数据结构与算法

Python 基础

张晓平

1 Python 高级特性

掌握了 Python 的数据类型、语句和函数，基本上就可以编写出很多有用的程序了。

例:
构造一个元素为1，3，5，7，...，99的列表.

```
L = []
n = 1
while n <= 99:
    L.append(n)
    n += 2
```

```
L = [n for n in range(100) if n % 2 == 1]
```

但是，Python 代码不是越多越好，而是越少越好；不是越复杂越好，而是越简单越好。

基于这一思想，这一节来介绍一些 Python 中非常有用的高级特性，能一行代码实现的功能，绝不写五行。请始终牢记，代码越少，开发效率越高。

1.1 切片

取一个 list 或 tuple 的部分元素是非常常见的操作。

例:
给定一个 list

```
L = ['apple', 'banana', 'cherry', 'grape', 'peach']
```

如何取出其前三个元素?

- 笨办法

```
r = [L[0], L[1], L[2]]
print(r)
```

- 使用循环

```
r = []
for i in range(3):
    r.append(L[i])
print(r)
```

- 使用切片 (slice)

```
r = L[0:3]
```

例:
给定一个 list

```
L = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

通过切片操作取出其中的部分元素。

- 正向切片

```
print( L[1:3] )
print( L[:3]  )
print( L[1:]  )
print( L[:]   )
```

```
['b', 'c']
['a', 'b', 'c']
['b', 'c', 'd', 'e', 'f', 'g']
['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

- 倒向切片

```
print( L[-4:-2] )
print( L[-4:]   )
print( L[:-2]   )
print( L[1:-1]  )
```

```
['d', 'e']
['d', 'e', 'f', 'g']
['a', 'b', 'c', 'd', 'e']
['b', 'c', 'd', 'e', 'f']
```

- 跳跃切片

```
print( L[1:-1:2] )
print( L[:-1:2] )
print( L[::5] )
```

```
['b', 'd', 'f']
['a', 'c', 'e']
['a', 'f']
```

例:
tuple 和 str 的切片

- tuple 的切片

```
t = (0, 1, 2, 3, 4, 5)
print( t[:3] )
print( ('a', 'b', 'c', 'd')[1::2] )
```

```
(0, 1, 2)
('b', 'd')
```

- str 的切片

```
print( 'abcdefgh'[:3] )
print( 'abcdefgh'[::3] )
```

```
abc
adg
```

1.2 迭代

如果给定一个 list 或 tuple，我们可以通过for循环来遍历这个 list 或 tuple，这种遍历称为迭代（Iteration）。

只要是可迭代对象，无论有无下标，都可以迭代。

例:
dict 没有下标，但它也可以迭代：

```
d = {'a': 1, 'b': 2, 'c': 3}
for key in d:
    print(key)
```

```
a
b
c
```

因为 dict 的存储不是顺序排列的，所以迭代出的结果顺序可能会不一样。

例:
因 str 也是可迭代对象，它也可用for循环进行迭代：

```
for ch in 'abcd':
    print(ch)
```

```
a
b
c
d
```

当我们使用for循环时，只要作用于一个可迭代对象，for循环就可以正常运行，而我们不太关心该对象究竟是 list 还是其他数据类型。

1.2.1 如何判断一个对象是可迭代对象？

可通过 collections 模块的 Iterable 类型判断：

```
from collections import Iterable
print( isinstance('abc', Iterable) )
print( isinstance([], Iterable) )
print( isinstance((), Iterable) )
print( isinstance({}, Iterable) )
print( isinstance({1}, Iterable) )
print( isinstance(123, Iterable) )
```

```
True
True
True
True
True
False
```

1.2.2 enumerate()函数

Python 内置的enumerate()可以把一个 list 变成索引-元素对，这样就可以在for循环中同时迭代索引和元素本身：

```
for i, value in enumerate(['a', 'b', 'c']):
    print(f"{i}: {value}")
```

```
0: a
1: b
2: c
```

```
for i, value in enumerate(('a', 'b', 'c')):
    print(f"{i}: {value}")
```

```
0: a
1: b
2: c
```

```
for i, value in enumerate('abc'):
    print(f"{i}: {value}")
```

```
0: a
1: b
2: c
```

以上代码中，for循环同时引用两个变量，这在 Python 中非常常见，如

```
for x, y in [[1, 1], [2, 4], [3, 9]]:
    print(f'{x}, {y}')
```

```
1, 1
2, 4
3, 9
```

1.3 列表生成式

列表生成式即 List Comprehensions，是 Python 内置的非常简单却强大的可以用来创建 list 的生成式。

例:
要生成列表[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]，可用list(range(1, 11))。但要生成[1, 4, 9, ..., 100]呢?

1. 用循环（过于繁琐）

```
L = []
for x in range(1, 11):
    L.append(x * x)
print(L)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81,
100]
```

2. 列表生成式

```
L = [x * x for x in range(1, 11)]
print(L)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81,
100]
```

写列表生成式时，for循环后还可以加上if判断。

例:
筛选出偶数的平方。

```
L = [x * x for x in range(1, 11)
if x % 2 == 0]
print(L)
```

```
[4, 16, 36, 64, 100]
```

还可以使用两层循环来得到全排列（三层及以上的循环很少用到）

例:

```
L = [m + n for m in 'ABC' for n in
 'XYZ']
print(L)
```

```
['AX', 'AY', 'AZ', 'BX', 'BY', 'BZ', '
CX', 'CY', 'CZ']
```

1.3.1 列表生成式的应用

例:
利用os模块列出当前目录下的所有文件和目录名

```
dirs = [d for d in os.listdir('.')]
print(dirs)
```

```
['lec01.out', 'lec01.log', 'src', 'tmp.out', 'lec01.aux', 'lec01.tex', 'asset
', 'lec01.pdf', '.DS_Store', 'makefile']
```

例:
将一个 list 中的字符串变为小写

```
L = ['Hello', 'Wuhan', 'University']
l = [s.lower() for s in L]
print(l)
```

```
['hello', 'wuhan', 'university']
```

例:
将字典{'a': 1, 'b': 2, 'c': 3}转换为列表的形式['a=1', 'b=2', 'c=3']

```
d = {'a': 1, 'b': 2, 'c': 3}
l = [k + '=' + str(v) for k, v in d.items()]
print(l)
```

```
['a=1', 'b=2', 'c=3']
```

1.4 生成器

1.5 迭代器

1.6 习题

1. 编写函数min_and_max(L)，求一个 list 的最小值和最大值，并以 tuple 将它们返回。