Python标准库（内置模块）

■random模块

|  |  |
| --- | --- |
| 方法 | 功能 |
| random.random() | 生成[0, 1) 之间的随机浮点数（包括0，不包括1） |
| random.randrange(a, b) | 生成[a, b) 之间的随机整数（包括a，不包括b） |
| random.randint(a, b) | 生成[a, b] 之间的随机整数（包括a和b）  等价于 random.randrange(a, b + 1) |
| random.uniform(a, b) | 生成 [a, b] 之间的随机浮点数（包括a和b） |
| random.choice(seq) | 从序列（有序数据）里随机选择一个数据 |
| random.choices(population, k) | 从序列里随机选择k个数据（允许重复） |
| random.sample(population, k) | 从序列里随机选择k个数据（不允许重复） |
| random.shuffle(seq) | 让序列重新排序 |

import random  
  
print(random.random())  
# 输出：[0, 1) 随机浮点数  
  
print(random.randrange(1, 10))  
# 输出：[1, 10) 随机整数  
  
print(random.randint(1, 10))  
# 输出：[1, 10] 随机整数，等价于print(random.randrange(1, 11))  
  
print(random.uniform(1, 10))  
# 输出：[1, 10] 随机浮点数。  
  
list1 = [1, 2, 3, 4, 5]  
  
print(random.choice(list1))  
# 输出：从list1中随机取出一个数据  
  
print(random.choices(list1, k=10))  
# 输出：从list1中随机选择10个可重复数据。list1数据不满10个，可重复取。  
  
print(random.sample(list1, k=3))  
# 输出：从list1中随机选择3个不可重复数据。注意：k值如果大于序列长度，则报错，因为不能重复取。  
  
print(list1)  
# 输出：[1, 2, 3, 4, 5]  
random.shuffle(list1)  
print(list1)  
# 输出：[5, 3, 1, 4, 2] 被shuffle重新排序了

■math模块

import math  
  
# floor向下取整  
print(math.floor(3.89))  
# 输出：3  
  
# ceil向上取整  
print(math.ceil(3.01))  
# 输出：4  
  
# pow求幂运算  
print(math.pow(2, 3))  
# 输出：8.0  
  
# sqrt开平方  
print(math.sqrt(9))  
# 输出：3.0  
  
# factorial求阶乘  
print(math.factorial(5))  
# 输出：120

■OS模块

import os  
  
  
# os.name, windows系统返回nt（NewTechnology）, 非windows系统（Unix或类Unix系统等）返回posix  
print(os.name)  
# 输出：nt  
  
# os.sep, windows系统返回\, 非windows系统返回 /  
print(os.sep)  
# 输出：\  
  
  
# 路径的三种表示方法：  
  
path1 = r'E:\computersContent\projects\pythonlearn1'  
path2 = 'E:\\computersContent\\projects\\pythonlearn1'  
path3 = 'E:/computersContent/projects/pythonlearn1'  
  
  
# 获取当前文件所在路径  
print(os.getcwd())  
# 输出：E:\computersContent\projects\pythonlearn1  
  
  
# 获取当前目录内的文件列表，不传参默认当前路径  
print(os.listdir())  
# 输出：['.idea', 'main.py', 'my\_module1.py', 'my\_module2.py', 'test1.py', 'testtxt', 'test\_package', 'venv', '\_\_pycache\_\_']  
  
  
# 切换路径  
os.chdir('../')  
print(os.getcwd())  
# 输出：E:\computersContent\projects  
  
# 再切回来  
os.chdir(r'E:\computersContent\projects\pythonlearn1')  
print(os.getcwd())  
# 输出：E:\computersContent\projects\pythonlearn1  
  
  
# 获取当前环境变量  
print(os.environ)  
  
  
# 获取Path规范的绝对路径  
print(os.path.abspath(\_\_file\_\_))  
# 输出：E:\computersContent\projects\pythonlearn1\test1.py  
  
  
# 判断路径是否存在  
print(os.path.exists('E:/computersContent/projects/pythonlearn1'))  
# 输出：True  
  
  
# 判断文件夹是否存在  
print(os.path.isdir('E:/computersContent/projects'))  
# 输出：True  
  
  
# 判断文件是否存在  
print(os.path.isfile('E:/computersContent/projects/pythonlearn1/test1.py'))  
# 输出：True  
  
  
# 分割文件名和扩展名  
print(os.path.splitext('abc\_123.3\_bc.ac2.txt'))  
# 输出：('abc\_123.3\_bc.ac2', '.txt')

■time模块

import time  
  
# 获取当前时间戳  
print(time.time())  
# 输出：1674204073.11312  
  
print(time.ctime())  
# 输出：Fri Jan 20 16:41:52 2023  
  
print(time.gmtime())  
# 输出：time.struct\_time(tm\_year=2023, tm\_mon=1, tm\_mday=20, tm\_hour=8, tm\_min=42, tm\_sec=51, tm\_wday=4, tm\_yday=20, tm\_isdst=0)  
  
print(time.gmtime().tm\_year)  
# 输出：2023  
  
  
# strftime时间转字符串  
# strptime字符串转时间  
  
x = time.strftime('%Y-%m-%d %H:%M:%S UTC+8', (2021, 1, 28, 10, 53, 21, 3, 28, 1))  
print(x)  
# 输出：2021-01-28 10:53:21 UTC+8  
  
y = time.strptime('2021-01-28 10:53:21', '%Y-%m-%d %H:%M:%S')  
print(y)  
# 输出：time.struct\_time(tm\_year=2021, tm\_mon=1, tm\_mday=28, tm\_hour=10, tm\_min=53, tm\_sec=21, tm\_wday=3, tm\_yday=28, tm\_isdst=-1)  
  
# 睡眠  
time.sleep(5)

●注意：

time模块一般只有time.time获取时间戳和time.sleep睡眠功能常用。

其他的获取日期时间等功能可以通过datetime模块来简单实现

■datetime模块

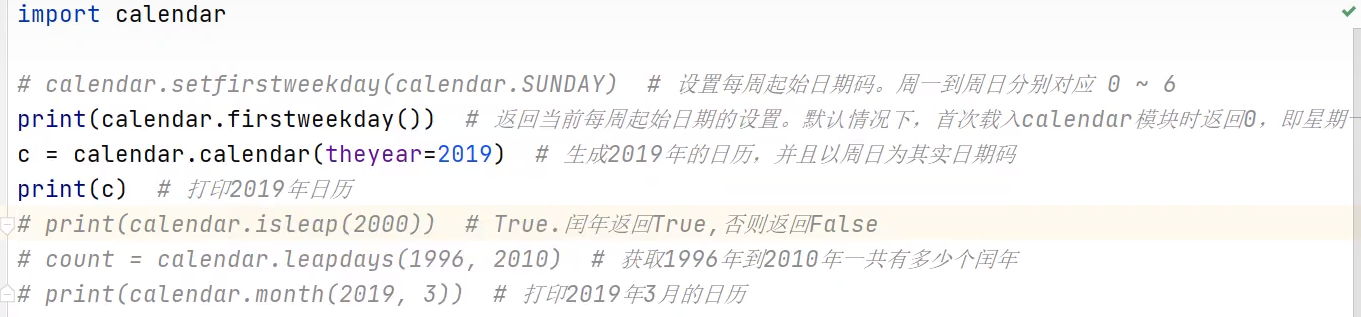
import datetime  
  
  
# datetime模块有四个比较重要的类型：  
# date类：用来表示日期  
# time类：用来表示时间  
# datetime类：用来表示日期时间  
# timedelta类：用来计算日期时间  
  
  
# date类  
# 获取一个date类型的对象  
today = datetime.date.today()  
# 然后取值  
print(today.year)  
print(today.month)  
print(today.day)  
print(today.weekday())  
"""  
输出：  
2023  
1  
20  
4（星期几）  
"""  
  
  
# time类  
# 获取一个time类型的对象  
now = datetime.datetime.now()  
# 然后可以获得日期对象  
now\_date = now.date()  
print(now\_date.year)  
print(now\_date.month)  
print(now\_date.day)  
print(now\_date.weekday())  
"""  
2023  
1  
20  
4  
"""  
  
# 也可以获得一个时间对象  
now\_time = now.time()  
print(now\_time.hour)  
print(now\_time.minute)  
print(now\_time.second)  
"""  
输出：  
17  
10  
48  
"""  
  
  
# 用于计算时间，比如100小时之后是什么日期时间  
print(now + datetime.timedelta(hours=100))  
# 输出：2023-01-24 21:12:10.950652

●综合应用时间戳互转

import time  
import datetime  
  
  
t = time.time() # 获取当前时间戳  
print(t)  
  
tt = datetime.datetime.fromtimestamp(t) # 时间戳转成日期或时间对象  
n\_date = tt.date()  
print(n\_date.year)

■calendar模块

用的不多



■copy模块

常见copy的问题

# 情况：当a修改元素时，list1也会被修改，因为a是一个list1的引用  
list1 = [1, 2, 3, 4, 5]  
a = list1  
a[0] = 9  
print(list1)  
# 输出：[9, 2, 3, 4, 5]  
  
# 通过列表的copy方法来避面污染  
list2 = [1, 2, 3, 4, 5]  
b = list2.copy()  
b[0] = 9  
print(list2)  
# 输出：[1, 2, 3, 4, 5]没有被b污染  
print(b)  
# 输出：[9, 2, 3, 4, 5]

以上例子可以使用列表自带的copy方法来解决。但有的数据类型没有copy方法，就需要使用copy模块来解决

import copy  
  
list1 = [1, 2, 3, 4, 5]  
a = copy.copy(list1)  
a[0] = 9  
print(a)  
# 输出：[9, 2, 3, 4, 5]  
print(list1)  
# 输出：[1, 2, 3, 4, 5]list1没有被a污染

●注意：copy.copy是浅拷贝。如下例子，浅拷贝只拷贝第一层，遇到list1[2]这样的嵌套，那么对于嵌套里面的内容仍然是引用而不是复制。

list1 = [1, 2, [2.1, 2.2, 2.3], 4, 5]  
a = copy.copy(list1)  
a[2][0] = 2.9  
print(a)  
# 输出：[1, 2, [2.9, 2.2, 2.3], 4, 5]  
print(list1)  
# 输出：[1, 2, [2.9, 2.2, 2.3], 4, 5]由于是浅拷贝，所以list1[2]里面的元素被污染

这种情况就需要用copy.deepcopy深拷贝了

list1 = [1, 2, [2.1, 2.2, 2.3], 4, 5]  
a = copy.deepcopy(list1)  
a[2][0] = 2.9  
print(a)  
# 输出：[1, 2, [2.9, 2.2, 2.3], 4, 5]  
print(list1)  
# 输出：[1, 2, [2.1, 2.2, 2.3], 4, 5]由于是深拷贝，所以list1[2]里面的元素没有被污染

■uuid模块



import uuid  
  
print(uuid.uuid1())  
print(uuid.uuid4())  
print(uuid.uuid3(uuid.NAMESPACE\_DNS, 'zhangsan'))  
print(uuid.uuid5(uuid.NAMESPACE\_DNS, 'zhangsan'))  
  
"""  
输出：  
1a76b990-98a9-11ed-9d72-2c6e8538b624  
309fd0b1-72e4-40fa-a2e7-3a14e9696f7e  
d51a47a6-3157-39a9-94d3-70ecb9fda86e  
8e117ce0-eb96-5ae8-84b7-621521f751b6  
"""