tedu_py0201

语法风格

变量赋值

```
[root@room8pc16 py0201]# python3
# python支持链式多重赋值
>>> a = b = 10

# 多元赋值
>>> x, y = 10, 20
>>> c, d = (100, 200)
>>> c
100
>>> d
200
>>> e, f = [1, 2]
>>> e
1
>>> f
2
```

标识符：

就是各种名字，如变量名、函数名、模块名、类名。它们有统一的命名约定。

关键字：

为了实现python的语法结构，python把一些单词作为保留字，不允许用户使用。

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def',
'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in',
'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with',
'yield']
```

内建：

不是关键字，用户可以把名称覆盖。

```
>>> type(len)
<class 'builtin_function_or_method'>
```

内建的官方说明：https://docs.python.org/zh-cn/3/library/functions.html

模块文件布局

```
#!/usr/local/bin/python3
"""模块的文档字符串，用于帮助信息"""

# 模块导入
import os
import time

hi = 'Hello World'   # 全局变量的定义
debug = True

# 类定义
class MyClass:
    pass

class MyClass2:
    pass

# 函数的定义
def func1():
    pass

def func2():
    pass

if __name__ == '__main__':
    func1()
```

编程思路

1. 发呆。思考程序的运作方式（交互式？非交互？）

```
filename: /etc
file already exists, try again.
filename: /etc/hosts
file already exists, try again.
filename: /tmp/aaa.txt
请输入内容，输入end结束
(end to quit)> hello world!
(end to quit)> 2nd line.
(end to quit)> ni hao.
(end to quit)> end
```

2. 思考程序有哪些功能，将这些功能写成功能函数。
3. 编写主程序代码，按顺序调用各个函数

```
def get_fname():
    '用于获取文件名'

def get_content():
    '用于获取内容'

def wfile(fname, content):
    '用于将内容写入文件'

if __name__ == '__main__':
    fname = get_fname()
    content = get_content()
    wfile(fname, content)
```

4. 编写各个功能模块

序列对象

序列对象内建函数

```
>>> str(100)
'100'
>>> list('abc')
['a', 'b', 'c']
>>> tuple('abc')
('a', 'b', 'c')
>>> max([10, 20, 5, 20, 30])
30
>>> min([10, 20, 5, 20, 30])
5
>>> alist = ['tom', 'jerry']
>>> enumerate(alist)
<enumerate object at 0x7f0f71a1aaf8>
>>> list(enumerate(alist))   # 把enumerate的结构转成列表
[(0, 'tom'), (1, 'jerry')]
>>> for ind, name in enumerate(alist):
...   print(ind, name)
...
0 tom
1 jerry

>>> from random import randint
>>> nums = [randint(1, 100) for i in range(10)]
>>> nums
[68, 50, 31, 44, 96, 31, 43, 98, 12, 7]
>>> list(reversed(nums))   # 翻转列表
[7, 12, 98, 43, 31, 96, 44, 31, 50, 68]
>>> sorted(nums)   # 排序
[7, 12, 31, 31, 43, 44, 50, 68, 96, 98]
```

字符串格式化

```
>>> '%s is %s years old' % ('bob', 20)
'bob is 20 years old'
>>> '%s is %d years old' % ('bob', 20)
'bob is 20 years old'
>>> '%s is %d years old' % ('bob', 20.5)
'bob is 20 years old'
>>> '%d is %d years old' % ('bob', 20)  # 报错，bob转不成整数
>>> '%#o' % 10   # 转成8进制
'0o12'
>>> '%#x' % 10   # 转成16进制
'0xa'
>>> '%f' % (5 / 3)   # 浮点数
'1.666667'
>>> '%.2f' % (5 / 3)   # 小数位2位
'1.67'
>>> '%6.2f' % (5 / 3)   # 小数位2位，总宽度为6，不足部分用空格补
'  1.67'
>>> '%10s%8s' % ('alice', 20)   # alice占10个宽度，20占8个宽度
'     alice      20'
>>> '%10s%8s' % ('bob', 20)
'       bob      20'
>>> '%-10s%-8s' % ('alice', 20)   # 左对齐
'alice     20      '
>>> '%-10s%-8s' % ('bob', 20)
'bob       20      '
>>> '{} is {} years old.'.format('bob', 20)
'bob is 20 years old.'
```

python调用系统命令的方法

```
>>> import subprocess
>>> subprocess.run('ls', shell=True)
>>> result = subprocess.run('ls', shell=True)
check_id.py  mktxtfile.py  tedu_py0201.md
>>> result.returncode    # 相当于$?
0
>>> result = subprocess.run('id root; id john', stdout=subprocess.PIPE,
stderr=subprocess.PIPE, shell=True)
>>> result.stdout
b'uid=0(root) gid=0(root) \xe7\xbb\x84=0(root)\n'
>>> result.stderr
b'id: john: no such user\n'
```

原始字符串、真实字符串

```
>>> win_path = 'c:\temp\new'
>>> print(win_path)
c:  emp
ew
>>> wpath = r'c:\temp\new'
>>> print(wpath)
c:\temp\new
>>> wpath
'c:\\temp\\new'
```

字符串方法

```
>>> '\thello world   '.strip()   # 去除两端空白字符
'hello world'
>>> '\thello world   '.lstrip()   # 去除左端空白字符
'hello world   '
>>> '\thello world   '.rstrip()   # 去除右端空白字符
'\thello world'
>>> hi = 'hao123'
>>> hi.upper()
'HAO123'
>>> 'HAO123'.lower()
'hao123'
>>> 'hello greet welcome'.split()
['hello', 'greet', 'welcome']
>>> hi.center(30)
'            hao123            '
>>> hi.center(30, '#')
'############hao123############'
>>> hi.ljust(30, '*')
'hao123************************'
>>> hi.rjust(30, '*')
'************************hao123'
>>> hi.startswith('ha')   # 以ha开头吗？
True
>>> hi.endswith('abc')   # 以abc结尾吗？
False
>>> hi.islower()  # 字符串中的字母都是小写吗？
True
>>> hi.isdigit()   # 所有的字符都是数字吗？
False
>>> '123'.isdigit()
True
```

列表和元组

```
>>> alist = [10, 30, 25, 15, 40, 22]
>>> alist.append(35)   # 追加
>>> alist.extend([20, 30, 10, 20])
>>> alist
[10, 30, 25, 15, 40, 22, 35, 20, 30, 10, 20]
```

```
>>> alist.remove(10)   # 删除第一个10
>>> alist.index(22)   # 取出22的下标
4
>>> alist.reverse()   # 翻转列表
>>> alist
[20, 10, 30, 20, 35, 22, 40, 15, 25, 30]
>>> alist.insert(5, 20)  # 在下标为5的位置插入20
>>> alist.sort()   # 排序
>>> alist
[10, 15, 20, 20, 20, 22, 25, 30, 30, 35, 40]
>>> alist.count(20)   # 统计20有多少个
3
>>> alist.pop()   # 默认弹出列表中最后一项
40
>>> alist
[10, 15, 20, 20, 20, 22, 25, 30, 30, 35]
>>> alist.pop(6)   # 弹出下标为6的项目
25
>>> alist
[10, 15, 20, 20, 20, 22, 30, 30, 35]
```

元组相当于是静态的列表，列表可变，元组不可变。

```
>>> atuple = (10, 15, 30, 15)
>>> atuple.count(15)
2
>>> atuple.index(15)
1

# 单元素元组必须有逗号，否则构不成元组
>>> a = (10)
>>> a
10
>>> type(a)
<class 'int'>
>>> a = (10,)
>>> type(a)
<class 'tuple'>
>>> a
(10,)
>>> len(a)
1
```

列表练习：

模拟栈结构，要求有压栈、出栈、查询功能

1. 程序运行过程

```
(0) push
(1) pop
(2) view
(3) quit
```

```
Please input your choice(0/1/2/3): abc
Invalid choice. Try again.
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 2
[]
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 0
item to push: hao
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 2
['hao']
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 1
From stack popped: hao
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 1
Empty stack
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 3
Bye-bye
```

2. 把功能编写成功能函数
3. 主程序代码

```
def push_it():


def pop_if():


def view_it():


def show_menu():
```

```
if __name__ == '__main__':
    show_menu()
```

4. 编写每个函数功能

字典

字典是可变、容器、映射类型。它的值可以相同，但是key肯定不相同。字典的key必须是不可变对象。

```
>>> adict = {}.fromkeys(['bob', 'tom', 'jerry'], 23)
>>> adict
{'bob': 23, 'tom': 23, 'jerry': 23}
```

访问字典：通过key找value

```
>>> adict = {'name': 'tom', 'age': 23}
>>> adict
{'name': 'tom', 'age': 23}
>>> 'tom' in adict    # tom是字典的key吗？
False
>>> 'name' in adict
True
>>> for key in adict:
...   print(key, adict[key])
...
name tom
age 23
>>> '%(name)s: %(age)s' % adict
'tom: 23'
```

更新字典

```
# 因为字典的key不能重复，所以直接对字典赋值时，如果key不在字典中是创建新项目，如果已经有key了，则更新它的值
>>> adict['email'] = 'tom@tedu.cn'
>>> adict
{'name': 'tom', 'age': 23, 'email': 'tom@tedu.cn'}
>>> adict['age'] = 25
>>> adict
{'name': 'tom', 'age': 25, 'email': 'tom@tedu.cn'}
```

字典的方法

```
>>> adict.keys()   # 所有的key
dict_keys(['name', 'age', 'email'])
>>> adict.values()   # 所有的值
dict_values(['tom', 25, 'tom@tedu.cn'])
>>> adict.items()
dict_items([('name', 'tom'), ('age', 25), ('email', 'tom@tedu.cn')])
```

```
>>> adict.pop('email')   # 根据key弹出一项
'tom@tedu.cn'
>>> adict
{'name': 'tom', 'age': 25}
>>> print(adict.get('qq'))   # 取出key为qq的值，不存在返回None
None
>>> adict.get('qq', 'not found')  # 指定key为qq的项不存在，返回not found
'not found'
>>> adict.get('age')
25
```

集合

集合用{}来表示。集合中的元素是不可变的、无序的、不能重复。所以集合就像是一个无值的字典。

```
>>> aset = set('abc')
>>> bset = set('bcd')
>>> aset
{'a', 'b', 'c'}
>>> bset
{'b', 'd', 'c'}
>>> aset | bset   # 并集
{'a', 'b', 'd', 'c'}
>>> aset & bset   # 交集
{'b', 'c'}
>>> aset - bset   # 差补，aset中有但是bset中没有
{'a'}

# 去重
>>> from random import randint
>>> nums = [randint(1, 10) for i in range(20)]
>>> nums
[1, 9, 3, 8, 6, 6, 3, 6, 3, 7, 4, 4, 1, 7, 9, 1, 4, 3, 3, 7]
>>> set(nums)
{1, 3, 4, 6, 7, 8, 9}

# 判断哪些行在b.txt中有，而a.txt中没有
[root@room8pc16 py0201]# cp /etc/passwd /tmp/pwd
[root@room8pc16 py0201]# cp /etc/passwd /tmp/pwd2
[root@room8pc16 py0201]# vim /tmp/pwd2   # 修改pwd2文件
# 找出pwd2中有，而pwd中没有的行
>>> with open('/tmp/pwd') as f1:
...   s1 = set(f1)   # 将文件中的每一行都存入集合
...
>>> with open('/tmp/pwd2') as f2:
...   s2 = set(f2)
>>> s2 - s1
{'ni hao\n', 'hello world\n'}
```

时间模块

时间表示方式：

- 时间戳：1970-1-1 0:0:00 到某一时间之间的秒数

```
>>> import time
>>> time.time()
1563091826.5869324
```

- UTC时间：世界协调时。

```
>>> time.ctime()
'Sun Jul 14 16:12:11 2019'
```

- strcut_time：由9项构成的元组

```
>>> time.localtime()
time.struct_time(tm_year=2019, tm_mon=7, tm_mday=14, tm_hour=16, tm_min=13, tm_sec=5,
tm_wday=6, tm_yday=195, tm_isdst=0)
>>> t = time.localtime()
>>> t.tm_year
2019
>>> t.tm_hour
16
```

time模块常用方法

```
>>> import time
>>> time.sleep(3)
>>> time.time()
1563092265.051062
>>> time.strftime('%Y-%m-%d %H:%M:%S')
'2019-07-14 16:18:31'
>>> time.strftime('%a')
'Sun'
# 将字符串格式的时间转成struct_time格式
>>> time.strptime('2019-07-14 16:18:31', '%Y-%m-%d %H:%M:%S')
time.struct_time(tm_year=2019, tm_mon=7, tm_mday=14, tm_hour=16, tm_min=18, tm_sec=31,
tm_wday=6, tm_yday=195, tm_isdst=-1)
```

datetime模块

```
>>> from datetime import datetime
>>> datetime.now()
datetime.datetime(2019, 7, 14, 16, 48, 24, 370629)
>>> t = datetime.now()
>>> t.year, t.month, t.day, t.hour, t.minute, t.second, t.microsecond
(2019, 7, 14, 16, 48, 43, 586114)
>>> t1 = datetime(2019, 7, 13)   # 2019年7月13日
>>> t1
datetime.datetime(2019, 7, 13, 0, 0)
```

```
>>> t > t1
True
>>> t.strftime('%Y-%m-%d %H:%M:%S')
'2019-07-14 16:48:43'

# 100天4小时之前的时间、之后的时间
>>> from datetime import datetime, timedelta
>>> t = datetime.now()
>>> days = timedelta(days=100, hours=4)
>>> t - days
datetime.datetime(2019, 4, 5, 12, 52, 39, 20586)
>>> t + days
datetime.datetime(2019, 10, 22, 20, 52, 39, 20586)
```

异常处理

```
try:
    有可能发生异常的语句
except 异常名字:
    处理异常的代码
else:
    不发生异常才执行的代码
finally:
    不管异常是否发生，都要执行的代码
```

os模块

```
>>> os.listdir()   # ls
>>> os.listdir('/tmp/')  # ls /tmp
>>> os.mkdir('/tmp/weekend')   # mkdir /tmp/weekend
>>> os.chdir('/tmp/weekend')   # cd /tmp/weekend
>>> os.getcwd()  # pwd
'/tmp/weekend'
>>> os.symlink('/etc/hosts', 'zhuji')   # ln -s /etc/hosts zhuji
>>> os.remove('zhuji')   # rm -f zhuji
>>> os.path.abspath('.')   # 获取绝对路径
>>> os.path.basename('/etc/weekend/hello.txt')
'hello.txt'
>>> os.path.dirname('/etc/weekend/hello.txt')
'/etc/weekend'
>>> os.path.split('/etc/weekend/hello.txt')
('/etc/weekend', 'hello.txt')
>>> os.path.join('/etc/weekend', 'hello.txt')
'/etc/weekend/hello.txt'
>>> os.path.isfile('/etc/hosts')   # /etc/hosts存在并且是文件吗？
>>> os.path.isdir('/etc/hosts')   # /etc/hosts存在并且是目录吗？
>>> os.path.exists('/etc/abc')   # 存在/etc/abc吗？
```

pickle模块

文件普通的读写行为只能操作字符串。如果希望把各种数据类型写到文件中，取出时还是这种类型，就需要用到pickle存储器了。pickle可以把任意的数据对象写入文件，并无损地取出来。

```
>>> import pickle
>>> shopping_list = ['apple', 'pear', 'banana']
>>> with open('/tmp/shop.data', 'wb') as fobj:  # 将列表存入文件
...   pickle.dump(shopping_list, fobj)

>>> with open('/tmp/shop.data', 'rb') as fobj:
...   mylist = pickle.load(fobj)  # 从文件中读出列表

>>> type(mylist)
<class 'list'>
>>> mylist
['apple', 'pear', 'banana']
```

记账练习

日期	收入	支出	余额	说明
2019-7-1	0	0	10000	init
2019-7-10	10000	0	20000	salary
2017-7-10	0	200	19800	eat

可以把整个文件对应成一个大列表，每行记录是一个小列表。把大列表通过pickle存到文件中。

```
[
    ['2019-7-1', 0, 0, 10000, 'init'],
    ['2019-7-10', 10000, 0, 20000, 'salary'],
    ['2017-7-10', 0, 200, 19800, 'eat'],
]
```