Builtin

Baboo J. Cui

July 1, 2019

Contents

1 Basic Properties of Python

- code **block** is identified by 4 white space(not tab) following a line end up with ":"
- **comments** are start with "#", for multi-line case, use triple comma
- python is **case sensitive**

Here is a list of size issue:

- 1 Byte = 8 bit, bit is the smallest unit in computer
- size of ASCII is 1 Byte = 8bit, 256 cases
- size of Unicode is 2 Bytes = 16bit, 65536 cases
- to get size of an object in memory, use `sys.getsizeof(o)` function, note this will return the size of an object, which may be quite big

2 Keywords

There are 33 keywords in Python.

```
from keyword import kwlist
print(len(kwlist))  # 33
print(kwlist)
```

keyword list:

- True, False, not, or, and (5)
- def, pass, return, yield, lambda (5)
- if, else, elif, while, for, in, finally, continue, break (9)
- from, import, as (3)
- try, except, raise, assert (4)
- global, nonlocal (2)
- class, None (2)
- with, del, is (3)

3 Variable

Variables in python may be any type, naming a var should follow that:

- combination of letter, number and "_"
- cannot start with a number

Value can be assigned to a variable by "=" operator. Assignment do two things:

- create value in memory

- create variable name in memory and point it to corresponding value

To declare a constant, name the var with all capital letters(fake constant, just for reading). There are two types of methods for naming variables:

- camel-case
- underscore naming

just choose the one you like.

4 Operations

Python offers lots of basic operations:

- basic ones: "+, -, *, /"
- exponential: "**", or use **pow()** function
- floor division: "//", also known as whole part division
- reminder division: "%", to get the residue

5 Type

A type is a kind of data structure in python. To find the type of an object, simply use `type(o)`.

5.1 Duck Type

Python is weak type language. If an object looks like a type, then methods for that type can be applied to it. This is one of the most important feather in Python.

5.2 Type Coercion

Force to convert any type to a certain type, this may cause error and some complicate problem.

```
print(int("123"))  # 123
print(int(12.34))  # round off error 12
# print(int("abc123"))  # error
```

here is a list of important type coercion

- Boolean to int: True to 1 and False to 0
- int to Boolean: 0 to False and the others to True
- str to Boolean: empty str to False and non-empty str to True
- Boolean to str: True to "True" and False to "False"

5.3 Integer

it has type `int`:

- **infinite** range(different from others), should avoid too **big** number
- **hex** notation: add `0x` prefix, or use `hex()` to convert
- **binary** notation: add `0b` prefix, or use `bin()` to convert
- **oct** notation: add `0o` prefix, or use `oct()` to convert
- has method `bit_length()` that return the bit length of an int

5.4 Float

it has type `float`:

- **round off error** may occur
- **direct** notation: simply write the value
- **scientific** notation: **mantissa** exp **order of magnitude**

5.5 String

it has type `str`:

- declare by putting string literal in single or double quotes
- escape character: begin with \(backslash)

```
# commonly used
"\n"  # return
"\t"  # table
```

- coding: python code is in **UTF-8**, data in memory is in **unicode**
- direct string(simply what's inside): `r"..."`
- *binary string*(has type **bytes**, it's ASCII code,NOT str): `b"..."`, every letter is 1 byte

5.5.1 Coding Functions

Unicode is used for coding in python 3:

- `ord()`: convert string to Unicode
- `chr()`: convert Unicode to string

5.5.2 Encode and Decode

To covert a string literal to a certain code is called encoding, and the reverse process is call decoding. The common encoding methods are:

- "ascii"
- "utf-8"
- "gb2312"
- tip: for Unicode, use **ord()**

```
str.encoding()  # encoding a string to binary
bytes.decoding()  # decode binary to string, has parameter errors
                                ="ignore" to avoid error
```

5.5.3 Multiplication Between int and str

This will repeat the *str* for *int* times:

```
print("abc"*3)  # abcabcabc
```

5.6 Boolean

It has type `bool`, it has only two value: **True** and **False**(case sensitive). Operator on Boolean:

- **and**(&)
- **or**(|)
- **not**($\hat{()}$)

operation properties:

- priority order: not $>$ and $>$ or
- x or y: if x is true, return x, otherwise return y
- x and y: If x is true, return y, otherwise return x

5.7 None

None is a special value in Python to indicate empty.

5.8 List

5.9 Tuple

6 Control Flow

6.1 If

6.2 While

in while-else structure, else part will be execute if break statement in while is not executed. "while 1" is faster than "while True" because bool is a subclass of int, similarly "if x" is faster than "if x=True"

1. How to use python to run bash command?
   Use os.system("COMMAND")
2. How to format a decimal?
   Use {num:a.b f} where num is the position, a is num of integer bit, b is for decimal, f represent float
3. How to find length of a list?
   Use len() function, len(LIST) will return
4. How to find the last element of a list?
   Use LIST[-1], index of -1 represent the last element, -2 for the second last one, etc
5. Difference between list and tuple? List is mutable, declared by [ ] and tuple is immutable and declared by ()
6. List.append(o)?
   Append obj o to the end of list
7. List.insert(idx, o)?
   Insert obj o to index idx
8. List.pop() or list.pop(i)?
   Pop out the last element, and pop(i) pop the element with index i
9. Acquire the idx element of list?
   List[idx]
10. How to init a tuple with one element?
    T = (obj1,), the key is to add a comma after obj1
11. How to understand immutability of tuple?
    Each element that points to won't be mutable, but the content that each points may be mutable
12. Condition control structure?
    If else, or if elif else
13. How to loop an iterable object by for?
    Use for loop, for item in iterable_obj

14. How to loop by while?
    Use while structure: while condition: block_exe

15. How to generate a range?
    Use range() function, range(a) will generate [0...a-1] and range(a, b) will generate [a, a+1...b-1]

16. Difference between break and continue?
    Break will stop the loop and continue will simply jump to next loop, usually both are related to if condition

17. How to terminate a python program directly?
    Use ctrl + c to kill the process

18. What is dict?
    A key-value mechanism, also known as map in other languages

19. How to declare a dict?
    Format: {key1: val1, key2: val2, ...}, curly braces, comma separation, colon separation, key must be immutable

20. Why dict can find a key value so fast?
    Use binary tree mechanism, which lead to log(n) time scale

21. How to determine if a dict has a key?
    Use code: if key_val in dict, or: dict.get(key_val)

22. How to delete a key-value in dict?
    Use code: dict.pop(key)

23. Trade off between list and dict?
    List is slower but occupy smaller memory, dict is faster but requires more memory

24. How to create a set?
    Use curly braces:{item1, item2, ...} or use code set(list), which will turn a list to set

25. Properties of set?
    It has no order and there is no repeats

26. Add and remove key of a set?
    Corresponding to set.add(key) and set.remove(key)

27. Common operation on sets?
    Union "|" and intersection "&"

28. Difference between dict and set?
    Share the same mechanism but set doesn't have value, can't put mutable obj in

29. Is str mutable? What about list?
    Str is immutable and list is mutable

30. Immutable objects?
    Number, string and tuple. Contents in certain add can't be changed

31. How to find the memory address of an object?
    Use function id(), id(o) will return the memory add

32. How to get console args?
    By import sys, and then use sys.argv[idx]

33. What is configurative programming?
    A framework is created such that coding is like setting configuration

34. Mechanism of importing a module?
    Python interpreter just go through line by line(interview question)