nsd1912-py02-day01

时间

time模块

时间表示方法

- 时间戳：从1970年1月1日00:00:00开始按秒计算的偏移量
- UTC时间：世界协调时
- struct_time：九元组时间表示方式，(年，月，日，时，分，秒，一周中的第几天，一年中的第几天，是否使用夏季节约时间)

```
>>> import time
>>> time.time()    # 返回当前时间的时间戳
1590197012.924316
>>> time.ctime()   # 返回当前UTC时间的字符串
'Sat May 23 09:24:13 2020'
>>> time.localtime()
time.struct_time(tm_year=2020, tm_mon=5, tm_mday=23, tm_hour=9, tm_min=24, tm_sec=47,
tm_wday=5, tm_yday=144, tm_isdst=0)
>>> t1 = time.localtime()  # 九元组时间
>>> t1.tm_year
2020
>>> t1.tm_wday
5
>>> t1.tm_yday
144

>>> time.sleep(3)   # 睡眠
>>> time.strftime('%Y-%m-%d %a %H:%M:%S')
'2020-05-23 Sat 09:35:21'
# 将字符串格式的时间转换成 9元组时间
>>> time.strptime('2020-05-23 09:35:21', '%Y-%m-%d %H:%M:%S')
time.struct_time(tm_year=2020, tm_mon=5, tm_mday=23, tm_hour=9, tm_min=35, tm_sec=21,
tm_wday=5, tm_yday=144, tm_isdst=-1)
```

datetime模块

```
>>> import datetime
>>> datetime.datetime.now()
datetime.datetime(2020, 5, 23, 10, 47, 9, 752103)
# 或
>>> from datetime import datetime
>>> datetime.now()
datetime.datetime(2020, 5, 23, 10, 47, 36, 645495)

>>> t = datetime.now()
>>> t   # 内容是年月日时分秒毫秒
datetime.datetime(2020, 5, 23, 10, 48, 37, 535224)
>>> t.year, t.month, t.day, t.hour, t.minute, t.second, t.microsecond
(2020, 5, 23, 10, 48, 37, 535224)
>>> t.ctime()
'Sat May 23 10:48:37 2020'

>>> t.strftime('%Y/%m/%d %H:%M:%S')
'2020/05/23 10:48:37'
# 将时间字符串转成datetime对象
>>> datetime.strptime('2020-05-23 09:35:21', '%Y-%m-%d %H:%M:%S')
datetime.datetime(2020, 5, 23, 9, 35, 21)
```

- 计算时间差

```
>>> from datetime import datetime, timedelta
>>> t = datetime.now()
>>> days = timedelta(days=100, hours=1)
>>> t
datetime.datetime(2020, 5, 23, 11, 25, 7, 414768)
>>> t - days  # 100天1小时之前的时间
datetime.datetime(2020, 2, 13, 10, 25, 7, 414768)
>>> t + days  # 100天1小时之后的时间
datetime.datetime(2020, 8, 31, 12, 25, 7, 414768)
```

异常处理

- 异常就是不正常。当程序运行时，如果出现错误，程序将会崩溃终止执行，在屏幕上抛出异常
- 异常处理就是提前想到程序可能出现的问题，然后编写出现问题的补救代码，使得程序不要崩溃，可以继续向下执行。
- 异常语法结构

```
try:
    有可能发生异常的语句
except 异常1:
    异常1发生时执行的语句
except 异常2:
    异常2发生时执行的语句
except (异常3, 异常4):
    异常3或4发生时执行的语句
else:
    异常不发生才执行的语句
finally:
    不管异常是否发生都要执行的语句
```

触发异常

- 通过raise语句触发指定的异常

```
def set_info(name, age):
    if not 1 <= age <= 120:
        raise ValueError('年龄超过范围(1~120)')
    print('%s is %s years old.' % (name, age))

if __name__ == '__main__':
    set_info('牛老师', 200)
```

- 通过assert关键字实现断言异常，assert后面的表示式如果为假，那么一定会出现AssertionError异常

```
def set_info(name, age):
    if not 1 <= age <= 120:
        raise ValueError('年龄超过范围(1~120)')
    print('%s is %s years old.' % (name, age))

def set_info2(name, age):
    assert 1 <= age <= 120, '年龄超过范围(1~120)'
    print('%s is %s years old.' % (name, age))

if __name__ == '__main__':
    try:
        set_info('牛老师', 200)
    except ValueError as e:
        print('Error:', e)

    set_info2('庞老师', 188)
```

os模块

- os模块是python访问文件系统的主要模块

```
>>> import os
>>> os.getcwd()   # pwd
'/root/nsd2019/nsd1912/py02/day01'
>>> os.listdir()  # ls
>>> os.mkdir('/tmp/abc')  # mkdir /tmp/abc
>>> os.makedirs('/tmp/nsd1912/demo')  # mkdir -p
>>> os.listdir('/tmp/nsd1912')   # ls /tmp/nsd1912
>>> os.chdir('/tmp/nsd1912/demo')  # cd /tmp/nsd1912/demo
>>> os.getcwd()
'/tmp/nsd1912/demo'
>>> os.mknod('mytest.txt')   # touch mytest.txt
>>> os.listdir()
['mytest.txt']
>>> os.symlink('/etc/hosts', 'zhuji')  # ln -s

>>> os.chmod('mytest.txt', 0o755)  # chmod 755
>>> os.chmod('mytest.txt', 420)    # chmod 644
>>> os.environ    # env
>>> os.environ['HOSTNAME']
'localhost.localdomain'
>>> os.environ['HOME']
'/root'
>>> os.rename('mytest.txt', 'mytest.md')  # mv
>>> os.remove('mytest.md')  # rm
>>> os.rmdir('/tmp/abc')    # rmdir 只能删除空目录

>>> st = os.stat('/etc/hosts')  # stat /etc/hosts
>>> st.st_uid
0
>>> st.st_atime
1590195123.220157
>>> import time
>>> time.ctime(st.st_atime)
'Sat May 23 08:52:03 2020'

>>> os.path.abspath('zhuji')  # 返回绝对路径
'/tmp/nsd1912/demo/zhuji'
>>> os.path.basename('/tmp/nsd1912/demo/abc.txt')
'abc.txt'
>>> os.path.dirname('/tmp/nsd1912/demo/abc.txt')
'/tmp/nsd1912/demo'
>>> os.path.split('/tmp/nsd1912/demo/abc.txt')
('/tmp/nsd1912/demo', 'abc.txt')
>>> os.path.join('/tmp/nsd1912/demo', 'abc.txt')
'/tmp/nsd1912/demo/abc.txt'
>>> os.path.splitext('hello.txt')  # 切分扩展名
('hello', '.txt')

>>> os.path.isabs('/aaa/bb/ccc/ddd')  # 是绝对路径吗？
True
>>> os.path.isdir('/etc')  # 存在并且是目录吗？
True
>>> os.path.isfile('/etc')  # 存在并且是文件吗？
False
>>> os.path.islink('/etc/grub2.cfg')  # 存在并且是链接吗？

True
```

```
>>> os.path.ismount('/home')  # 存在并且是挂载点吗？
True
>>> os.path.exists('/tmp')   # 存在吗？
True
```

os.walk方法

```
[root@localhost day01]# mkdir -p /tmp/demo/{aaa,bbb}
[root@localhost day01]# touch /tmp/demo/{1..3}.txt
[root@localhost day01]# touch /tmp/demo/aaa/a{1,2}.txt
[root@localhost day01]# touch /tmp/demo/bbb/b{1..3}.txt
[root@localhost day01]# ls -R /tmp/demo/
/tmp/demo/:
1.txt  2.txt  3.txt  aaa  bbb

/tmp/demo/aaa:
a1.txt  a2.txt

/tmp/demo/bbb:
b1.txt  b2.txt  b3.txt

>>> list(os.walk('/tmp/demo'))
>>> result = list(os.walk('/tmp/demo'))
>>> pprint.pprint(result)
[('/tmp/demo', ['aaa', 'bbb'], ['1.txt', '2.txt', '3.txt']),
 ('/tmp/demo/aaa', [], ['a1.txt', 'a2.txt']),
 ('/tmp/demo/bbb', [], ['b1.txt', 'b2.txt', 'b3.txt'])]
>>> len(result)
3
>>> result[0]
('/tmp/demo', ['aaa', 'bbb'], ['1.txt', '2.txt', '3.txt'])
>>> result[1]
('/tmp/demo/aaa', [], ['a1.txt', 'a2.txt'])
>>> result[2]
('/tmp/demo/bbb', [], ['b1.txt', 'b2.txt', 'b3.txt'])
# 分析，result列表是由元组构成的：
# (路径字符串，目录列表，文件列表 )

>>> for line in os.walk('/tmp/demo'):
...   print(line)
...
('/tmp/demo', ['aaa', 'bbb'], ['1.txt', '2.txt', '3.txt'])
('/tmp/demo/aaa', [], ['a1.txt', 'a2.txt'])
('/tmp/demo/bbb', [], ['b1.txt', 'b2.txt', 'b3.txt'])

>>> for line in os.walk('/tmp/demo'):
...   print('%s:' % line[0])
...   for d in line[1]:
...     print('\033[34;1m%s\033[0m' % d, end='\t')
...   for f in line[2]:
...     print(f, end='\t')
...   print('\n')
...
/tmp/demo:
aaa     bbb     1.txt   2.txt   3.txt

/tmp/demo/aaa:
a1.txt  a2.txt

/tmp/demo/bbb:
b1.txt  b2.txt  b3.txt
```

```
# 获取每个文件的绝对路径
>>> for line in os.walk('/tmp/demo'):
...   for f in line[2]:
...     os.path.join(line[0], f)
...
'/tmp/demo/1.txt'
'/tmp/demo/2.txt'
'/tmp/demo/3.txt'
'/tmp/demo/aaa/a1.txt'
'/tmp/demo/aaa/a2.txt'
'/tmp/demo/bbb/b1.txt'
'/tmp/demo/bbb/b2.txt'
'/tmp/demo/bbb/b3.txt'

# 多元赋值，使用3个变量
>>> for path, folders, files in os.walk('/tmp/demo'):
...   for file in files:
...     os.path.join(path, file)
...
'/tmp/demo/1.txt'
'/tmp/demo/2.txt'
'/tmp/demo/3.txt'
'/tmp/demo/aaa/a1.txt'
'/tmp/demo/aaa/a2.txt'
'/tmp/demo/bbb/b1.txt'
'/tmp/demo/bbb/b2.txt'
'/tmp/demo/bbb/b3.txt'
```

pickle模块

- pickle模块可以将各种各样的数据类型存储到文件中，还可以无损地取出来。
- 以前学习的文件，以w方式打开，只能写入str类型的数据；以wb方式打开，只能写入bytes类型的数据。

```
>>> import pickle
>>> d1 = {'name': 'bob', 'age': 20}
>>> with open('/tmp/a.data', 'wb') as fobj:
...   pickle.dump(d1, fobj)

>>> import pickle
>>> with open('/tmp/a.data', 'rb') as fobj:
...   d2 = pickle.load(fobj)
...
>>> d2
{'name': 'bob', 'age': 20}
>>> type(d2)
<class 'dict'>
```

作业：记账程序

- 数据保存的形式: [日期，收入，支出，余额，备注]

```
>>> from time import strftime
>>> records = []   # 所有的记账记录
>>> init_data = [strftime('%Y-%m-%d'), 0, 0, 10000, 'init data']
>>> records.append(init_data)
>>> records
[['2020-05-23', 0, 0, 10000, 'init data']]
>>> record = [strftime('%Y-%m-%d'), 0, 300, 9700, 'buy shoes']
>>> records.append(record)
>>> records
[['2020-05-23', 0, 0, 10000, 'init data'], ['2020-05-23', 0, 300, 9700, 'buy shoes']]
# 取出最新余额
>>> records[-1]
['2020-05-23', 0, 300, 9700, 'buy shoes']
>>> records[-1][-2]
9700
```