py2_day01

数据分类

存储

- 标量：数字、字符串
- 容器：列表、元组、字典

更新

- 可变：列表、字典
- 不可变：数字、字符串、元组

访问

- 直接：数字
- 顺序：字符串、列表、元组
- 映射：字典

时间

时间表示方式

时间戳

自1970-1-1 00:00:00到某一时间点之间的秒数

```
>>> import time
>>> time.time()
1557710785.3078065
```

UTC：世界协调时

字符串时间。以英国格林威治这个城市所在的经度为起始点，每隔15度角成为一个时区。

```
>>> time.ctime()
'Mon May 13 09:33:16 2019'
```

struct_time：九元组

```
>>> time.localtime()
time.struct_time(tm_year=2019, tm_mon=5, tm_mday=13,
tm_hour=9, tm_min=34, tm_sec=11, tm_wday=0, tm_yday=133,
tm_isdst=0)
>>> t1 = time.localtime()
>>> t1.tm_year
2019
>>> t1.tm_hour
9
```

time模块

```
>>> import time
>>> time.time()  # 自1970-1-1 00:00:00到time.time()之间的秒
数
1557711710.3989246
>>> time.sleep(3)  # 睡眠
>>> time.strftime('%Y-%m-%d %H:%M:%S')
'2019-05-13 09:51:36'
>>> time.strftime('%a %A')  # 周几
'Mon Monday'

# 给定字符串和时间样式，将字符串转换成struct_time
>>> t1 = time.strptime('2019-05-13 09:51:36', '%Y-%m-%d
%H:%M:%S')
>>> t1
time.struct_time(tm_year=2019, tm_mon=5, tm_mday=13,
tm_hour=9, tm_min=51, tm_sec=36, tm_wday=0, tm_yday=133,
tm_isdst=-1)
>>> t2 = time.localtime()
>>> t2
time.struct_time(tm_year=2019, tm_mon=5, tm_mday=13,
tm_hour=9, tm_min=56, tm_sec=58, tm_wday=0, tm_yday=133,
tm_isdst=0)
>>> t2 > t1
True
>>> t2 < t1
False
```

datetime模块

```
>>> from datetime import datetime
#返回的对象，各部分是年月日时分秒毫秒
>>> datetime.now()
datetime.datetime(2019, 5, 13, 10, 43, 57, 336172)

>>> t = datetime.now()
>>> t.year
2019
>>> t.month
5
>>> t.day
13
>>> t.hour
10
>>> t.minute
44
>>> t.second
39
>>> t.microsecond
319444

>>> t.strftime('%Y%m%d %H:%M:%S')
'20190513 10:44:39'
>>> datetime.strptime('2019-05-13 12:00:00', '%Y-%m-%d
%H:%M:%S')
datetime.datetime(2019, 5, 13, 12, 0)

>>> t1 = datetime(2019, 5, 13)
>>> t1
datetime.datetime(2019, 5, 13, 0, 0)
```

```
>>> from datetime import datetime, timedelta
>>> t1 = datetime.now()
>>> d1 = timedelta(days=100, hours=1)
>>> t1 - d1  # 100天零1小时之前的时间
datetime.datetime(2019, 2, 2, 10, 23, 29, 737582)
>>> t1 + d1  # 100天零1小时之后的时间
datetime.datetime(2019, 8, 21, 12, 23, 29, 737582)
```

异常处理

当程序不能正常工作时，程序出现错误，它将崩溃终止执行，这时程序默认向终端抛出异常。

把有可能发生异常的语句，放到try中执行。通过except捕获异常，异常不发生才需要执行的语句，放到else中。异常不管是否发生，都要执行的语句，放到finally中

```
try:
    nums = int(input('number: '))
    result = 100 / nums
except (ValueError, ZeroDivisionError):
    print('无效输入')
except (KeyboardInterrupt, EOFError):
    print('\nBye-bye')
else:
    print(result)
finally:
    print('Done')
```

在编写程序时，并不总是需要写全部的语法，用的最多的组合是***try-except***和***try-finally***

os模块

```
>>> import os
>>> os.getcwd()   # pwd
>>> os.listdir()  # ls
>>> os.listdir('/home')  # ls /home
>>> os.mkdir('/tmp/demo')  # mkdir /tmp/demo
>>> os.makedirs('/tmp/aaa/bbb/ccc') # mkdir -p
/tmp/aaa/bbb/ccc
>>> os.chdir('/tmp/demo')  # cd /tmp/demo
>>> os.getcwd()
'/tmp/demo'
>>> os.listdir()
[]
>>> os.symlink('/etc/hosts', 'zhuji')  # ln -s
>>> import shutil
>>> shutil.copy('/etc/passwd', 'passwd')
'passwd'
>>> os.listdir()
['zhuji', 'passwd']
>>> os.stat('passwd')
os.stat_result(st_mode=33188, st_ino=408593036,
st_dev=64768, st_nlink=1, st_uid=0, st_gid=0,
st_size=2727, st_atime=1557732810, st_mtime=1557732810,
st_ctime=1557732810)
>>> mima = os.stat('passwd')
>>> mima.st_size
2727
>>> time.ctime(mima.st_atime)
'Mon May 13 15:33:30 2019'
>>> os.chmod('passwd', 0o755)   # chmod 755 passwd
>>> os.chmod('passwd', 420)  # chmod 644 passwd
```

```
>>> os.chown('passwd', 1014, 1015)  # chown
>>> os.listdir()
['zhuji', 'passwd']
>>> os.remove('zhuji')  # rm -f
>>> os.listdir()
['passwd']
```

```
>>> os.getcwd()
'/tmp/demo'
>>> os.listdir()
['passwd']
>>> os.path.abspath('passwd')
'/tmp/demo/passwd'

>>> fname = os.path.abspath('passwd')
>>> fname
'/tmp/demo/passwd'
>>> os.path.basename(fname)
'passwd'
>>> os.path.dirname(fname)
'/tmp/demo'
>>> os.path.split(fname)
('/tmp/demo', 'passwd')
>>> os.path.join('/tmp/demo', 'passwd')
'/tmp/demo/passwd'

>>> os.path.isdir('/etc/abc')  # [ -d /etc/abc ]
False
>>> os.path.isfile('/etc/hosts') # [ -f /etc/hosts ]
True
>>> os.path.islink('/etc/grub2.cfg')  # 是链接吗？
True
>>> os.path.ismount('/boot')  # 是挂载点吗？
True
>>> os.path.exists('/etc')   # 存在吗？
True
```

pickle模块

常规的文件，只能写入字符串，不能写其他数据类型

```
>>> f = open('/tmp/data', 'w')
>>> f.write('ni hao\n')
7
>>> f.write({'name': 'bob', 'age': 20})
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: write() argument must be str, not dict
>>> f.close()
```

pickle模块可以把任意的数据类型写入到文件，还可以无损地取出来。

```
>>> import pickle
>>> shop_list = {'eggs': 2, 'apple': 5, 'banana': 5}
>>> with open('/tmp/shop.data', 'wb') as fobj:
...     pickle.dump(shop_list, fobj)

>>> with open('/tmp/shop.data', 'rb') as fobj:
...     mydict = pickle.load(fobj)
>>> type(mydict)
<class 'dict'>
>>> mydict
{'eggs': 2, 'apple': 5, 'banana': 5}
>>> mydict['apple']
5
```

记账程序

1. 记账的样式

日期	收入	支出	余额	备注
2019-05-13	0	0	10000	init
2019-05-14	10000	0	20000	salary
2019-05-14	0	200	19800	eat

2. 记录的表示方式

```
records = [
    ['2019-05-13', 0, 0, 10000, 'init'],
    ['2019-05-14', 10000, 0, 20000, 'salary'],
    ['2019-05-14', 0, 200, 19800, 'eat'],
]
```

3. 列表的形成

   只有收入、支出和备注是手工录入的，其他字段需要程序自动完成。

4. 取出最新余额

```
records[-1][-2]
```