常见系统模块

为了方便程序员开发代码，Python提供了很多内置的模块给程序员用来提高编码效率。常见的内置模块有:

- os模块
- sys模块
- math模块
- random模块
- datetime模块
- time模块
- calendar模块
- hashlib模块
- hmac模块
- copy模块
- uuid模块

OS模块

OS全称OperationSystem,即操作系统模块，这个模块可以用来操作系统的功能，并且实现跨平台操作。

```
import os
os.getcwd()  # 获取当前的工作目录，即当前python脚本工作的目录
os.chdir('test') # 改变当前脚本工作目录，相当于shell下的cd命令
os.rename('毕业论文.txt','毕业论文-最终版.txt') # 文件重命名
os.remove('毕业论文.txt') # 删除文件
os.rmdir('demo')  # 删除空文件夹
os.removedirs('demo') # 删除空文件夹
os.mkdir('demo')  # 创建一个文件夹
os.chdir('C:\\') # 切换工作目录
os.listdir('C:\\') # 列出指定目录里的所有文件和文件夹
os.name # nt->widonws posix->Linux/Unix或者MacOS
os.environ # 获取到环境配置
os.environ.get('PATH') # 获取指定的环境配置

os.path.abspath(path) # 获取Path规范会的绝对路径
os.path.exists(path)  # 如果Path存在，则返回True
os.path.isdir(path)  # 如果path是一个存在的目录，返回True。否则返回False
os.path.isfile(path) # 如果path是一个存在的文件，返回True。否则返回False
os.path.splitext(path)  # 用来将指定路径进行分隔，可以获取到文件的后缀名
```

sys模块

该模块提供对解释器使用或维护的一些变量的访问，以及与解释器强烈交互的函数。

```
import sys
sys.path # 模块的查找路径
sys.argv # 传递给Python脚本的命令行参数列表
sys.exit(code) # 让程序以指定的退出码结束

sys.stdin # 标准输入。可以通过它来获取用户的输入
sys.stdout # 标准输出。可以通过修改它来百变默认输出
sys.stderr # 错误输出。可以通过修改它来改变错误删除
```

math模块

math模块保存了数学计算相关的方法，可以很方便的实现数学运算。

```
import math
print(math.fabs(-100)) # 取绝对值
print(math.ceil(34.01))  #向上取整
print(math.factorial(5)) # 计算阶乘
print(math.floor(34.98))  # 向下取整
print(math.pi)   # π的值，约等于 3.141592653589793
print(math.pow(2, 10)) # 2的10次方
print(math.sin(math.pi / 6))  # 正弦值
print(math.cos(math.pi / 3))  # 余弦值
print(math.tan(math.pi / 2))  # 正切值
```

random模块

random 模块主要用于生成随机数或者从一个列表里随机获取数据。

```
print(random.random())  # 生成 [0,1)的随机浮点数
print(random.uniform(20, 30))  # 生成[20,30]的随机浮点数
print(random.randint(10, 30))  # 生成[10,30]的随机整数
print(random.randrange(20, 30))  # 生成[20,30)的随机整数
print(random.choice('abcdefg'))  # 从列表里随机取出一个元素
print(random.sample('abcdefghij', 3)) # 从列表里随机取出指定个数的元素
```

> 练习:
>
> 定义一个函数，用来生成由数字和字母组成的随机验证码。该函数需要一个参数，参数用来指定验证码的长度。

datetime模块

datetime模块主要用来显示日期时间，这里主要涉及 `date` 类，用来显示日期； `time` 类，用来显示时

间； `dateteime` 类，用来显示日期时间; `timedelta` 类用来计算时间。

```
import datetime
print(datetime.date(2020, 1, 1))  # 创建一个日期
print(datetime.time(18, 23, 45)) # 创建一个时间
print(datetime.datetime.now())  # 获取当前的日期时间
print(datetime.datetime.now() + datetime.timedelta(3))  # 计算三天以后的日期时间
```

time模块

除了使用datetime模块里的time类以外，Python还单独提供了另一个time模块，用来操作时间。time模块不仅可以用来显示时间，还可以控制程序，让程序暂停(使用sleep函数)

```
print(time.time())  # 获取从1970-01-01 00:00:00 UTC 到现在时间的秒数
print(time.strftime("%Y-%m-%d %H:%M:%S")) # 按照指定格式输出时间
print(time.asctime()) #Mon Apr 15 20:03:23 2019
print(time.ctime()) # Mon Apr 15 20:03:23 2019

print('hello')
print(time.sleep(10)) # 让线程暂停10秒钟
print('world')
```

calendar模块

calendar模块用来显示一个日历，使用的不多，了解即可。

```
calendar.setfirstweekday(calendar.SUNDAY) # 设置每周起始日期码。周一到周日分别对应 0 ~ 6
calendar.firstweekday()# 返回当前每周起始日期的设置。默认情况下，首次载入calendar模块时返回0，即星期一。
c = calendar.calendar(2019)  # 生成2019年的日历，并且以周日为其实日期码
print(c)  #打印2019年日历
print(calendar.isleap(2000)) # True.闰年返回True,否则返回False
count = calendar.leapdays(1996,2010) # 获取1996年到2010年一共有多少个闰年
print(calendar.month(2019, 3))  # 打印2019年3月的日历
```

hashlib模块

hashlib是一个提供字符加密功能的模块，包含MD5和SHA的加密算法，具体支持md5,sha1, sha224, sha256, sha384, sha512等算法。该模块在用户登录认证方面应用广泛，对文本加密也很常见。

```
import hashlib

# 待加密信息
str = '这是一个测试'

# 创建md5对象
hl = hashlib.md5('hello'.encode(encoding='utf8'))
print('MD5加密后为 ：' + hl.hexdigest())

h1 = hashlib.sha1('123456'.encode())
print(h1.hexdigest())
h2 = hashlib.sha224('123456'.encode())
print(h2.hexdigest())
h3 = hashlib.sha256('123456'.encode())
print(h3.hexdigest())
h4 = hashlib.sha384('123456'.encode())
print(h4.hexdigest())
```

hmac模块

HMAC算法也是一种一种单项加密算法，并且它是基于上面各种哈希算法/散列算法的，只是它可以在运算过程中使用一个密钥来增增强安全性。hmac模块实现了HAMC算法，提供了相应的函数和方法，且与hashlib提供的api基本一致。

```
h = hmac.new('h'.encode(),'你好'.encode())
result = h.hexdigest()
print(result)  # 获取加密后的结果
```

copy模块

copy模块里有copy和deepcopy两个函数，分别用来对数据进行深复制和浅复制。

```
import copy

nums = [1, 5, 3, 8, [100, 200, 300, 400], 6, 7]
nums1 = copy.copy(nums)  # 对nums列表进行浅复制
nums2 = copy.deepcopy(nums)  # 对nums列表进行深复制
```

uuid模块

UUID是128位的全局唯一标识符，通常由32字节的字母串表示，它可以保证时间和空间的唯一性，也称为GUID。通过MAC地址、时间戳、命名空间、随机数、伪随机数来保证生产的ID的唯一性。随机生成字符串，可以当成token使用，当成用户账号使用，当成订单号使用。

方法	作用
uuid.uuid1()	基于MAC地址，时间戳，随机数来生成唯一的uuid，可以保证全球范围内的唯一性。
uuid.uuid2()	算法与uuid1相同，不同的是把时间戳的前4位置换为POSIX的UID。不过需要注意的是python中没有基于DCE的算法，所以python的uuid模块中没有uuid2这个方法。
uuid.uuid3(namespace,name)	通过计算一个命名空间和名字的md5散列值来给出一个uuid，所以可以保证命名空间中的不同名字具有不同的uuid，但是相同的名字就是相同的uuid了。namespace并不是一个自己手动指定的字符串或其他量，而是在uuid模块中本身给出的一些值。比如uuid.NAMESPACE*DNS，uuid.NAMESPACE*OID，uuid.NAMESPACE_OID这些值。这些值本身也是UUID对象，根据一定的规则计算得出。
uuid.uuid4()	通过伪随机数得到uuid，是有一定概率重复的
uuid.uuid5(namespace,name)	和uuid3基本相同，只不过采用的散列算法是sha1

一般而言，在对uuid的需求不是很复杂的时候，uuid1或者uuid4方法就已经够用了，使用方法如下：

```
import uuid

print(uuid.uuid1())  # 根据时间戳和机器码生成uuid,可以保证全球唯一
print(uuid.uuid4())  # 随机生成uuid,可能会有重复

# 使用命名空间和字符串生成uuid.
# 注意一下两点:
# 1. 命名空间不是随意输入的字符串，它也是一个uuid类型的数据
# 2. 相同的命名空间和想到的字符串，生成的uuid是一样的
print(uuid.uuid3(uuid.NAMESPACE_DNS, 'hello'))
print(uuid.uuid5(uuid.NAMESPACE_OID, 'hello'))
```