Python3.x 和 Python2.x 的区别

3. 语法

1）去除了<>，全部改用!=

2）去除``，全部改用 repr()

```
>>> s = 'Hello, world.'
>>> repr(s)     "'Hello, world.'"      repr() 转化为供解释器读取的形式
```

3）关键词加入 as 和 with，还有 True,False,None

4）整型除法返回浮点数，要得到整型结果，请使用//

5）加入 nonlocal 语句。使用 noclocal x 可以直接指派外围（非全局）变量

nonlocal 关键字用来在函数或其他作用域中使用外层(非全局)变量

6）去除 print 语句，加入 print()函数实现相同的功能。同样的还有 exec 语句，已经改为 exec()函数

例如：

```
2.X: print "The answer is", 2*2
3.X: print("The answer is", 2*2)
2.X: print x,                    # 使用逗号结尾禁止换行
3.X: print(x, end=" ")           # 使用空格代替换行
2.X: print                       # 输出新行
3.X: print()                     # 输出新行
2.X: print >>sys.stderr, "fatal error"
3.X: print("fatal error", file=sys.stderr)
2.X: print (x, y)                # 输出 repr((x, y))
3.X: print((x, y))               # 不同于 print(x, y)!
```

7）改变了顺序操作符的行为，例如 x<y，当 x 和 y 类型不匹配时抛出 TypeError 而不是返回随即的 bool 值

8）输入函数改变了，删除了 raw_input，用 input 代替：

```
2.X:guess = int(raw_input('Enter an integer : ')) # 读取键盘输入的方法
3.X:guess = int(input('Enter an integer : '))
```

9）去除元组参数解包。不能 def(a, (b, c)):pass 这样定义函数了

10）新式的 8 进制字变量，相应地修改了 oct()函数。

2.X 的方式如下：

```
>>> 0666
438
>>> oct(438)
'0666'
```

3.X 这样：

```
>>> 0666
SyntaxError: invalid token (<pyshell#63>, line 1)
>>> 0o666
438
>>> oct(438)
'0o666'
```

11）增加了 2 进制字面量和 bin()函数

```
>>> bin(438)
'0b110110110'
>>> _438 = '0b110110110'
>>> _438
'0b110110110'
```

12）扩展的可迭代解包。在 Py3.X 里，a, b, *rest = seq 和 *rest, a = seq 都是合法的，只要求两点：rest 是 list
对象和 seq 是可迭代的。

13）新的 super()，可以不再给 super()传参数，

```
>>> class C(object):
    def __init__(self, a):
      print('C', a)
>>> class D(C):
    def __init(self, a):
      super().__init__(a) # 无参数调用 super()
>>> D(8)
C 8
<__main__.D object at 0x00D7ED90>
```

14）新的 metaclass 语法：

```
class Foo(*bases, **kwds):
 pass
```

15）支持 class decorator。用法与函数 decorator 一样：

```
>>> def foo(cls_a):
    def print_func(self):
      print('Hello, world!')
    cls_a.print = print_func
    return cls_a
>>> @foo
class C(object):
 pass
>>> C().print()
Hello, world!
```

class decorator 可以用来玩玩狸猫换太子的大把戏。更多请参阅 PEP 3129

4. 字符串和字节串

1）现在字符串只有 str 一种类型，但它跟 2.x 版本的 unicode 几乎一样。

2）关于字节串，请参阅“数据类型”的第 2 条目

5.数据类型

1）Py3.X 去除了 long 类型，现在只有一种整型——int，但它的行为就像 2.X 版本的 long

2）新增了 bytes 类型，对应于 2.X 版本的八位串，定义一个 bytes 字面量的方法如下：

```
>>> b = b'china'
>>> type(b)
<type 'bytes'>
```

str 对象和 bytes 对象可以使用.encode() (str -> bytes) or .decode() (bytes -> str)方法相互转化。

```
>>> s = b.decode()
```

```
>>> s
'china'
>>> b1 = s.encode()
>>> b1
b'china'
```

3）dict 的.keys()、.items 和.values()方法返回迭代器，而之前的 iterkeys()等函数都被废弃。同时去掉的还有

dict.has_key()，用 in 替代它吧

6.面向对象

1）引入抽象基类（Abstraact Base Classes，ABCs）。

2）容器类和迭代器类被 ABCs 化，所以 cellections 模块里的类型比 Py2.5 多了很多。

```
>>> import collections
>>> print('\n'.join(dir(collections)))
Callable
Container
Hashable
ItemsView
Iterable
Iterator
KeysView
Mapping
MappingView
MutableMapping
MutableSequence
MutableSet
NamedTuple
Sequence
Set
Sized
ValuesView
__all__
__builtins__
__doc__
__file__
__name__
_abcoll
_itemgetter
_sys
defaultdict
deque
```

另外，数值类型也被 ABCs 化。关于这两点，请参阅 PEP 3119 和 PEP 3141。

3)迭代器的 next()方法改名为__next__()，并增加内置函数 next()，用以调用迭代器的__next__()方法

4）增加了@abstractmethod 和 @abstractproperty 两个 decorator，编写抽象方法（属性）

更加方便。

7.异常

1）所以异常都从 BaseException 继承，并删除了 StardardError

2）去除了异常类的序列行为和.message 属性

3）用 raise Exception(args)代替 raise Exception, args 语法

4）捕获异常的语法改变，引入了 as 关键字来标识异常实例，在 Py2.5 中：

```
>>> try:
...     raise NotImplementedError('Error')
... except NotImplementedError, error:
...     print error.message
...
Error
```

在 Py3.0 中：

```
>>> try:
        raise NotImplementedError('Error')
      except NotImplementedError as error: #注意这个 as
        print(str(error))
Error
```

5）异常链，因为__context__在 3.0a1 版本中没有实现

8.模块变动

1）移除了 cPickle 模块，可以使用 pickle 模块代替。最终我们将会有一个透明高效的模块。

2）移除了 imageop 模块

3）移除了 audiodev, Bastion, bsddb185, exceptions, linuxaudiodev, md5, MimeWriter, mimify, popen2,
rexec, sets, sha, stringold, strop, sunaudiodev, timing 和 xmllib 模块

4）移除了 bsddb 模块(单独发布，可以从 http://www.jcea.es/programacion/pybsddb.htm 获取)

5）移除了 new 模块

6）os.tmpnam()和 os.tmpfile()函数被移动到 tmpfile 模块下

7）tokenize 模块现在使用 bytes 工作。主要的入口点不再是 generate_tokens，而是 tokenize.tokenize()

9.其它

1）xrange() 改名为 range()，要想使用 range()获得一个 list，必须显式调用：

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

2）bytes 对象不能 hash，也不支持 b.lower()、b.strip()和 b.split()方法，但对于后两者可以使用 b.strip(b’
\n\t\r \f’)和 b.split(b’ ‘)来达到相同目的

3）zip()、map()和 filter()都返回迭代器。而 apply()、 callable()、coerce()、 execfile()、reduce()和 reload
()函数都被去除了
现在可以使用 hasattr()来替换 callable(). hasattr()的语法如：hasattr(string, '__name__')

4）string.letters 和相关的.lowercase 和.uppercase 被去除，请改用 string.ascii_letters 等

5）如果 x < y 的不能比较，抛出 TypeError 异常。2.x 版本是返回伪随机布尔值的

6）__getslice__ 系列成员被废弃。a[i:j] 根据上下文转换为 a.__getitem__(slice(I, j)) 或

__setitem__和
__delitem__调用
7）file 类被废弃，在 Py2.5 中：

```
>>> file
<type 'file'>
```

在 Py3.X 中：

```
>>> file
Traceback (most recent call last):
File "<pyshell#120>", line 1, in <module>
  file
NameError: name 'file' is not defined
```