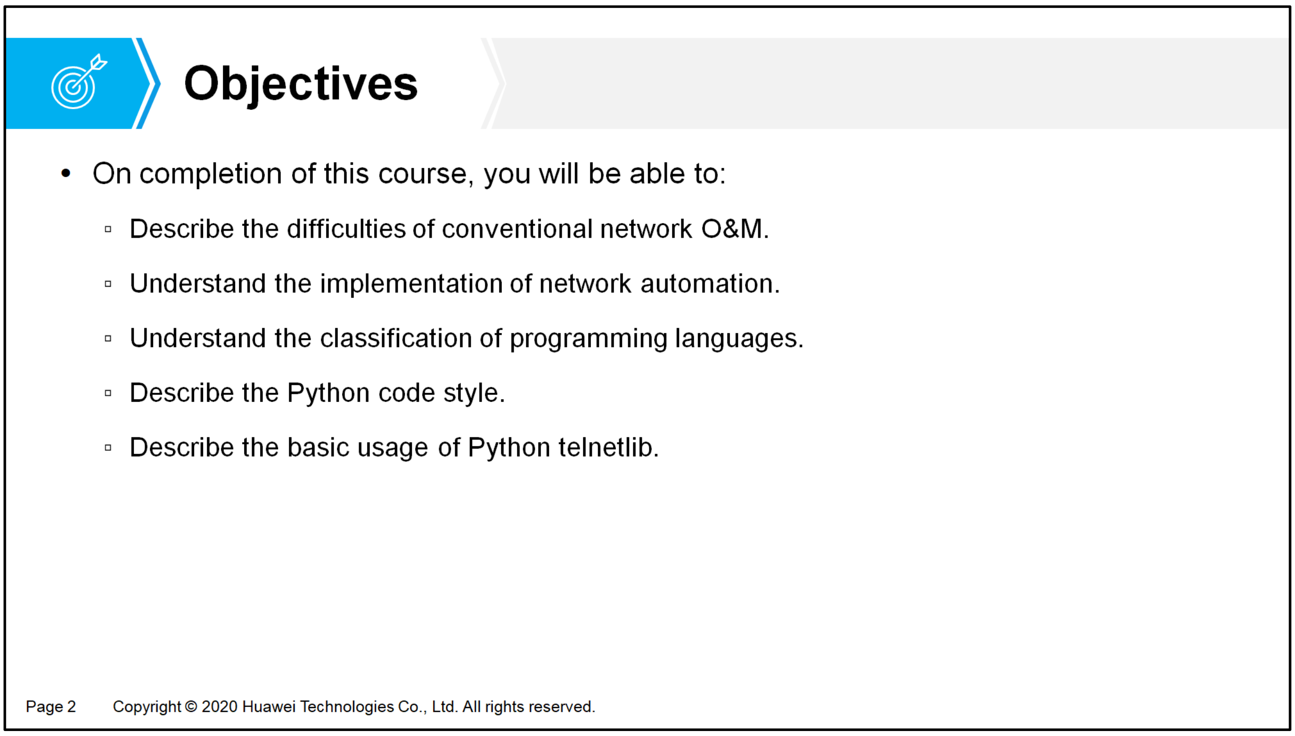
Fondamenti di Internet

Network Programmability and Automation

Prof. Gianluca Reali

Foreword

- New protocols, technologies, and delivery and O&M modes are emerging in the network engineering field. Conventional networks face challenges from new connection requirements, such as requirements for cloud computing and artificial intelligence (AI). Enterprises are also pursuing service agility, flexibility, and elasticity. Against this backdrop, network automation becomes increasingly important.
- Network programmability and automation is to simplify network configuration, management, monitoring, and operations for engineers and improve deployment and O&M efficiency. This course is to help network engineers understand Python programming and implement network automation.



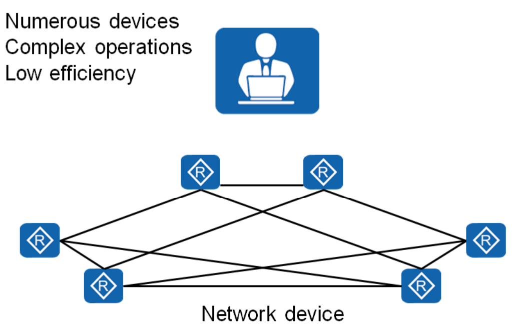
Objectives

- On completion of this course, you will be able to:
  - Describe the difficulties of conventional network O&M.
  - Understand the implementation of network automation.
  - Understand the classification of programming languages.
  - Describe the Python code style.
  - Describe the basic usage of Python telnetlib.

Contents

Background: Difficulties in Conventional Network O&M

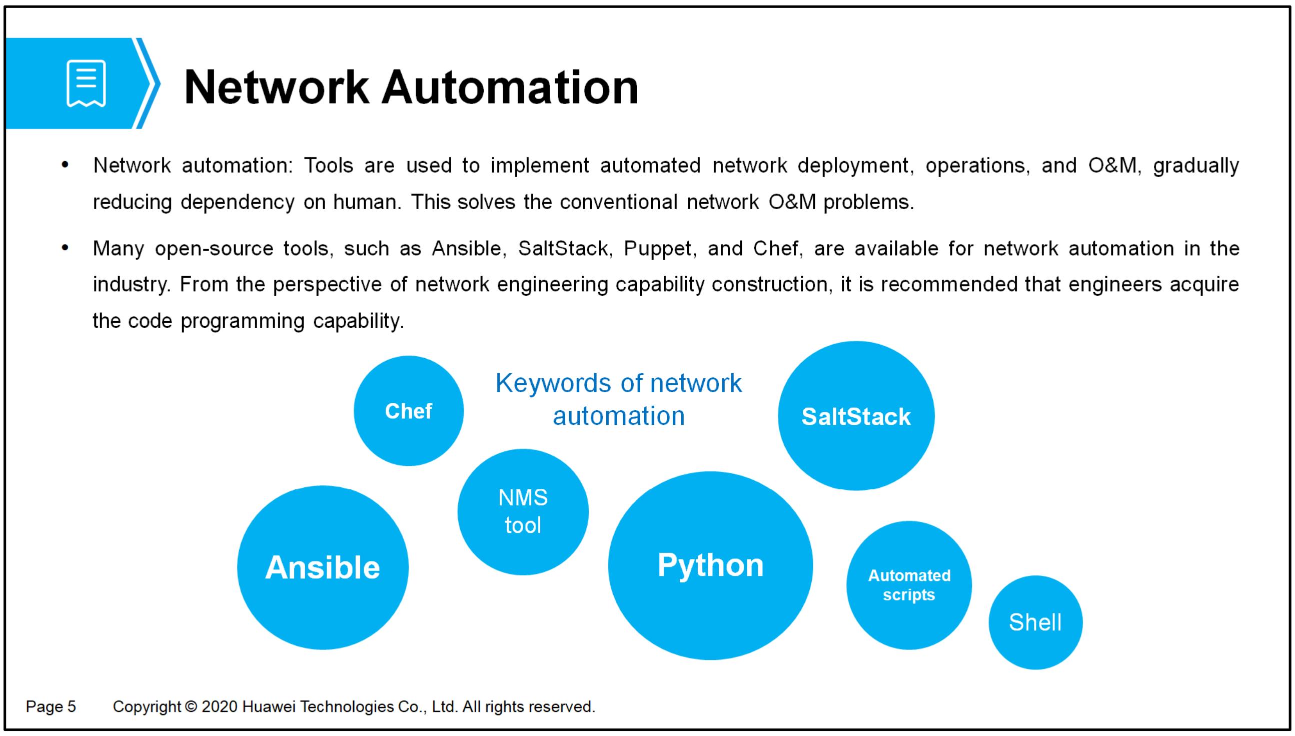
- Conventional network O&M requires network engineers to manually log in to network devices, query and execute configuration commands, and filter command output. This highly human-dependent working mode is time-consuming, inefficient, and difficult to audit.



Typical O&M Scenarios

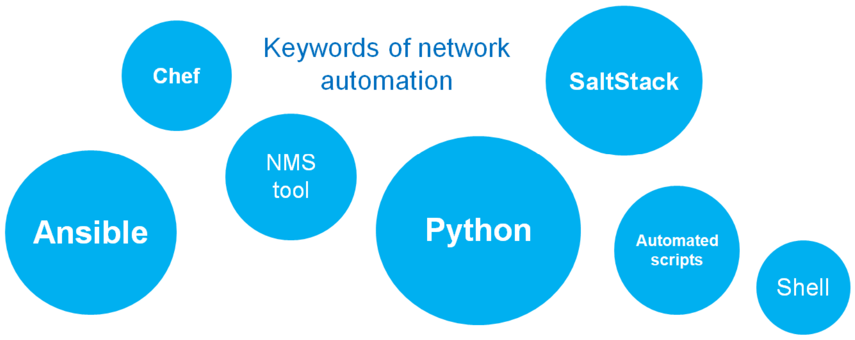
Are the following working scenes familiar to you?

1. Device upgrade: Thousands of network devices reside on a live network. You have to periodically upgrade the devices in batches.
2. Configuration audit: An enterprise needs to audit the configuration of devices every year. For example, the enterprise requires that STelnet be enabled on all devices and spanning tree security be configured on Ethernet switches. In this case, you have to quickly find out the devices that do not meet the requirements.
3. Configuration change: Due to network security requirements, device accounts and passwords need to be changed every three months. You have to delete the original account and create an account on thousands of network devices.



Network Automation

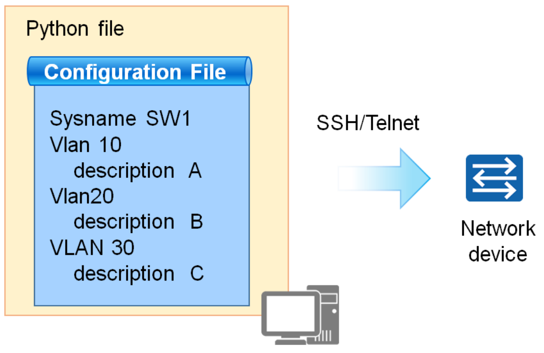
- Network automation: Tools are used to implement automated network deployment, operations, and O&M, gradually reducing dependency on human. This solves the conventional network O&M problems.
- Many open-source tools, such as Ansible, SaltStack, Puppet, and Chef, are available for network automation in the industry. From the perspective of network engineering capability construction, it is recommended that engineers acquire the code programming capability.



- Many network automation tools in the industry, such as Ansible, SaltStack, Puppet, and Chef, are derived from open-source tools. It is recommended that network engineers acquire the code programming capability.

Programming-based Network Automation

- In recent years, with the emergence of network automation technologies, Python-based programming capabilities have become a new skill requirement for network engineers.
- Automation script written in Python can execute repeated, time-consuming, and rule-based operations.

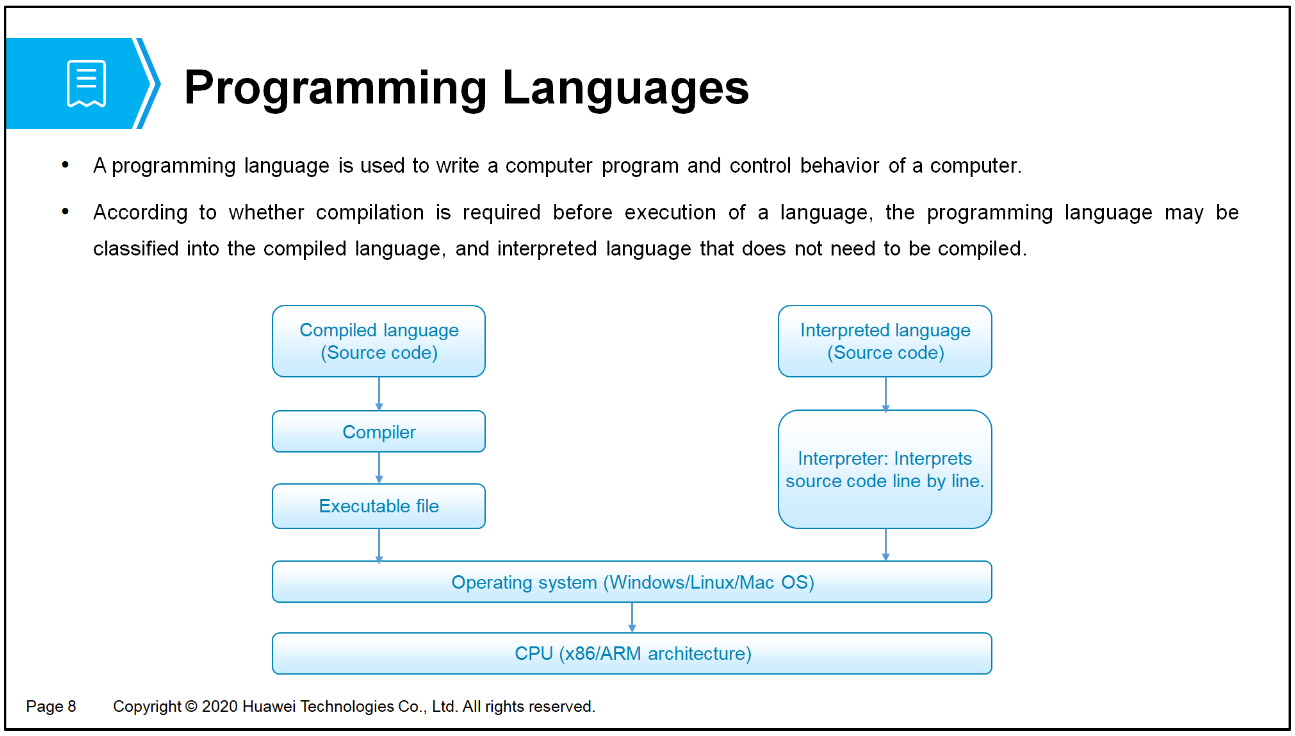


```
Sysname SW1
Vlan 10
  description  A
Vlan20
  description  B
VLAN 30
  description  C
```

Example: Implementing automated device configuration using Python

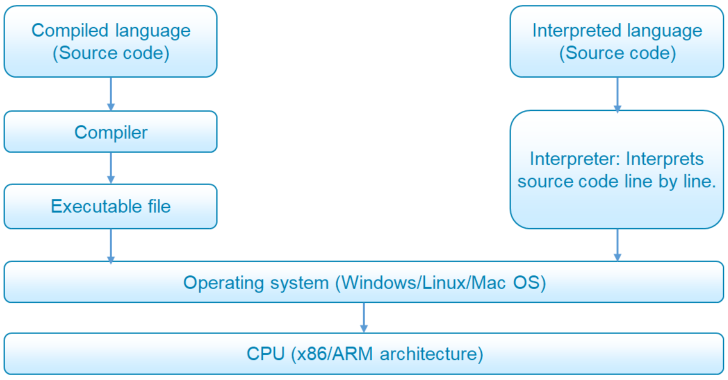
- What can network automation do? The most intuitive example of network automation is automated device configuration. This process can be divided into two steps: writing a configuration file, and writing Python code to push the configuration file to a device.
- Write the configuration script in command line interface (CLI) mode, and then upload the script to the device using Telnet/SSH. This method is easy to understand for network engineers who are beginning to learn network programmability and automation. This presentation describes how to implement network automation.

Contents

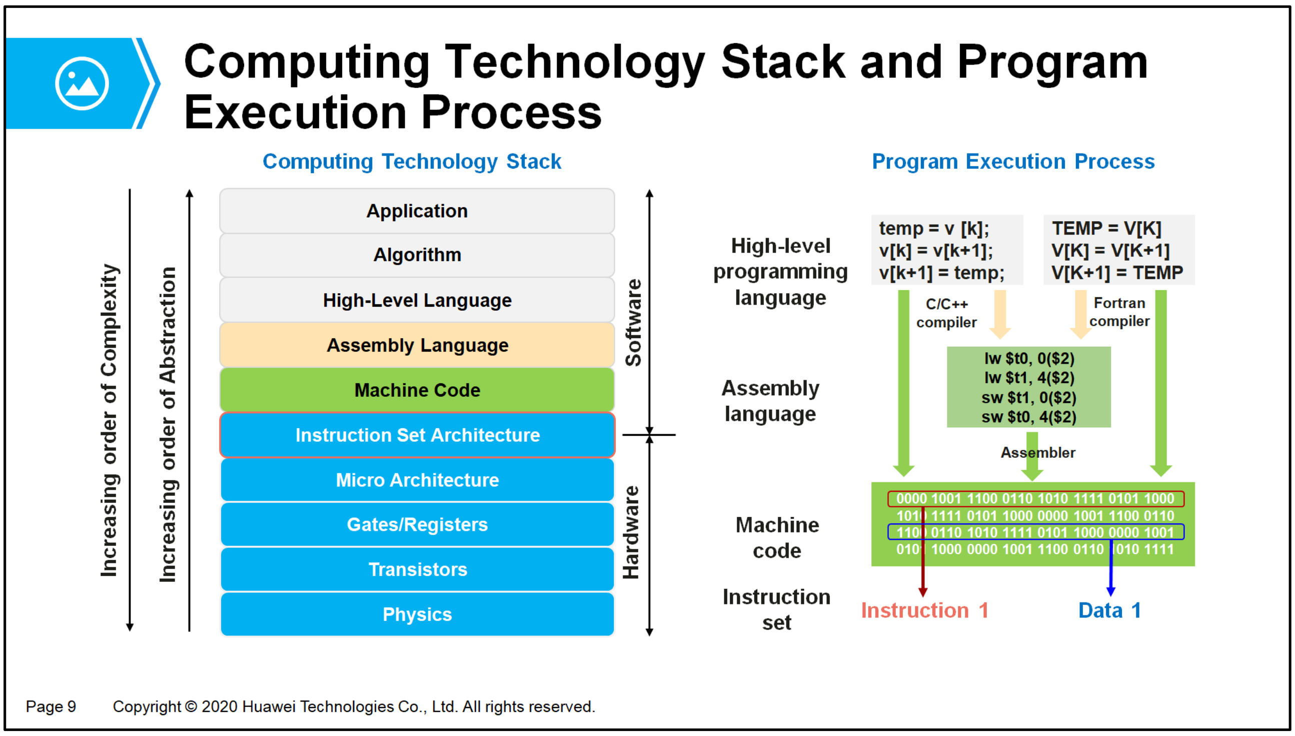


Programming Languages

- A programming language is used to write a computer program and control behavior of a computer.
- According to whether compilation is required before execution of a language, the programming language may be classified into the compiled language, and interpreted language that does not need to be compiled.


Compiled language (Source code)
Compiler
Executable file
Interpreted language (Source code)
Interpreter: Interprets source code line by line.
Operating system (Windows/Linux/Mac OS)
CPU (x86/ARM architecture)

- Based on language levels, computer languages can also be classified into machine language, assembly language, and high-level language. The machine language consists of 0 and 1 instructions that can be directly identified by a machine. Because machine languages are obscure, hardware instructions 0 and 1 are encapsulated to facilitate identification and memory (such as MOV and ADD), which is assembly language. The two languages are low-level languages, and other languages are high-level languages, such as C, C++, Java, Python, Pascal, Lisp, Prolog, FoxPro, and Fortran. Programs written in high-level languages cannot be directly identified by computers. The programs must be converted into machine languages before being executed.



Computing Technology Stack and Program Execution Process

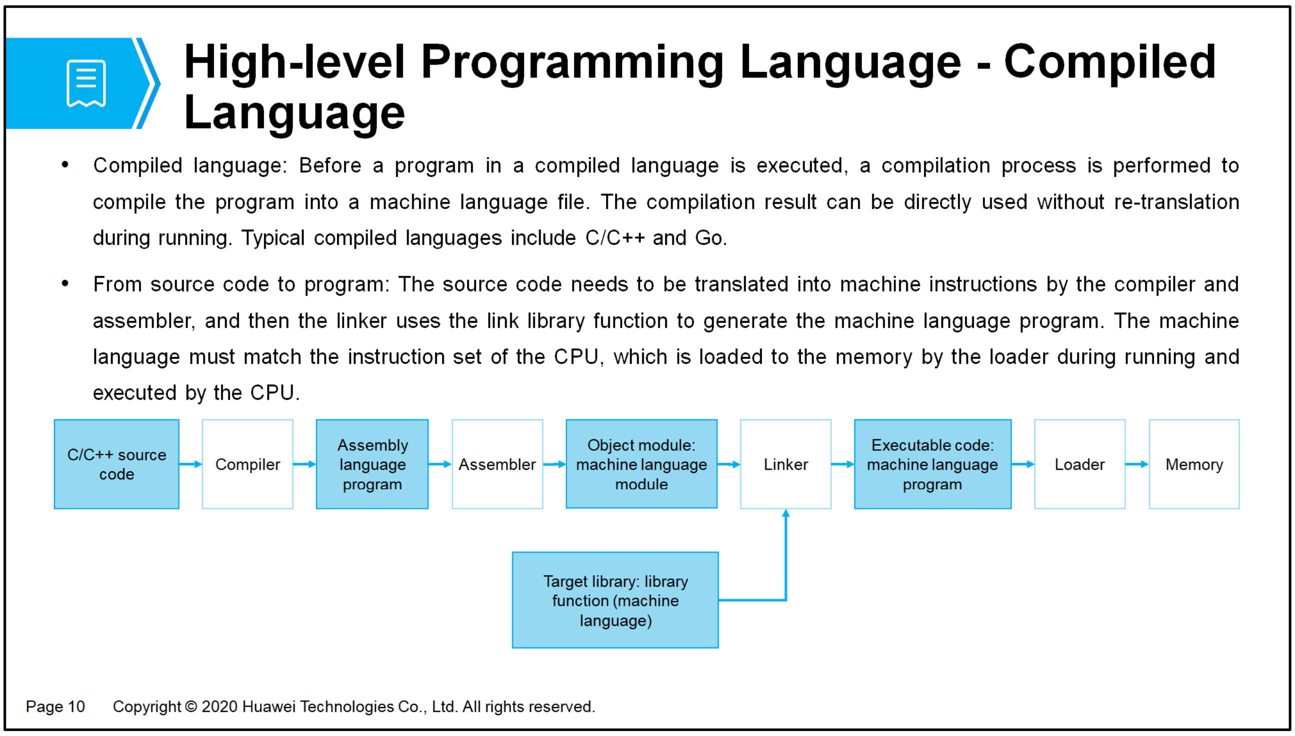
```
temp = v [k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
TEMP = V[K]
V[K] = V[K+1]
V[K+1] = TEMP
```

```
lw $t0, 0($2)
lw $t1, 4($2)
sw $t1, 0($2)
sw $t0, 4($2)
```

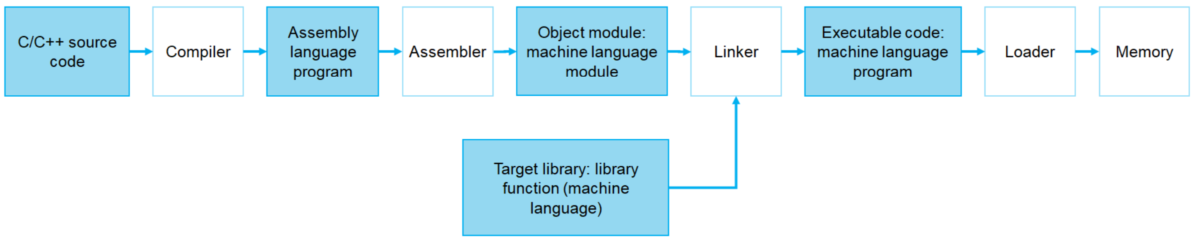
```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

- A process of executing a computer's technology stack and programs. On the left is the computing technology stack. From the bottom layer of the hardware, physical materials and transistors are used to implement gate circuits and registers, and then the micro architecture of the CPU is formed. The instruction set of the CPU is an interface between hardware and software. An application drives hardware to complete calculation using an instruction defined in the instruction set.
- Applications use certain software algorithms to implement service functions. Programs are usually developed using high-level languages, such as C, C++, Java, Go, and Python. The high-level language needs to be compiled into an assembly language, and then the assembler converts the assembly language into binary machine code based on a CPU instruction set.
- A program on disk is a binary machine code consisting of a pile of instructions and data, that is, a binary file.



High-level Programming Language - Compiled Language

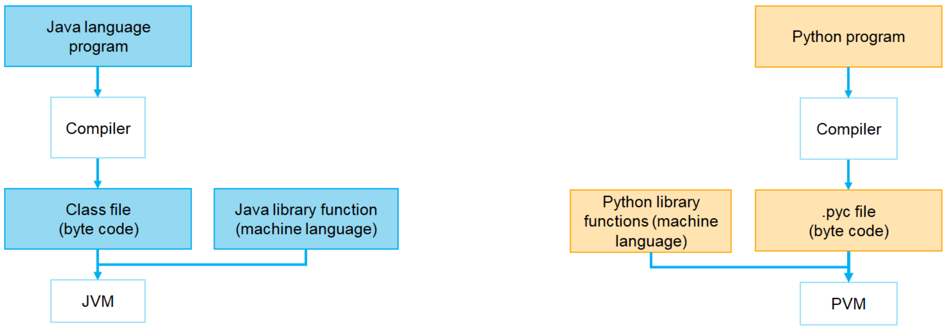
- Compiled language: Before a program in a compiled language is executed, a compilation process is performed to compile the program into a machine language file. The compilation result can be directly used without re-translation during running. Typical compiled languages include C/C++ and Go.
- From source code to program: The source code needs to be translated into machine instructions by the compiler and assembler, and then the linker uses the link library function to generate the machine language program. The machine language must match the instruction set of the CPU, which is loaded to the memory by the loader during running and executed by the CPU.


C/C++ source code
Compiler
Assembly language program
Assembler
Object module: machine language module
Linker
Executable code: machine language program
Loader
Memory
Target library: library function (machine language)

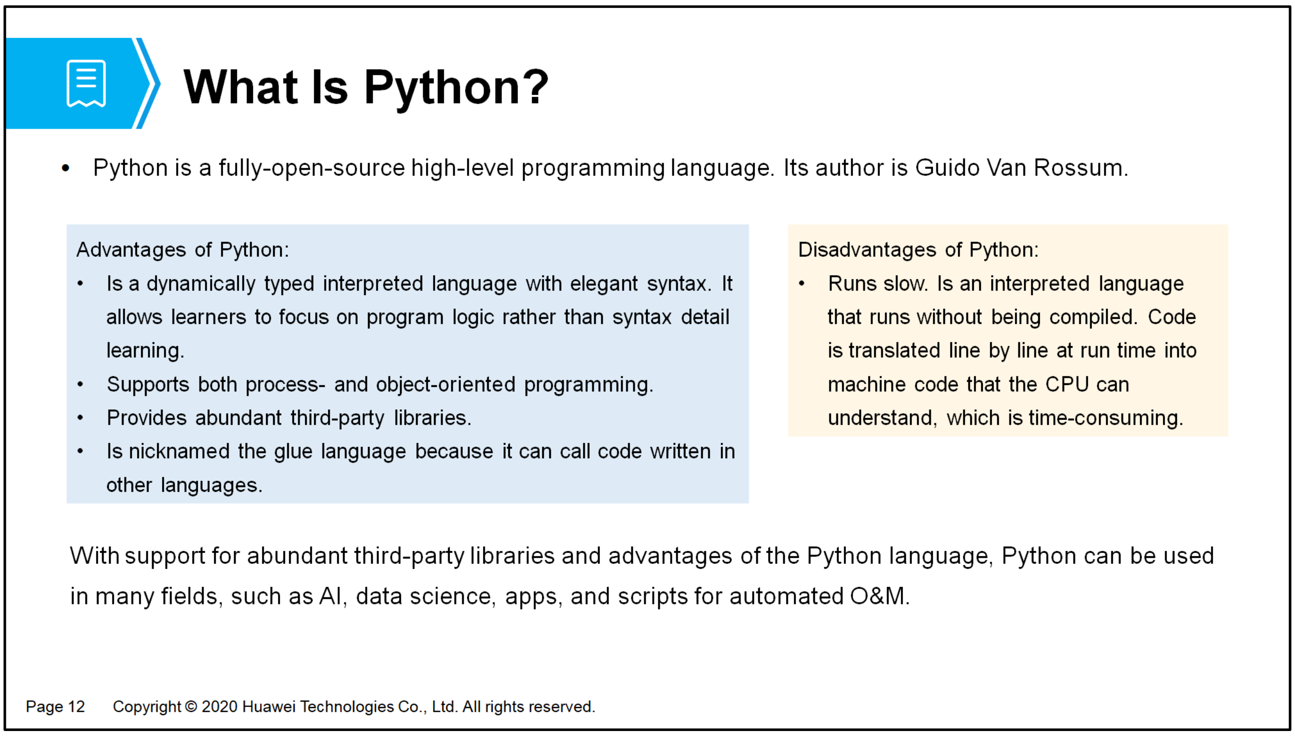
- Compiled languages are compiled into formats, such as .exe, .dll, and .ocx, that can be executed by machines. Compilation and execution are separated and cannot be performed across platforms. For example, x86 programs cannot run on ARM servers.

High-level Programming Language - Interpreted Language

- Interpreted language: Interpreted language programs do not need to be compiled before running. They are translated line by line when running. Typically, Java and Python are interpreted languages.
- Process from source code to programs: Source code of an interpreted language is generated by the compiler and then interpreted and executed by a virtual machine (VM) (for example, JVM/PVM). The VM shields the differences between CPU instruction sets. Therefore, portability of the interpreted language is relatively good.



- JVM: Java virtual machine
- PVM: Python VM



What Is Python?

- Python is a fully-open-source high-level programming language. Its author is Guido Van Rossum.

Advantages of Python:

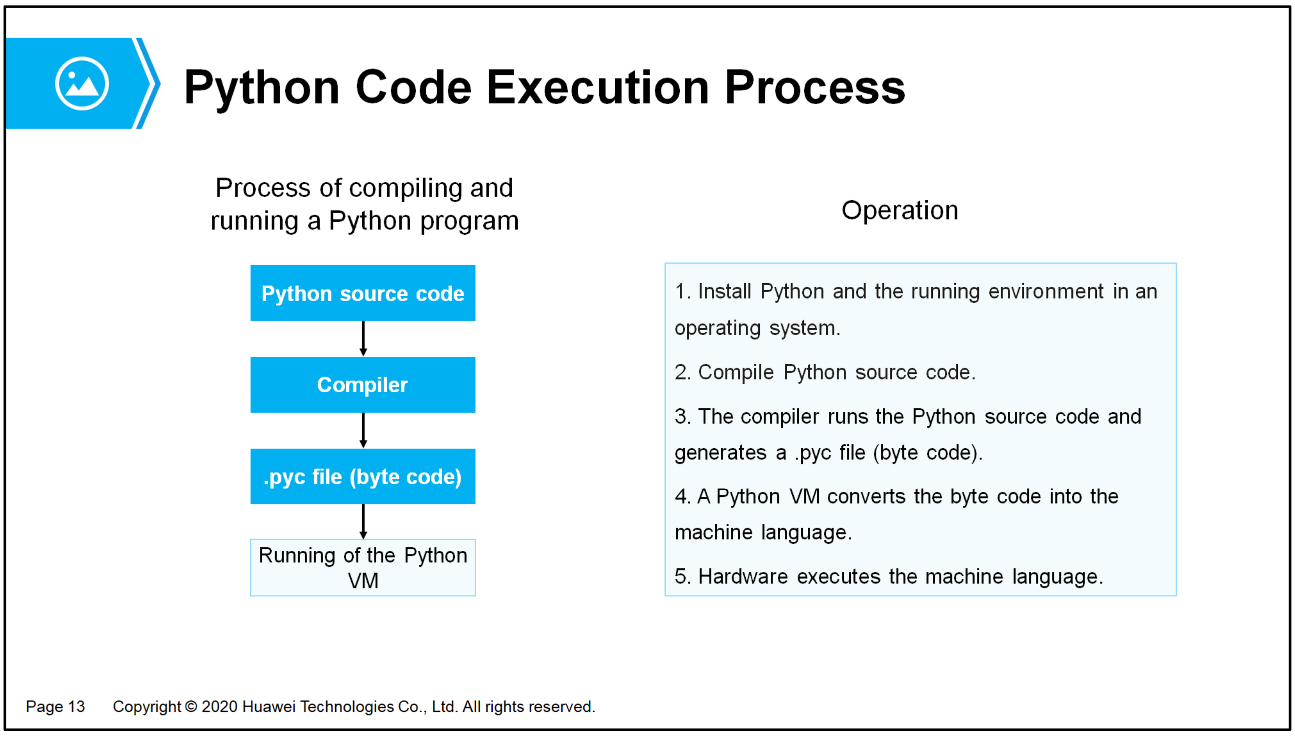
- Is a dynamically typed interpreted language with elegant syntax. It allows learners to focus on program logic rather than syntax detail learning.
- Supports both process- and object-oriented programming.
- Provides abundant third-party libraries.
- Is nicknamed the glue language because it can call code written in other languages.

Disadvantages of Python:

- Runs slow. Is an interpreted language that runs without being compiled. Code is translated line by line at run time into machine code that the CPU can understand, which is time-consuming.

With support for abundant third-party libraries and advantages of the Python language, Python can be used in many fields, such as AI, data science, apps, and scripts for automated O&M.

- Python is also a dynamically typed language. The dynamically typed language automatically determines the type of variable during program running. The type of a variable does not need to be declared.



Python Code Execution Process

Process of compiling and running a Python program

$$\begin{tikzcd}[row sep=small]
\text{Python source code} \arrow[d] \\
\text{Compiler} \arrow[d] \\
\text{.pyc file (byte code)} \arrow[d] \\
\text{Running of the Python VM}
\end{tikzcd}$$

Operation

1. Install Python and the running environment in an operating system.
2. Compile Python source code.
3. The compiler runs the Python source code and generates a .pyc file (byte code).
4. A Python VM converts the byte code into the machine language.
5. Hardware executes the machine language.

- Python source code does not need to be compiled into binary code. Python can run programs directly from the source code. When Python code is run, the Python interpreter first converts the source code into byte code, and then the Python VM executes the byte code.
- The Python VM is not an independent program and does not need to be installed independently.

Getting Started with Python Code - Interactive Running

- Python runs in either interactive or script mode.
- Interactive programming does not require script files to be created, and code is written in the interactive mode of the Python interpreter.

```
               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.
1. Input   --  >>> print ("hello world")
2. Output --   hello world
3. Input   --  >>> a = 1
4. Input   --  >>> b = 2
5. Input   --  >>> print ( a + b )
6. Output --   3
               >>>
```

Getting Started with Python Code - Script-based Running

- Code in script mode can run on various Python compilers or in integrated development environments. For example, IDLE, Atom, Visual Studio, Pycharm, and Anaconda provided by Python can be used.

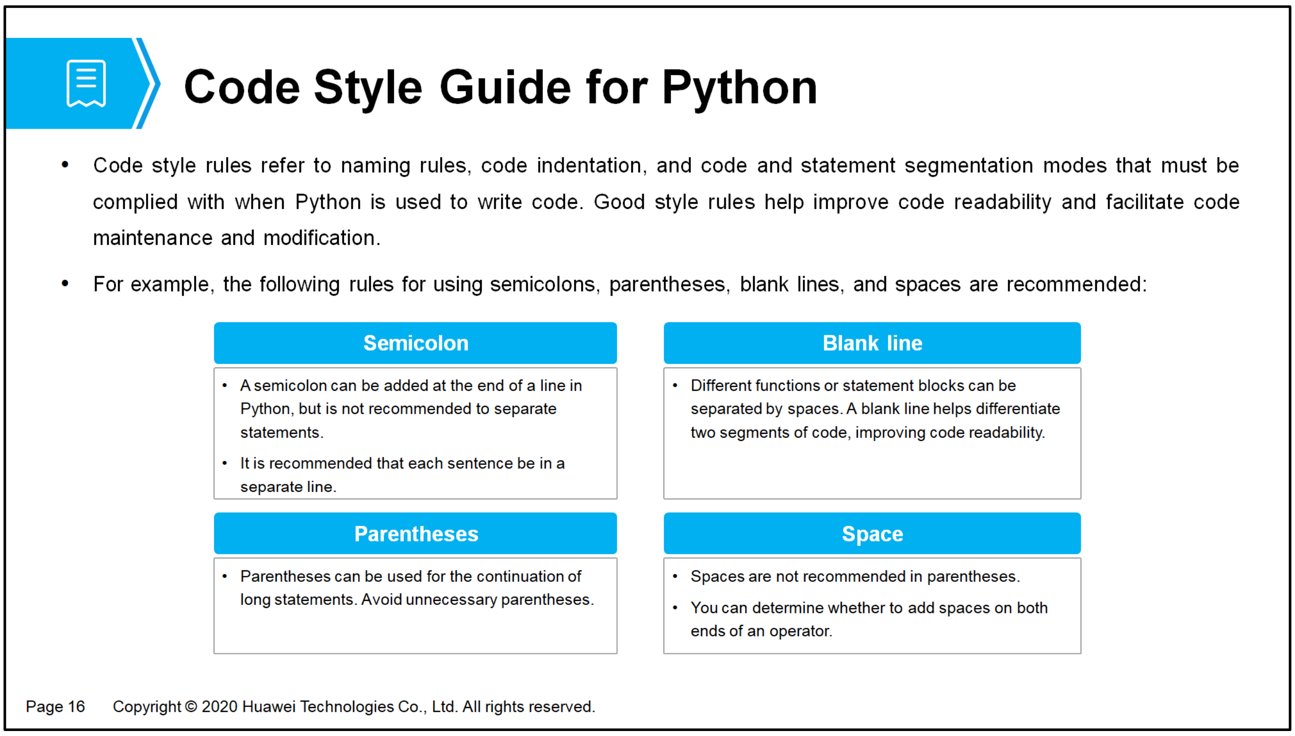
demo.py

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

1 Write a Python script file (.py).

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

2 Execute the script file.



Code Style Guide for Python

- Code style rules refer to naming rules, code indentation, and code and statement segmentation modes that must be complied with when Python is used to write code. Good style rules help improve code readability and facilitate code maintenance and modification.
- For example, the following rules for using semicolons, parentheses, blank lines, and spaces are recommended:

Semicolon

- A semicolon can be added at the end of a line in Python, but is not recommended to separate statements.
- It is recommended that each sentence be in a separate line.

Blank line

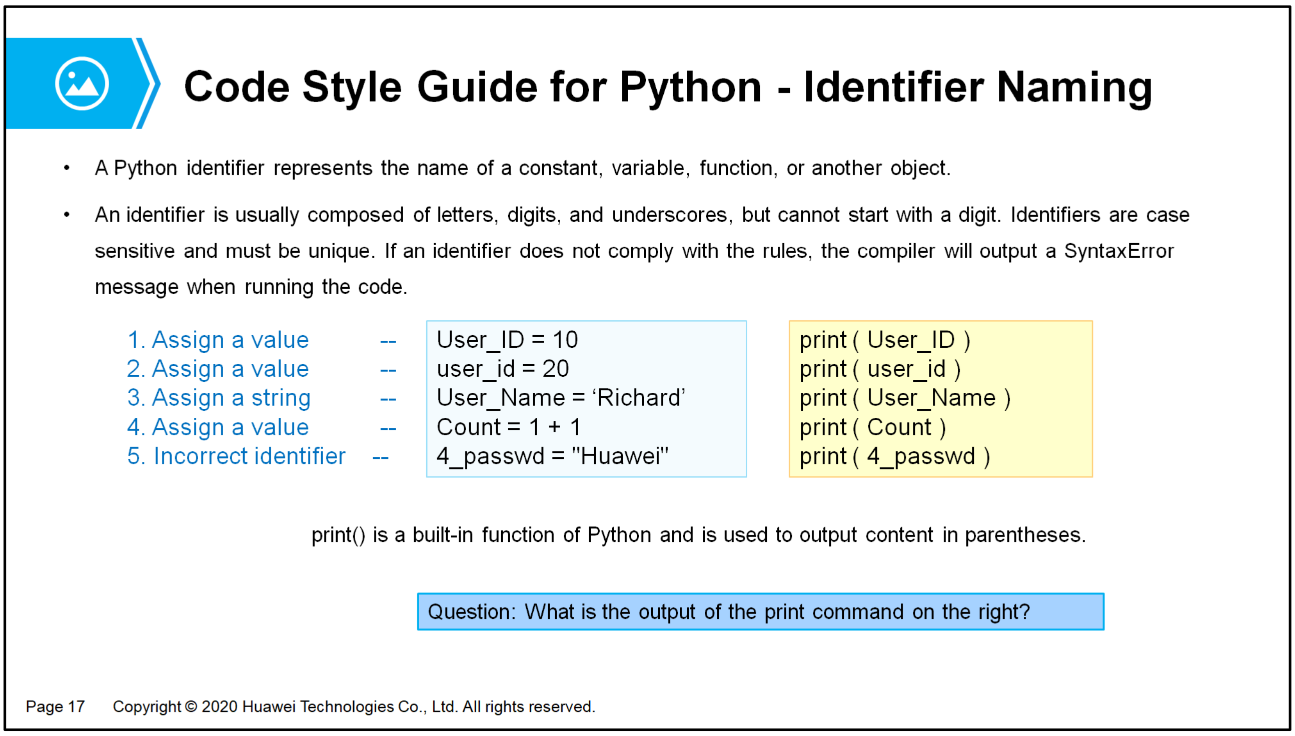
- Different functions or statement blocks can be separated by spaces. A blank line helps differentiate two segments of code, improving code readability.

Parentheses

- Parentheses can be used for the continuation of long statements. Avoid unnecessary parentheses.

Space

- Spaces are not recommended in parentheses.
- You can determine whether to add spaces on both ends of an operator.



Code Style Guide for Python - Identifier Naming

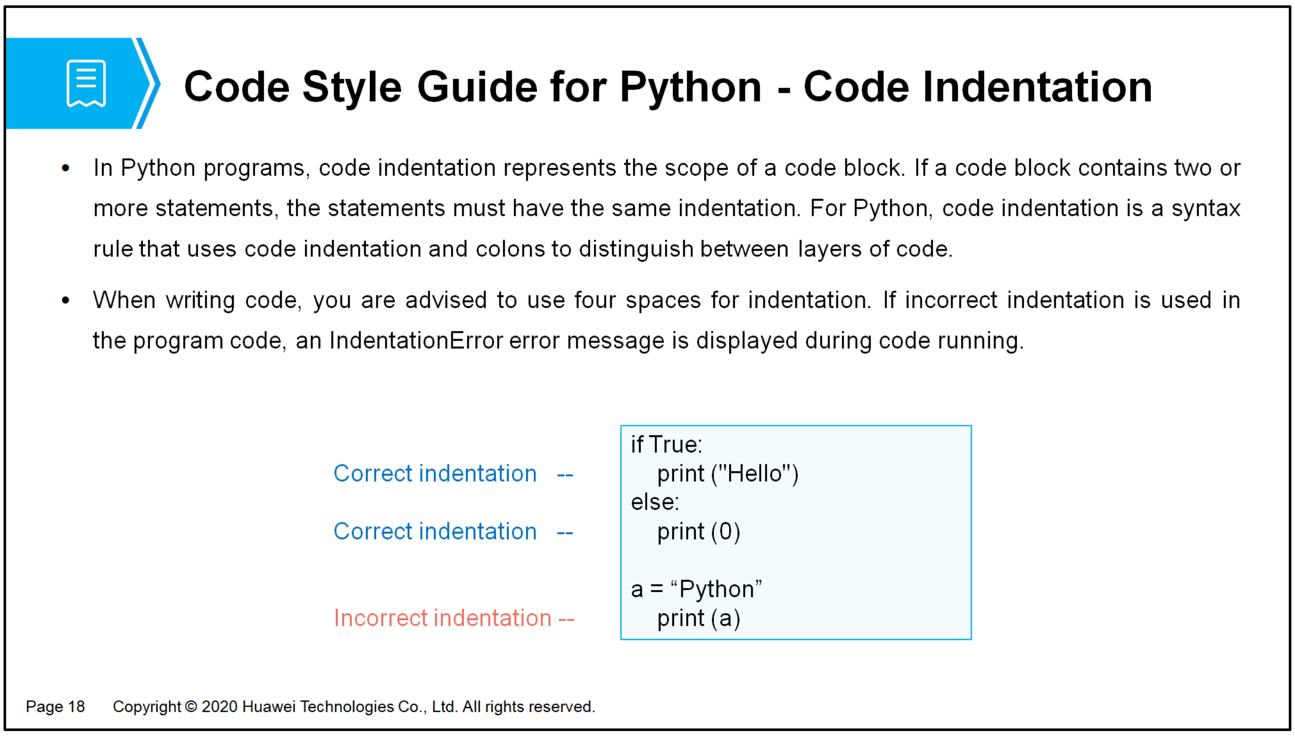
- A Python identifier represents the name of a constant, variable, function, or another object.
- An identifier is usually composed of letters, digits, and underscores, but cannot start with a digit. Identifiers are case sensitive and must be unique. If an identifier does not comply with the rules, the compiler will output a SyntaxError message when running the code.

1. Assign a value	--	User_ID = 10	print (User_ID)
2. Assign a value	--	user_id = 20	print (user_id)
3. Assign a string	--	User_Name = 'Richard'	print (User_Name)
4. Assign a value	--	Count = 1 + 1	print (Count)
5. Incorrect identifier	--	4_passwd = "Huawei"	print (4_passwd)

print() is a built-in function of Python and is used to output content in parentheses.

Question: What is the output of the print command on the right?

- Basic data types of Python are Boolean (True/False), integer, floating point, and string. All data (Boolean values, integers, floating points, strings, and even large data structures, functions, and programs) in Python exists in the form of objects. This makes the Python language highly unified.
- The execution results are 10, 20, Richard, 2, and SyntaxError, respectively.
- This presentation does not describe Python syntax. For Python syntax details, see the HCIP course.



Code Style Guide for Python - Code Indentation

- In Python programs, code indentation represents the scope of a code block. If a code block contains two or more statements, the statements must have the same indentation. For Python, code indentation is a syntax rule that uses code indentation and colons to distinguish between layers of code.
- When writing code, you are advised to use four spaces for indentation. If incorrect indentation is used in the program code, an IndentationError error message is displayed during code running.

```
                          if True:
Correct indentation   --      print ("Hello")
                          else:
Correct indentation   --      print (0)

                          a = “Python”
Incorrect indentation --      print (a)
```

- if...else... is a complete block of code with the same indentation.
- print(a) calls parameter a, and it is in the same code block with if...else...clause.

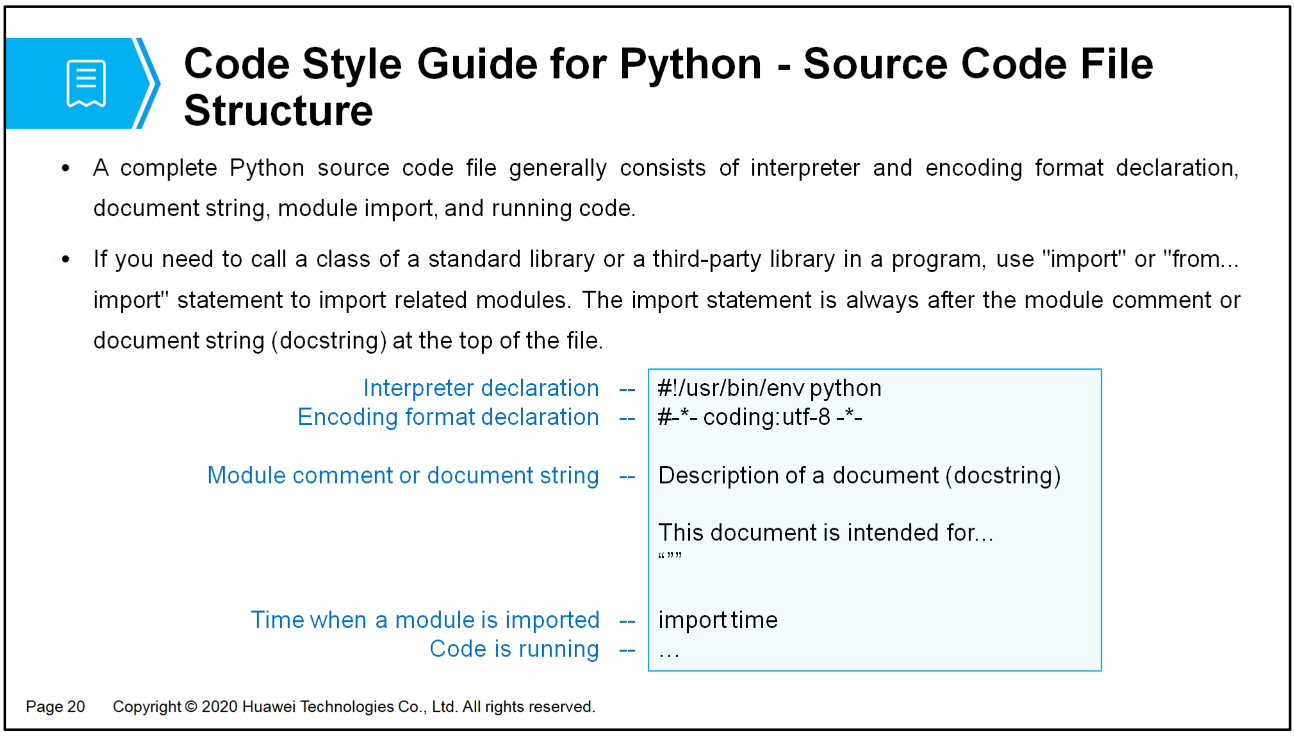
Code Style Guide for Python - Using Comments

- Comments are explanations added to programs to improve program readability. In the Python program, comments are classified into single-line comments and multi-line comments.
- A single-line comment starts with a pound sign (#).
- A multi-line comment can contain multiple lines, which are enclosed in a pair of three quotation marks ('''...''' or """ ...""").

Single-line comment --

Multi-line comment --

```
# Assign a string to a.
a = “Python”
print (a)

“””
The output is Python.
“””
```



Code Style Guide for Python - Source Code File Structure

- A complete Python source code file generally consists of interpreter and encoding format declaration, document string, module import, and running code.
- If you need to call a class of a standard library or a third-party library in a program, use "import" or "from... import" statement to import related modules. The import statement is always after the module comment or document string (docstring) at the top of the file.

Interpreter declaration --	#!/usr/bin/env python
Encoding format declaration --	#-*- coding:utf-8 -*-
Module comment or document string --	Description of a document (docstring) This document is intended for... “””
Time when a module is imported --	import time
Code is running --	...

- The interpreter declaration is used to specify the path of the compiler that runs this file (the compiler is installed in a non-default path or there are multiple Python compilers). In the Windows , you can omit the first line of the interpreter declaration in the preceding example.
- The encoding format declaration is used to specify the encoding type used by the program to read the source code. By default, Python 2 uses ASCII code (Chinese is not supported), and Python 3 supports UTF-8 code (Chinese is supported).
- docstring is used to describe the functions of the program.
- time is a built-in module of Python and provides functions related to processing time.

Python Functions and Modules

- A function is a block of organized, reusable code that is used to perform a single, related action. It can improve the modularity of the program and code utilization. The function is defined using the def keyword.
- A module is a saved Python file. Modules can contain definitions of functions, classes, and variables that can then be utilized in other Python programs. The only difference between a module and a regular Python program is that the module is used for importing by other programs. Therefore, a module usually does not have a main function.

demo.py

```
def sit(): #Define a function.
    print ('A dog is now sitting')

sit() #Call a function.
```

Execution result:

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

1 Write a Python file (.py).

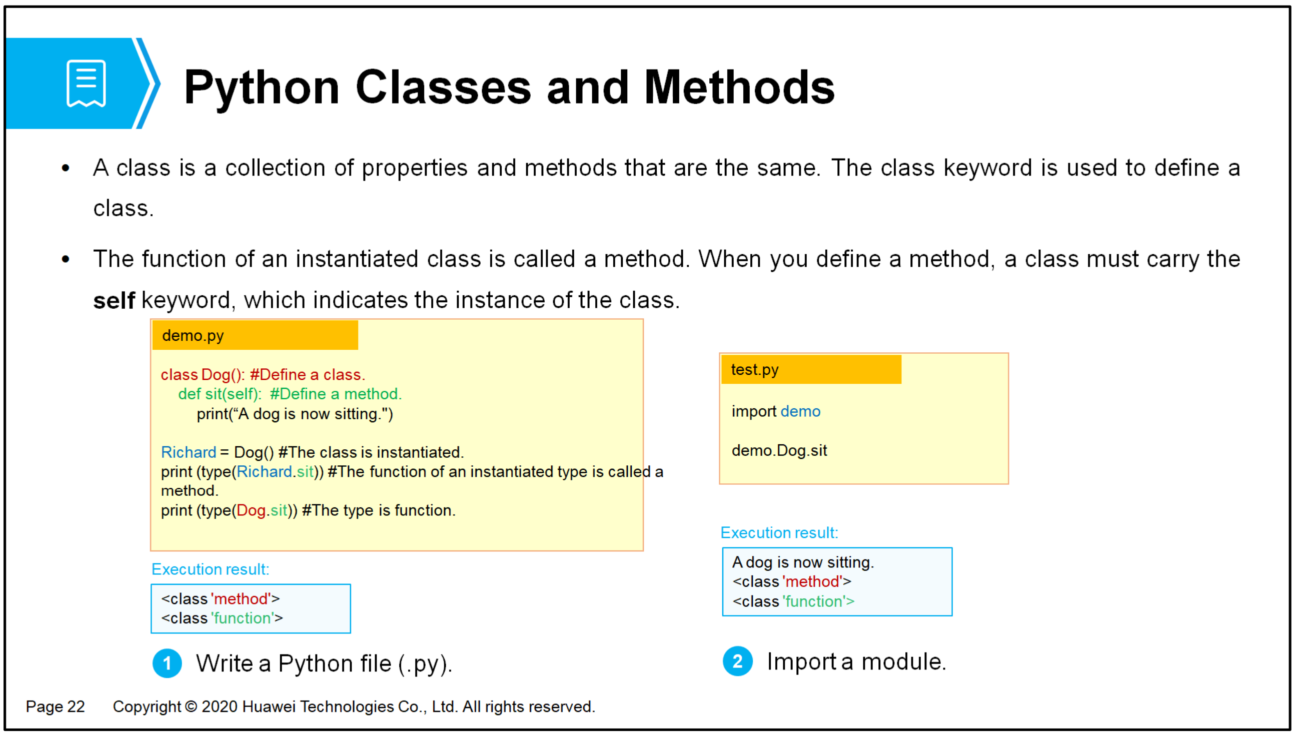
test.py

```
import demo #Import a module.

demo.sit()     #Call a function.
```

Execution result:

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

2 Import a module.



Python Classes and Methods

- A class is a collection of properties and methods that are the same. The class keyword is used to define a class.
- The function of an instantiated class is called a method. When you define a method, a class must carry the **self** keyword, which indicates the instance of the class.

demo.py

```
class Dog(): #Define a class.
    def sit(self):  #Define a method.
        print("A dog is now sitting.")

Richard = Dog() #The class is instantiated.
print (type(Richard.sit)) #The function of an instantiated type is called a method.
print (type(Dog.sit)) #The type is function.
```

Execution result:

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

1 Write a Python file (.py).

test.py

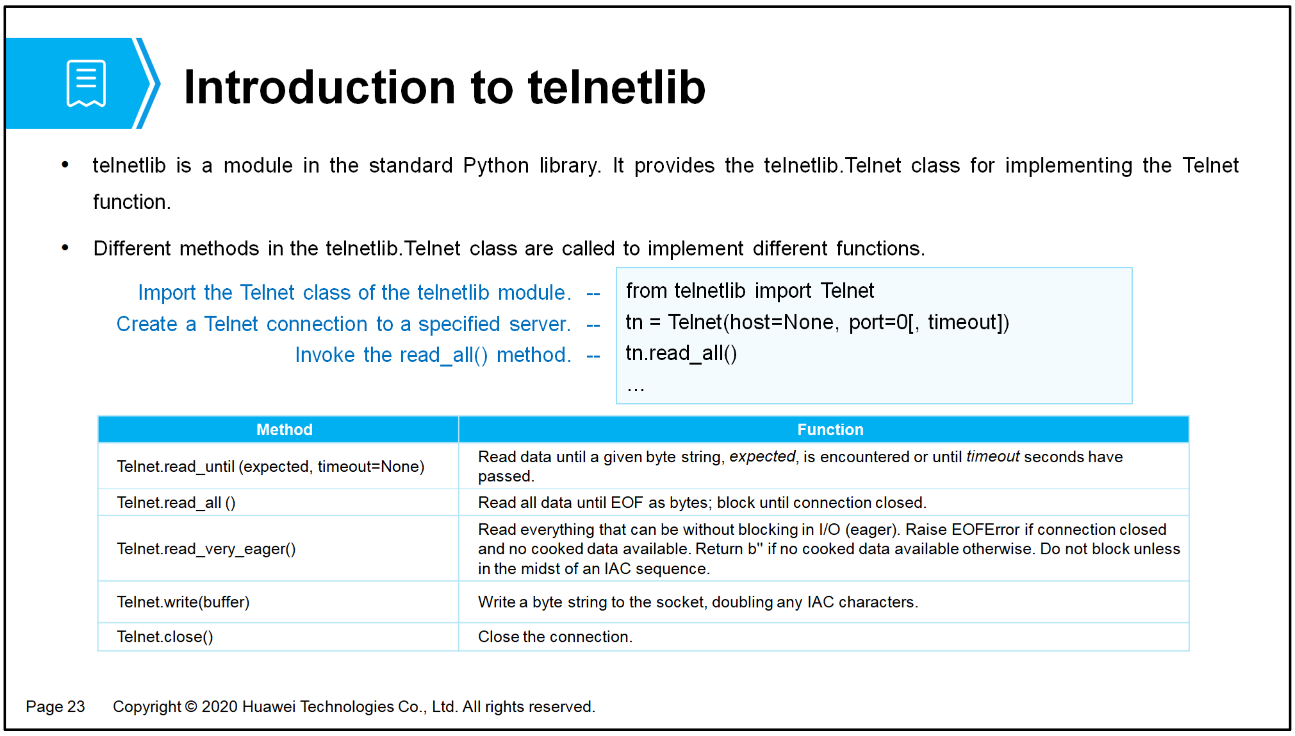
```
import demo

demo.Dog.sit
```

Execution result:

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

2 Import a module.

- Official definitions of functions and methods:
- A series of statements which returns some value to a caller. It can also be passed zero or more arguments which may be used in the execution of the body.
- A function which is defined inside a class body. If called as an attribute of an instance of that class, the method will get the instance object as its first argument (which is usually called self).
- For more information about classes, see https://docs.python.org/3/tutorial/classes.html.



Introduction to telnetlib

- telnetlib is a module in the standard Python library. It provides the telnetlib.Telnet class for implementing the Telnet function.
- Different methods in the telnetlib.Telnet class are called to implement different functions.

Import the Telnet class of the telnetlib module. --
Create a Telnet connection to a specified server. --
Invoke the read_all() method. --

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

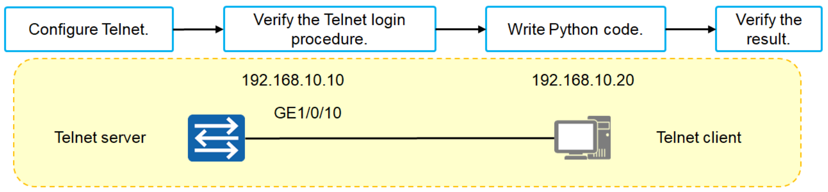
Method	Function
Telnet.read_until (expected, timeout=None)	Read data until a given byte string, *expected*, is encountered or until *timeout* seconds have passed.
Telnet.read_all ()	Read all data until EOF as bytes; block until connection closed.
Telnet.read_very_eager()	Read everything that can be without blocking in I/O (eager). Raise EOFError if connection closed and no cooked data available. Return b'' if no cooked data available otherwise. Do not block unless in the midst of an IAC sequence.
Telnet.write(buffer)	Write a byte string to the socket, doubling any IAC characters.
Telnet.close()	Close the connection.

- Telnet defines the network virtual terminal (NVT). It describes the standard representation of data and sequences of commands transmitted over the Internet to shield the differences between platforms and operating systems. For example, different platforms have different line feed commands.
- Telnet communication adopts the inband signaling mode. That is, Telnet commands are transmitted in data streams. To distinguish Telnet commands from common data, Telnet uses escape sequences. Each escape sequence consists of 2 bytes. The first byte (0xFF) is called Interpret As Command (IAC), which indicates that the second byte is a command. EOF is also a Telnet command. Its decimal code is 236.
- A socket is an abstraction layer. Applications usually send requests or respond to network requests through sockets.
- For more information, see https://docs.python.org/3/library/telnetlib.html.

Contents

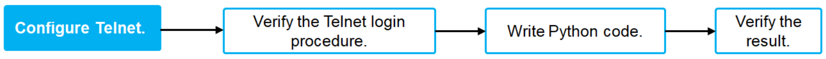
Case: Logging In to a Device Using telnetlib

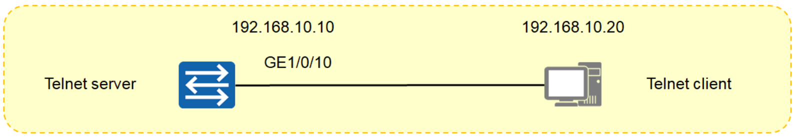
- Case description：
- A network device functions as a Telnet server, and the Python telnetlib needs to be used as a Telnet client to log in to the device.



- The implementation process is as follows：
- Configure the Telnet service.
- Manually verify and view the Telnet login procedure as a reference for code implementation.
- Compile and run Python code.
- Verify the result.

Case: Logging In to a Device Using telnetlib



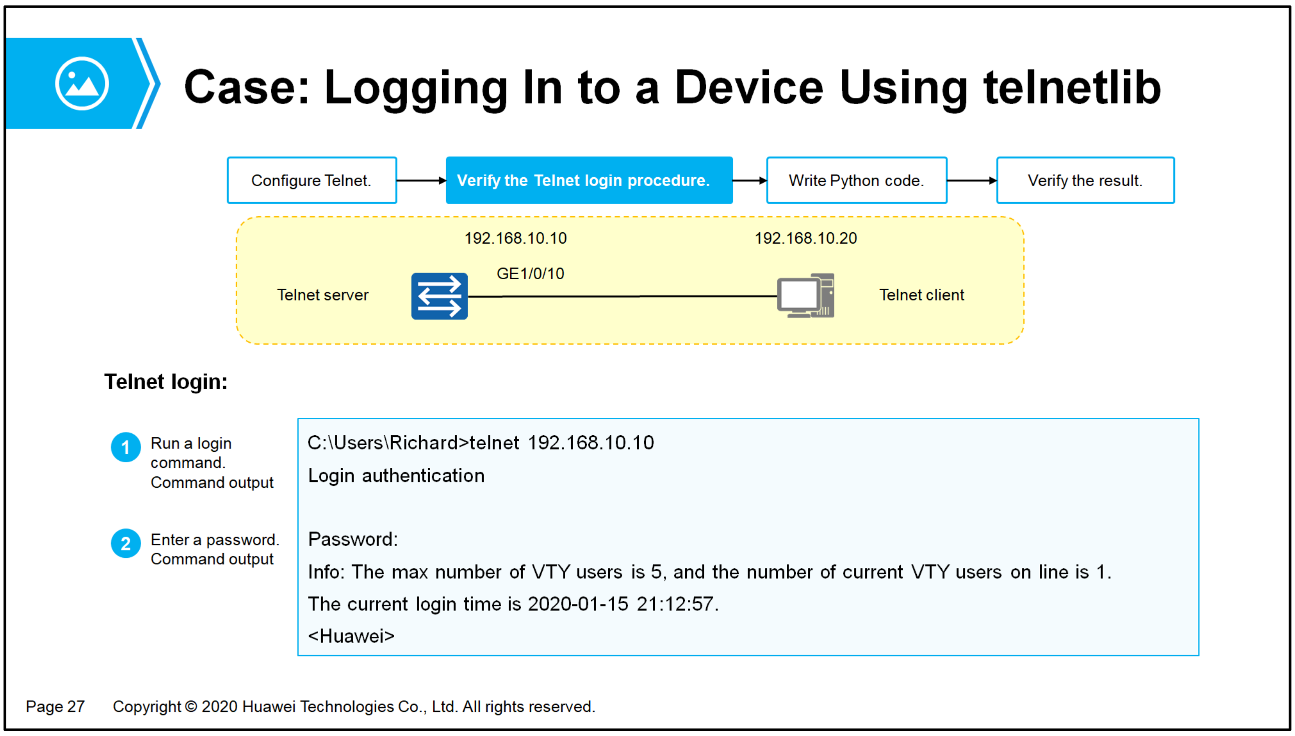


Configure the IP address of interface on the device:

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

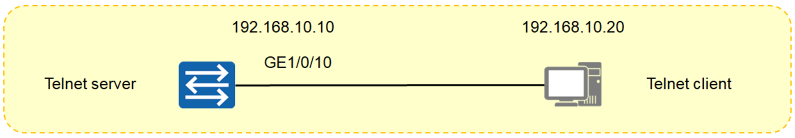
Configure the Telnet service:

```
[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
```



Case: Logging In to a Device Using telnetlib

Configure Telnet. → **Verify the Telnet login procedure.** → Write Python code. → Verify the result.

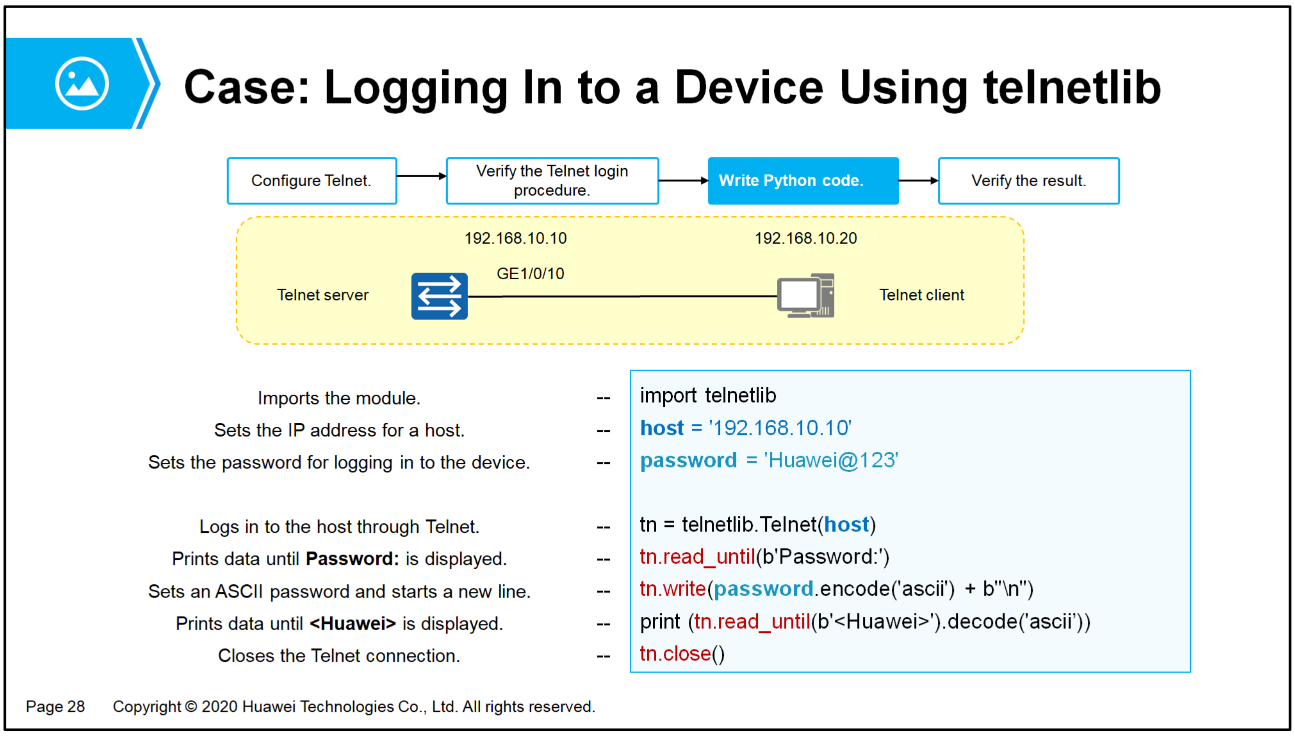


Telnet login:

1. Run a login command. Command output
2. Enter a password. Command output

```
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>
```

- In this case, the Windows operating system is used as an example. Run the telnet 192.168.10.10 command. In the preceding step, a Telnet login password is set. Therefore, the command output is
- Password:
- Enter the password Huawei@123 for authentication. The login is successful.



Case: Logging In to a Device Using telnetlib

Configure Telnet.
Verify the Telnet login procedure.
Write Python code.
Verify the result.
192.168.10.10
192.168.10.20
GE1/0/10
Telnet server
Telnet client

Imports the module.	--	`import telnetlib`
Sets the IP address for a host.	--	`host = '192.168.10.10'`
Sets the password for logging in to the device.	--	`password = 'Huawei@123'`
Logs in to the host through Telnet.	--	`tn = telnetlib.Telnet(host)`
Prints data until **Password:** is displayed.	--	`tn.read_until(b'Password:')`
Sets an ASCII password and starts a new line.	--	`tn.write(password.encode('ascii') + b"\n")`
Prints data until **<Huawei>** is displayed.	--	`print (tn.read_until(b'<Huawei>').decode('ascii'))`
Closes the Telnet connection.	--	`tn.close()`

- In Python, the encode() and decode() functions are used to encode and decode strings in a specified format, respectively. In this example, password.encode('ascii') is to convert the string Huawei@123 into the ASCII format. The encoding format complies with the official requirements of the telnetlib module.
- Add a string b, b'str', indicating that the string is a bytes object. In this example, b'Password:' indicates that the string Password:' is converted into a string of the bytes type. The encoding format complies with the official requirements of the telnetlib module.
- For more information about Python objects, see https://docs.python.org/3/reference/datamodel.html#objects-values-and-types.

Case: Running Result Comparison

Configure Telnet. → Verify the Telnet login procedure. → Write Python code. → **Verify the result.**

Manual Telnet login result:

```
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 code execution result:

```
#Run Python code in the compiler.
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>
```