Python Programming Basics

www.huawei.com

HUAWEI

Foreword

- Python is an easy-to-learn and functional programming language.
- Python has an effective advanced data structure and enables simple and rapid object-oriented programming.
- With its beautiful syntax, dynamic types and good interpretability, Python has become an ideal language in terms of scripting and developing apps on most platforms.

Objectives

- After completing this course, you will be able to:
  - Understand the compilation environment and installation process of Python.
  - Master the data structure, data types, conditional statements, loop statements, functions, and modules of Python.
  - Apply the Python programming basics to actual scenarios.

Contents

History of Python

- Python is one of the achievements of free software.
- Python is purely free software and both its source code and interpreter comply with the GNU General Public License (GPL) protocol.

Founder	Guido van Rossum
When and Where	Created in Amsterdam during Christmas in 1989.
Meaning of Name	A big fan of Monty Python's Flying Circus
Origin	Influenced by Modula-3, Python is a descendant of ABC that would appeal to Unix/C hackers.

Origin of Python

- Guido van Rossum:
  - Master of Mathematics
  - Master of Computer Science
- Philosophy of Python:
  - Python is engineering but not art.
  - There should be one, and preferably only one, obvious way to do it.
  - Simple is better than complex, and explicit is better than implicit.

What is Python?

- Python is a programming language.
- Python is a general-purpose and advanced programming language.
- Python applies to programming in <u>many fields</u>:
  - Data science
  - Writing system tools
  - Developing applications with graphical UIs
  - Writing network-based software
  - Interacting with databases

Python VS Other Languages (1)

- Python VS C:
  - Python is dynamic while C is static.
  - Memory is managed by the developer in C but by the interpreter in Python.
  - Python has many libraries but C has no standard library for a hybrid tuple (Python list) and a hash table (Python dictionary).
  - Python cannot be used to write a kernel but C can.
  - C or C++ extends Python functions based on the Python APIs.
- Python VS SHELL:
  - Python has simple syntax and is easy to transplant.
  - Shell has a longer script.
  - Python can reuse code and embraces simple code design, advanced data structure, and modular components.

Python VS Other Languages (2)

- Python VS Java:
  - Python is dynamic while Java is static.
  - Python supports object-oriented and function-based programming while Java supports object-oriented programming only.
  - Python is simpler than Java, and typically applies to quick prototyping.
  - Python and Java enable multiple programmers to develop a large project together step by step.

Development Environment of Python

- VIM: mainly for Linux
- IDLE: integrated development environment
- Sublime Text: a lightweight editing tool
- Eclipse: chargeable
- Eric4: powerful, based on PyQT4
- Boa: similar to IDE (wxPython) of Delphi
- WingIDE: shared software
- Other editors: notepad++, editplus…

Advantages of Python (1)

- Simple: Python is an ideal programming language of simplicity.
- Easy-to-learn: few key words, simple structure, and explicit syntax
- Open-source: Python is one of Free/Libre and Open-Source Software (FLOSS).
- Interpretability: Python program does not need to be compiled into binary code, and can be directly run from source code.
- Transplantability: Python has been transplanted to multiple platforms thanks to its open-source nature.
    - The platforms include Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS,Psion, Acom RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE, and even Symbian and Android developed by Google based on Linux.
- Scalability: If you want to speed up running of some key code or keep some algorithms not open, you can write some programs using C or C++ and use them in Python.

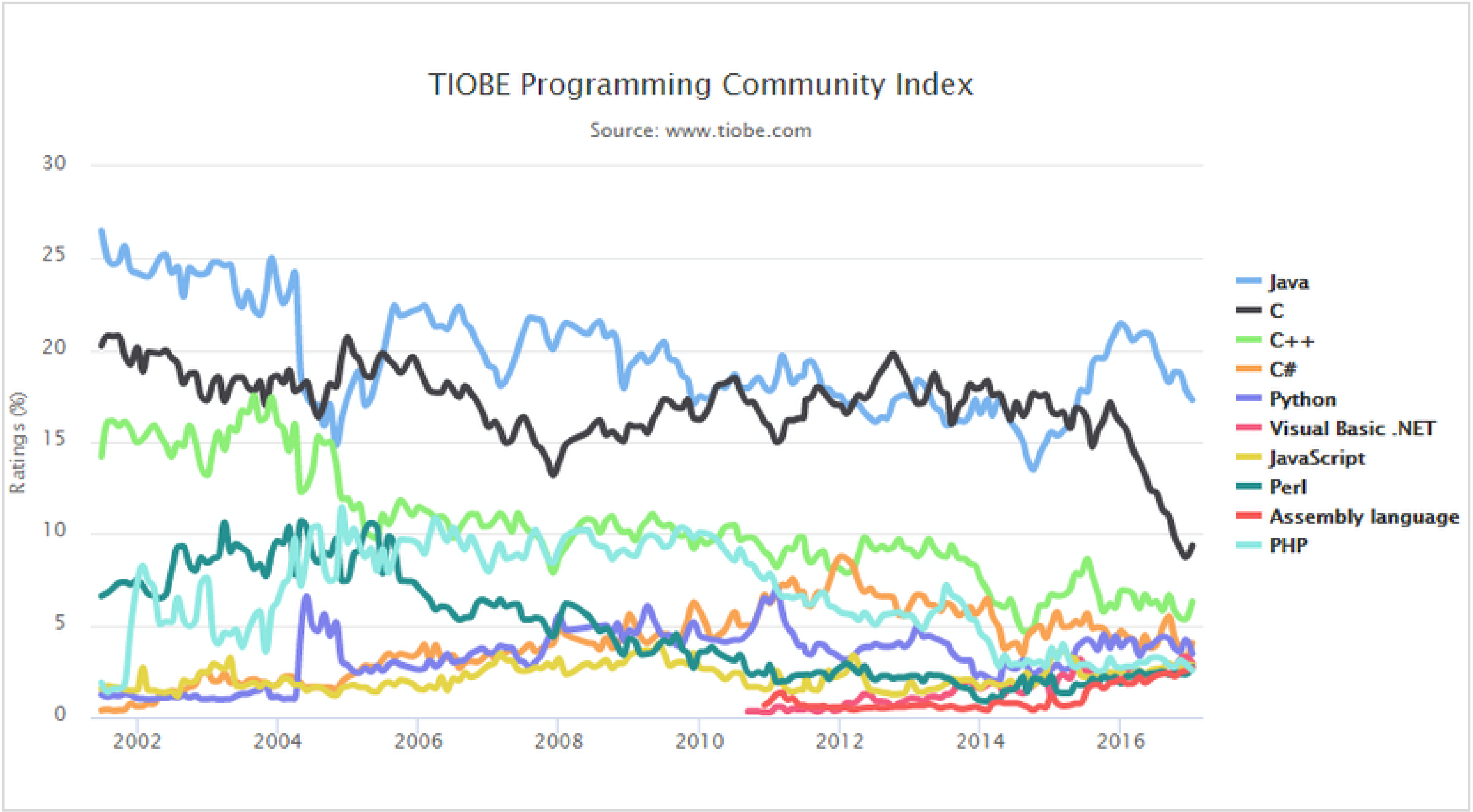
Advantages of Python (2)

- Advanced language: When you write a program in Python, you do not need to consider the underlying-layer details, like how to manage the memory that your program uses.
- Embedded: Python can be embedded in a C or C++ program to provide scripting to program users.
- Object-oriented: Python supports both process-oriented programming and object-oriented programming. In a process-oriented language, a program is built from a process or a function that is merely reusable code. In an object-oriented language, programs are built from the combination of data and functionality.
- Rich libraries: The Python standard library is huge. It can help you with all kinds of work, including regular expressions, document generation, unit tests, threads, databases, Web browsers, CGI, FTP, e-mail, XML-RPC, HTML, WAV files, password systems, GUI, TK, and other system-related operations.

Features of Python Statements

- Dynamic: Objects (like attributes and methods) can be changed during execution.
- Python uses indentation instead of a pair of curly braces {} to divide a block of statements.
- Multiple statements on one line are separated by “;”.
- The symbol used for commenting out a line is #, and doc string ("... ") is used to comment out multiple lines.
- Variables do not need type definitions.
- Functional Programming (FP) is available.

Popularity Ranking



Differences Between Python 2 and Python 3 (1)

- Python 3 cannot be backwards compatible with Python 2, which requires people to decide which version of the language is to be used.
- Many libraries are only for Python 2, but the development team behind Python 3 has reaffirmed the end of support for Python 2, prompting more libraries to be ported to Python 3.
- Judging from the number of Python packages that are supported by Python 3, Python 3 has become increasingly popular.

Differences Between Python 2 and Python 3 (2)

- print function
- Unicode
- Integer division
- Exceptions
- Xrange
- Data type
- Inequality operator

Installing Python (1)

- For Linux users:
  - Download the Python package and install it.

```
$tar -zxf python3.6.4.tar.gz
$cd Python3.6.4
$./configure
$make && make install
```

  - Create soft connections.

```
$mv /usr/bin/python  /usr/bin/python.bak
$ln -s /usr/local/bin/python3.6.4
/usr/bin/python
```

  - Verify installation.

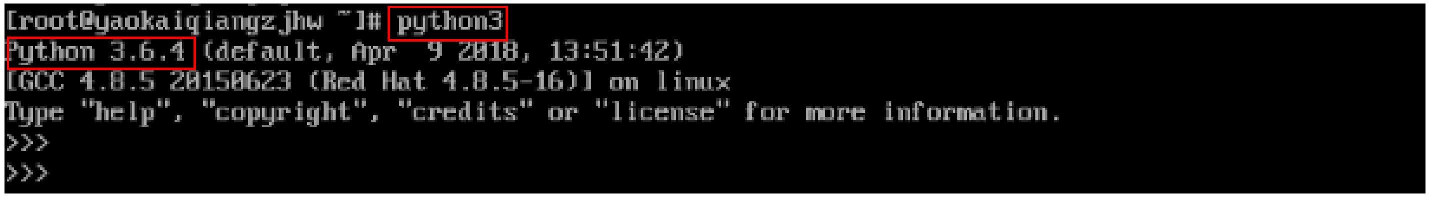
```
 $python -V
```

Installing Python (2)

- For Windows users:
  - Download the official Python setup program.
  - Select the latest Python Windows installer to download the .exe installation file.
  - Double-click the setup program, Python-3.x.exe.
  - Add environment variables: Choose **My Computer** > **Properties** > **Advanced** > **Environment Variables**, and enter your Python installation location in path.
  - Verify installation: **Start** > **Program** > **Python 3.x** > **Start Python command line**, and then enter: print("Hello World"). If the output is "Hello World", it indicates that the installation was successful.

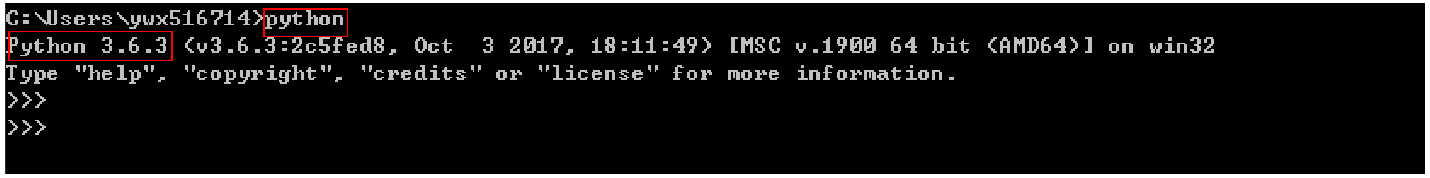
Starting Python

- Start Python on Linux:



```
[root@yaokaiqiangzjhw ~]# python3
Python 3.6.4 (default, Apr  9 2018, 13:51:42)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-16)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
```

- Start Python on Windows:



```
C:\Users\ywx516714>python
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 18:11:49) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
```

Executing a Python Program

- Using command lines:
  - On Linux:
    - Enter a Python command in the Linux command line.
  - On Windows:
    - Enter a Python command on the DOS prompt.
- Using scripts:
  - Store Python Statements in a script file and execute it in the command line.

```
Input: python hello.py
Output: hello world !
```

Vim

- Vim is a lightweight IDE.
- If you do not install too many plug-ins or plug-in performance is good, using VIM for development has a low requirement on hardware.
- Vim can achieve a consistent programming experience on local and remote servers.
- Vim has an editing speed of "what you think is what you have”.

Contents

Lists

- A list is an ordered set of elements that you can add and delete at any time.
- The element types in the list can be different. You can access each element in the list by index. The first digit of the index starts from 0, and the reverse index starts from -1.

Common Operations on Lists

- Access
- Update (append and insert)
- Delete elements (del)
- Operator on list script (+/*)
- List interception

Functions of Python Lists

- cmp(list1, list2),
- len(list)
- max(list)
- min(list)
- list(seq)

Methods of Python Listing

- list.append(obj)
- list.count(obj)
- list.extend(seq)
- list.index(obj)
- list.insert(index, obj)
- list.pop(obj=list[-1])
- list.remove(obj)
- list.sort([func])
- list.reverse()

- Understand the difference between "append" and "extend".

```
>>>x = [1, 2, 3]
>>>y = [4, 5]
>>>x.append(y)
>>>print(x)
[1,2,3,[4,5]]
```

```
>>>x = [1, 2, 3]
>>>y = [4, 5]
>>>x.extend(y)
# equal to
>>>for i in y:
>>>x.append(i)
>>>print(x)
[1,2,3,4,5]
```

- Check whether the list is empty.

```
if len(items) == 0:
    print("empty list")
#or
if items == []:
   print("empty list")
-------------------------------------------
```

- Copy a list.

```
#Method one:
new_list = old_list[:]
#Method two:
new_list = list(old_list)
```

- Get last element of a list

```
>>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> a[len(a)-1]
10
>>> a[-1]
10
```

- Remove elements from a list.

```
>>> a = [0, 2, 2, 3]
>>> a.remove(2)
>>> a
[0, 2, 3]
# ValueError will be returned if the removed element is not within the list.
>>> a.remove(7)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: list.remove(x): x not in list·
```

"del" removes a certain element in a specified position.

"pop" is similar to "del", but "pop" can return the removed element.

- Connect two lists.

```
>>>listone = [1, 2, 3]
>>>listtwo = [4, 5, 6]
>>>mergedlist = listone + listtwo
>>>print(mergelist)
[1, 2, 3, 4, 5, 6]
```

Tuples

- Tuple is expressed using ().
- Unlike a list, a tuple cannot be modified once initialized, and the elements need to be identified when a tuple is defined.
- A tuple does not have the append () or insert () method, nor can it be assigned to another element. It has the same fetching methods as a list.
- Because a tuple is unchangeable, the code is more secure. Therefore, if possible, use a tuple instead of a list.
- A tuple is simple to create. You only need to add elements to parentheses and separate them with commas.

Common Operations on Tuples

- Access
- Modify (tuple calculation)
- Delete (del tuple)
- Tuple operators (+,*)
- Tuple index and interception
- No-close separator

Embedded Functions of Tuples

- cmp(tuple1, tuple2)
- len(tuple)
- max(tuple)
- min(tuple)
- tuple(seq)

- Define a tuple for an element.

```
t=(1,)
```

Note: In t=(1), t is not a tuple type, as the parentheses () can represent a tuple, or mathematical formula. Python stipulates that in this case, () is a mathematical formula, and a tuple with only one element must have a comma to eliminate ambiguity.

```
>>> cn=('yi','er','san')
>>> en=('one','two','three')
>>> num=(1,2,3)
>>> tmp=[cn,en,num,[1.1,2.2],'language']
>>>print(tmp)
[('yi', 'er', 'san'), ('one', 'two', 'three'), (1, 2, 3), [1.1, 2.2], 'language']
>>>print(tmp[0])
('yi', 'er', 'san')
>>>print(tmp[0][0])
yi
>>>print(tmp[0][0][0])
y
```

Contents

Definition of a String

- In Python, a string is a sequence of 0 or more characters, and a string is one of several sequences built in Python.
- In Python, strings are unchangeable, and are similar to string constants in the C and C++ languages.
- Python strings may be expressed using single quotes, double quotes and triple quotes, as well as escape characters, raw strings, and so on.
  - name='JohnSmith‘
  - name="Alice“
  - name="""Bob"""

String Formatting (1)

- Python supports the output of formatted strings. Although a complex expression may be used, the most basic use is to insert a value into a string with the string format character %s.
- String formatting in Python is accomplished by the string formatting operator (%), and its string conversion type table and its formatting operator auxiliary instructions are shown in the following tables.

```
Input: print(“My name is %s and age is %d !” %(‘AI', 63))

Output: My name is AI and age is 63 !
```

String Formatting (2)

Format	Description
%c	Character and its ASCII code
%s	String
%d	Signed integer (decimal)
%u	Unsigned integer (decimal)
%o	Unsigned integer (octal)
%x	Unsigned integer (hexadecimal)
%X	Unsigned integer (hexadecimal upper-case letters)
%e	Floating number (scientific counting method)
%E	Floating number (scientific counting method, E replacing e)
%f	Floating number (decimal point sign)
%g	Floating number (%e or %f, depending on a value)
%G	Floating number (similar to %g)
%p	Pointer (print memory address of a value using hexadecimal)

String Formatting (2)

- Auxiliary commands for formatting operators:

Symbol	Function
*	Defines width or precision of the decimal point.
-	Aligns to the left.
+	Displays + before a positive value.
<sp>	Displays space before a positive value.
#	Displays 0 before an octal number or 0x or 0X before a hexadecimal value (depending on whether x or X is used).
0	Adds 0 instead of default space before numbers.
%	%% outputs a single %.
(var)	Maps variables (dictionary arguments).
m.n	m means the minimum width and n means the number of digits after the decimal point.

String Operators

- Python does not have dedicated Char type and one character is a string with a length of 1. Python strings are not changeable and do not end with '\0'. A string is the sequence of a character and is stored in memory as follows:

	P	y	t	h	o	n
Superscript	0	1	2	3	4	5
Subscript	-6	-5	-4	-3	-2	-1

- In Python, subscripts start from 0. -1 indicates the subscript of the last value in the sequence, 1 indicates the subscript of the second character, -2 indicates the subscript of the second to last character, and so on.

String Methods

- You need to know the most useful built-in function of Python, dir, before learning more about functions. You can use the dir function to check the attributes and methods of Python strings.

```
>>> dir('')
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__ge__',
'__getattribute__', '__getitem__', '__getnewargs__', '__getslice__', '__gt__', '__hash__', '__init__',
'__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__str__', 'capitalize', 'center',
'count', 'decode', 'encode', 'endswith', 'expandtabs','find', 'index', 'isalnum', 'isalpha', 'isdigit',
'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfind', 'rindex',
'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate',
'upper', 'zfill']
>>>
```

String Modules

- In addition to the above mentioned attributes and methods, Python strings also have a string module. You can also use the dir function to check attributes and methods of the string module.

```
>>> import string
>>> dir(string)
[‘Template’, ‘_TemplateMetaclass’, ‘__builtins__’, ‘__doc__’, ‘__file__’, ‘__name__’, ‘_float’,
‘_idmap’, ‘_idmapL’, ‘_int’, ‘_long’, ‘_multimap’, ‘_re’, ‘ascii_letters’, ‘ascii_lowercase’,
‘ascii_uppercase’, ‘atof’, ‘atof_error’, ‘atoi’, ‘atoi_error’, ‘atol’, ‘atol_error’, ‘capitalize’, ‘capwords’,
‘center’, ‘count’, ‘digits’, ‘expandtabs’, ‘find’, ‘hexdigits’, ‘index’, ‘index_error’, ‘join’, ‘joinfields’,
‘letters’, ‘ljust’, ‘lower’, ‘lowercase’, ‘lstrip’, ‘maketrans’, ‘octdigits’, ‘printable’, ‘punctuation’,
replace‘, ’rfind‘, ’rindex‘, ’rjust‘, ’rsplit‘, ’rstrip‘, ’split‘, ’splitfields‘, ’strip‘, ’swapcase‘, ’translate‘, ’upp
er‘, ’uppercase‘, ’whitespace‘, ’zfill‘]
>>>
```

- Single quotes and double quotes

```
>>> s = 'python string'
>>>print(s)
python string
>>> ss="python string"
>>>print(ss)
python string
>>> sss='python "Hello World"string'
>>>print(sss)
python "Hello World"string
```

- Long strings

```
>>>print('''this is a long string
                for phrases or
                for paragraphs''')
this is a long string
for phrases or
For paragraphs
```

- Original strings (raw strings)

```
>>> rawStr = r'D:\SVN_CODE\V900R17C00_TRP\omu\src'
>>>print(rawStr)
D:\SVN_CODE\V900R17C00_TRP\omu\src
```

Width, precision, and alignment of strings

```
>>>print("%c" % 97)
a
print("%.3f" % 2.5)
2.500
>>>print("%6.3f" % 2.5)
 2.500
>>>print("%.*f" % (4, 1.5))
1.5000
>>>print("%+10x" % 10)
        +a
```

- Connect and repeat strings

```
>>> s = 'I' + 'want' + 'Python' + '.'
>>>print(s)
IwantPython.
>>> ss='Python'*3
>>>print(ss)
PythonPythonPython
```

- Delete strings

```
>>> ss='Python'*3
>>>print(ss)
PythonPythonPython
>>> del ss
>>>print(ss)
Traceback (most recent call last):
  File "<pyshell#35>", line 1, in <module>
print(ssNameError: name 'ss' is not defined)
```

Contents

Dictionaries

- A dictionary is another variable container model and can store any type of object.
- Each key value of the dictionary is separated with a colon “:" key value pair are separated by a comma "," and the entire dictionary is included in the curly braces "{}".
- The key is generally unique, and the type of the key is unchangeable. If the key repeats, the last key-value pair replaces the previous one. Key values do not need to be unique, and can take any data type.
- A dictionary has the following format:
  - d = {key1 : value1, key2 : value2 }

Python Dictionary Operations

- Access
- Modify
- Delete

Built-in Functions of Dictionaries

Function	Meaning
cmp(dict1,dict2)	Compares elements between dictionaries.
len(dict)	Counts elements in a dictionary, or total number of keys.
str(dict)	Outputs printable string expressions of a dictionary.
type(variable)	Returns the types of input variables, and returns dictionary types if the variables are dictionaries.

Built-in Methods of Dictionaries

Method	Description
has_key(x)	Returns the value if the dictionary has a key x.
keys()	Returns a key list of the dictionary.
values()	Returns a value list of the dictionary.
items()	Returns a tuple list. Each tuple includes a key and value pair of the dictionary.
clear()	Clears all items in the dictionary.
copy()	Returns a copy of the higher-layer structure of the dictionary. It does not copy the embedded structure but only reference to the structure.
update(x)	Updates the dictionary with key values in dictionary x.
get(x[,y])	Returns key x. It returns none if key x is not found and returns y if y is provided and x is not found.
pop()	Deletes the value of a given key in the dictionary. It returns the deleted value and key value has to be provided; otherwise, it returns the default value.
popitem()	Randomly returns and deletes a key and value pair in the dictionary.

Create a dictionary

```
>>> a = {'one': 1, 'two': 2, 'three': 3}
>>>print(a)
{'three': 3, 'two': 2, 'one': 1}
>>> b = dict(one=1, two=2, three=3)
>>>print(b)
{'three': 3, 'two': 2, 'one': 1}
>>> c = dict([('one', 1), ('two', 2), ('three', 3)])
>>>print(c)
{'three': 3, 'two': 2, 'one': 1}
>>> d = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
>>>print(d)
{'three': 3, 'two': 2, 'one': 1}
>>> e = dict({'one': 1, 'two': 2, 'three': 3})
>>>print(e)
{'one': 1, 'three': 3, 'two': 2}
>>>print(a==b==c==d==e)
True
```

Hands On

Task1:

Create a list 1,2,3,”Ahmed”

Modify the third element to 50.2

Convert the list to tuple

Task2:

Create a dictionary

Modify the first value in the dictionary to “First”

Print the dictionary