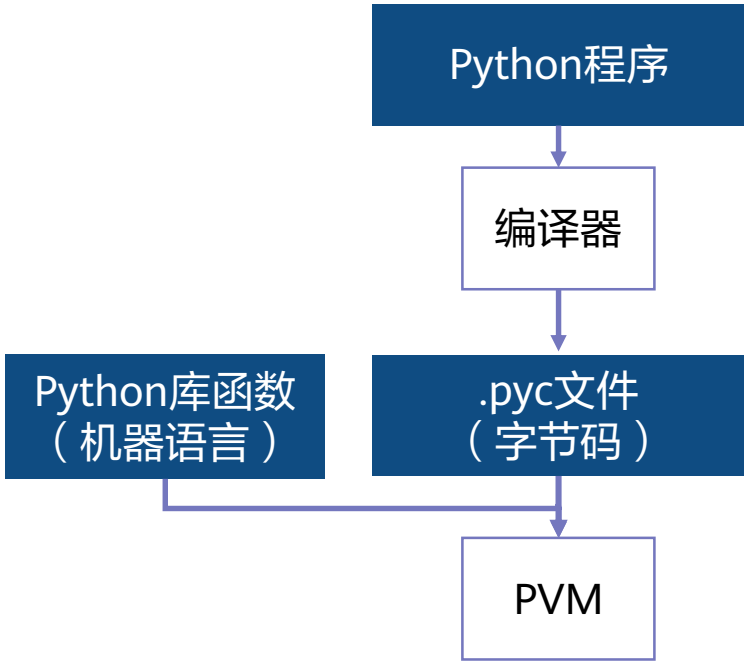
- **编译型语言**：编译型语言的程序在执行之前有一个编译过程，把程序编译成为机器语言的文件。运行时不需要重新翻译，直接使用编译的结果。典型的如C/C++/Go语言，都属于编译型语言。
- **从源码到程序的过程**：源码需要由编译器、汇编器翻译成机器指令，再通过链接器链接库函数生成机器语言程序。机器语言必须与CPU的指令集匹配，在运行时通过加载器加载到内存，由CPU执行指令。



高级编程语言 - 解释型语言

- **解释型语言**：解释型语言的程序不需要在运行前编译，在运行程序的时候才逐行翻译。典型的如Java/Python语言，都属于解释型语言。
- **从源码到程序的过程**：解释型语言的源代码由编译器生成字节码，然后再由虚拟机（JVM/PVM）解释执行。虚拟机将不同CPU指令集的差异屏蔽，因此解释型语言的可移植性相对较好。





什么是Python?

- Python是一门完全开源的高级编程语言。它的作者是Guido Van Rossum。

Python的优点：

- Python拥有优雅的语法、动态类型具有解释性质。能够让学习者从语法细节的学习中抽离，专注于程序逻辑。
- Python同时支持面向过程和面向对象的编程。
- Python拥有丰富的第三方库。
- Python可以调用其他语言所写的代码，又被称为胶水语言。

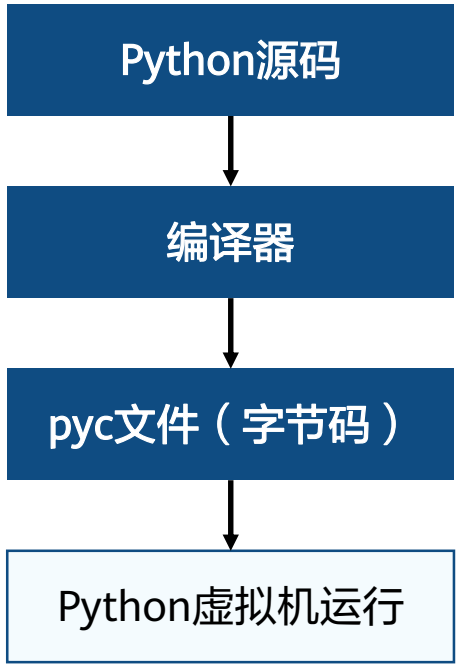
Python的缺点：

- 运行速度慢。Python是解释型语言，不需要编译即可运行。代码在运行时会逐行地翻译成CPU能理解的机器码，这个翻译过程非常耗时。

由于Python具有非常丰富的第三方库，加上Python语言本身的优点，所以Python可以在非常多的领域内使用：人工智能、数据科学、APP、自动化运维脚本等。

Python代码执行过程

Python程序编译运行的过程



操作过程

1、在操作系统上安装Python和运行环境。

2、编写Python源码。

3、编译器运行Python源码，编译生成pyc文件（字节码）。

4、Python虚拟机将字节码转换为机器语言。

5、硬件执行机器语言。

初识Python代码 - 交互式运行

- Python有两种运行方式，交互式运行和脚本式运行。
- 交互式编程不需要创建脚本文件，是通过 Python 解释器的交互模式编写代码。

1. Input --
2. Output --
3. Input --
4. Input --
5. Input --
6. Output --

```
C:\Users\Richard>python
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)] ::
Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("hello world")
hello world
>>> a = 1
>>> b = 2
>>> print ( a + b )
3
>>>
```

初识Python代码 - 脚本式运行

- 脚本模式里的代码可以在各种Python编译器或者集成开发环境上运行。例如Python自带的IDLE、Atom、Visual Studio、Pycharm和Anaconda等。

demo.py文件

```
print ("hello world")
a = 1
b = 2
print ( a + b )
```

```
1. Input    --  C:\Users\Richard>python demo.py
2. Output   --  hello world
3. Output   --  3
```

1 编写Python脚本文件(.py)

2 执行脚本文件

Python编码规范

- 编码规范是使用Python编写代码时应遵守的命名规则、代码缩进、代码和语句分割方式等。良好的编码规范有助于提高代码的可读性，便于代码的维护和修改。
- 例如分号、圆括号、空行和空格的使用规范建议如下：

分号

- Python程序允许在行尾添加分号，但是不建议使用分号隔离语句。
- 建议每条一句单独一行。

空行

- 不同函数或语句块之间可以使用空格来分隔。用以区分两段代码，提高代码可读性。

圆括号

- 圆括号可用于长语句的续行。一般不使用不必要的括号。

空格

- 不建议在括号内使用空格。
- 对于运算符，可以按照个人习惯决定是否在两侧加空格。

Python编码规范 - 标识符命名

- Python标识符用于表示常量、变量、函数以及其他对象的名称。
- 标识符通常由字母、数字和下划线组成，但不能以数字开头。标识符大小写敏感，不允许重名。如果标识符不符合规则，编译器运行代码时会输出SyntaxError语法错误。

1. 数值赋值 --
2. 数值赋值 --
3. 字符串赋值 --
4. 数值赋值 --
5. 错误的标识符 --

```
User_ID = 10
user_id = 20
User_Name = 'Richard'
Count = 1 + 1
4_passwd = "Huawei"
```

```
print ( User_ID )
print ( user_id )
print ( User_Name )
print ( Count )
print ( 4_passwd )
```

print ()为Python内置的函数，作用是输出括号内的内容。

问题：右侧print的运行结果是?

Python编码规范 - 代码缩进

- 在Python程序中，代码缩进代表代码块的作用域。如果一个代码块包含两个或更多的语句，则这些语句必须具有相同的缩进量。对于Python而言代码缩进是一种语法规则，它使用代码缩进和冒号来区分代码之间的层次。
- 编写代码时候，建议使用4个空格来生成缩进。如果程序代码中使用了错误的缩进，则会在运行中发出IndentationError错误信息。

正确缩进 --

正确缩进 --

错误缩进 --

```
if True:
    print (“Hello”)
else:
    print (0)

a = “Python”
    print (a)
```

Python编码规范 - 使用注释

- 注释就是在程序中添加解释说明，能够增强程序的可读性。在Python程序中，注释分为单行注释和多行注释。
- 单行注释以 # 字符开始直到行尾结束。
- 多行注释内容可以包含多行，这些内容包含在一对三引号内（'''...'''或者"""..."""）。

单行注释 --

多行注释 --

```
#将字符串赋值给a
a = “Python”
print (a)

“””
运行输出结果为Python
“””
```

Python编码规范 - 源码文件结构

- 一个完整的Python源码文件一般包含几个组成部分：解释器和编码格式声明、文档字符串、模块导入和运行代码。
- 如果会在程序中调用标准库或其他第三方库的类时，需要先使用import或from... import语句导入相关的模块。导入语句始终在文件的顶部。在模块注释或文档字符串(docstring)之后。

解释器声明	--	#!/usr/bin/env python
编码格式声明	--	#-*- coding:utf-8 -*-
模块注释或文档字符串	--	"""本文档的说明（ docstring ） 本文档作用是... """
导入模块time	--	import time
运行代码	--	...

Python的函数与模块

- 函数(Function)是组织好的、可重复使用的一段代码。它能够提高程序的模块化程度和代码利用率。函数使用关键字 def 定义。
- 模块(Module)是一个保存好的Python文件。模块可以由函数或者类组成。模块和常规Python程序之间的唯一区别是用途不同：模块用于被其他程序调用。因此，模块通常没有main函数。

demo.py文件

```
def sit():      #定义函数
    print ('A dog is now sitting')

sit()           #调用函数
```

运行结果：

```
A dog is now sitting.
```

1 编写Python文件(.py)

test.py文件

```
import demo     #导入模块

demo.sit()      #调用函数
```

运行结果：

```
A dog is now sitting.
A dog is now sitting.
```

2 调用模块

Python的类与方法

- 类(Class)是用来描述具有一类相同的属性和方法的集合。类的定义使用关键字 class。
- 被实例化的类的”函数”被称作方法(Method)。类定义方法时候必须携带 self 关键字，它表示类的实例本身。

demo.py文件

```
class Dog():      # 定义类
    def sit(self):  # 定义方法
        print(“A dog is now sitting.")

Richard = Dog()    #实例化类
print (type(Richard.sit)) #实例化后类型为方法
print (type(Dog.sit))     #类型为函数
```

运行结果：

```
<class 'method'>
<class 'function'>
```

1 编写Python文件(.py)

test.py文件

```
import demo

demo.Dog.sit
```

运行结果：

```
A dog is now sitting.
<class 'method'>
<class 'function'>
```

2 调用模块

telnetlib介绍

- telnetlib是Python标准库中的模块。它提供了实现Telnet功能的类telnetlib.Telnet。这里通过调用telnetlib.Telnet类里的不同方法实现不同功能。

导入telnetlib模块Telnet类 --
Telnet连接到指定服务器上 --
调用read_all()方法 --

```
from telnetlib import Telnet
tn = Telnet(host=None, port=0[, timeout])
tn.read_all()
```

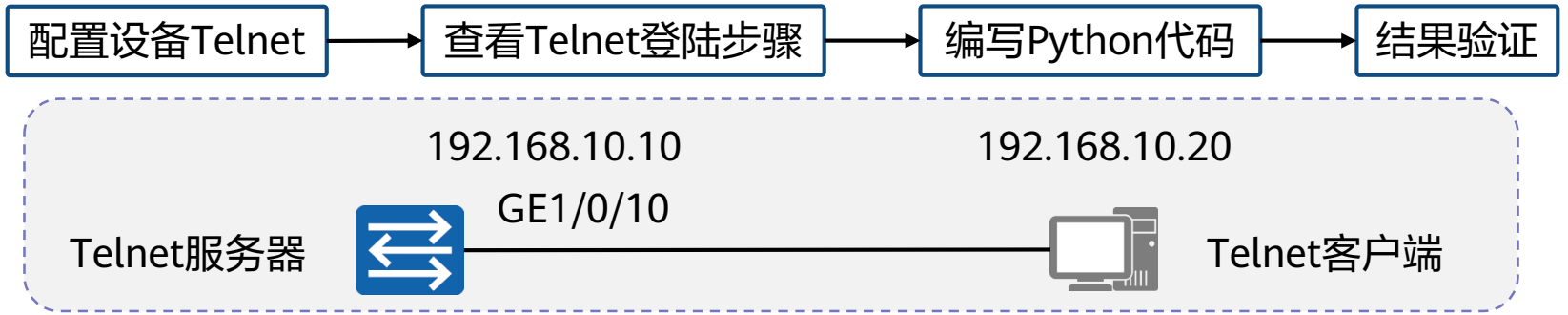
方法	功能
Telnet.read_until（expected, timeout=None）	读取直到给定的字符串expected或超时秒数。
Telnet.read_all ()	读取所有数据直到EOF(End Of File)。阻塞直到连接关闭。
Telnet.read_very_eager()	读取从上次IO阻断到现在所有的内容，返回字节串。连接关闭或者没有数据时触发EOFError异常。
Telnet.write(buffer)	写入数据。在套接字(Socket)上写一个字节串，加倍任何IAC(Interpret As Command)字符。
Telnet.close()	关闭连接。

目录

案例：使用telnetlib登陆设备

案例描述：

- 现有一台网络设备作为Telnet服务器，需要实现使用Python telnetlib作为Telnet客户端登录此设备。

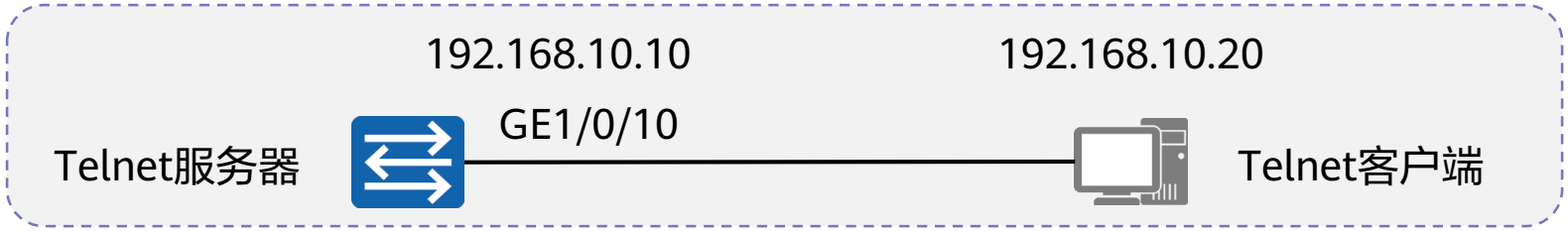


实现过程分为四个步骤：

- 配置设备Telnet服务。
- 手动验证和查看Telnet登录步骤，作为代码实现的参考。
- 编写和运行Python代码。
- 验证结果。

案例：使用telnetlib登陆设备

配置设备Telnet → 查看Telnet登陆步骤 → 编写Python代码 → 结果验证



配置设备接口地址：

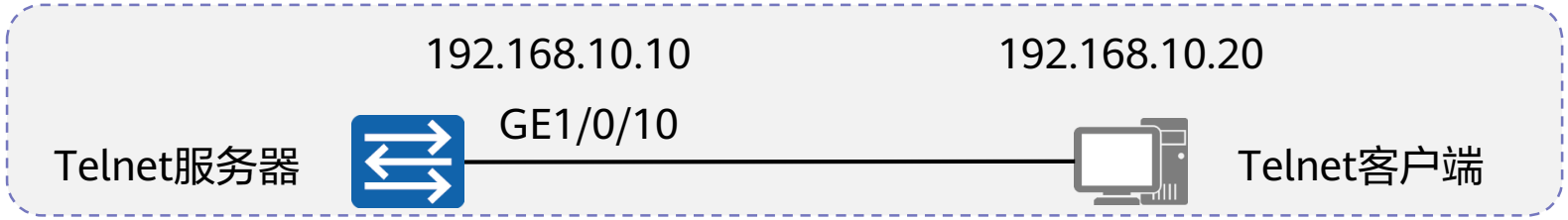
```
[Huawei] interface GE 1/0/0
[Huawei -GE1/0/0] ip add 192.168.10.10 24
[Huawei -GE1/0/0] quit
```

配置设备Telnet服务：

```
[Huawei] user-interface vty 0 4
[Huawei-ui-vty0-4] authentication-mode password
[Huawei-ui-vty0-4] set authentication password simple Huawei@123
[Huawei-ui-vty0-4] protocol inbound telnet
[Huawei-ui-vty0-4] user privilege level 15
[Huawei-ui-vty0-4] quit
[Huawei] telnet server enable
```

案例：使用telnetlib登陆设备

配置设备Telnet → 查看Telnet登陆步骤 → 编写Python代码 → 结果验证



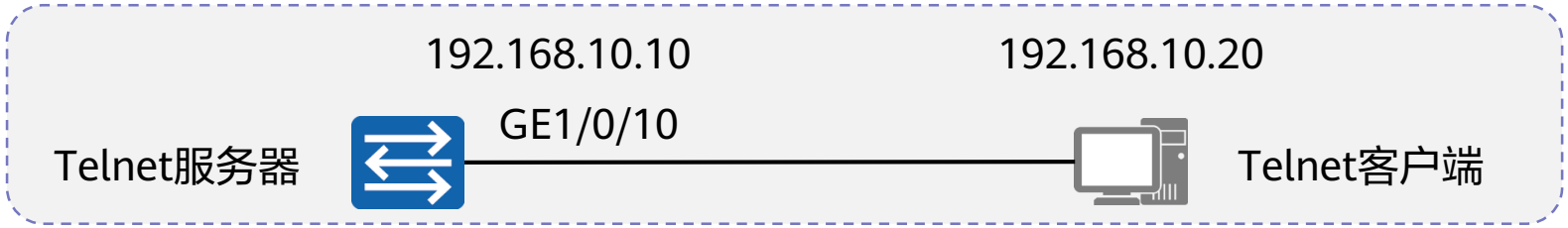
Telnet登录操作：

1. 登录命令 -- C:\Users\Richard>telnet 192.168.10.10
   回显信息 -- Login authentication

2. 输入密码 -- Password:
   回显信息 -- Info: The max number of VTY users is 5, and the number of current VTY users on line is 1.
   The current login time is 2020-01-15 21:12:57.
   <Huawei>

案例：使用telnetlib登陆设备

配置设备Telnet → 查看Telnet登陆步骤 → **编写Python代码** → 结果验证



导入模块	--	import telnetlib
定义登录设备IP	--	host = '192.168.10.10'
定义登录设备密码	--	password = 'Huawei@123'
Telnet登录到主机	--	tn = telnetlib.Telnet(host)
读取直到回显信息为”Password:”	--	tn.read_until(b’Password:’)
输入编码为ASCII的密码并换行	--	tn.write(password.encode('ascii') + b"\n")
输出读取直到到”<Huawei>”的信息	--	print (tn.read_until(b'<Huawei>').decode('ascii’))
关闭Telnet连接	--	tn.close()

案例：运行结果对比

配置设备Telnet → 查看Telnet登陆步骤 → 编写Python代码 → **结果验证**

手动Telnet登录结果：

```
C:\Users\Richard>telnet 192.168.10.10
Login authentication

Password:
Info: The max number of VTY users is 5, and the number of current VTY users on line is 1.
The current login time is 2020-01-15 21:12:57.
<Huawei>
```

Python代码运行结果：

```
#编译器运行Python代码
Info: The max number of VTY users is 5, and the number
      of current VTY users on line is 1.
      The current login time is 2020-01-15 22:12:57.
<Huawei>
```

本章总结

- 网络自动化是通过工具实现网络自动化的部署、运行和运维，逐步减少对“人”的依赖。可以通过编程语言或者工具实现。
- Python是一门完全开源的高级编程语言，语法简单，容易学习。拥有丰富的标准库和第三方库，适用于网络工程领域。
- Python的telnetlib模块提供了实现Telnet功能的类telnetlib.Telnet。可以让您初窥网络编程与自动化世界！华为更多开放可编程内容请参考HCIP-Datacom！