nsd1902_py01_day03

作业

列表解析生成IP地址

```
>>> ['192.168.1.1']
['192.168.1.1']
>>> ['192.168.1.1' for i in range(1, 11)]
['192.168.1.1', '192.168.1.1', '192.168.1.1', '192.168.1.1', '192.168.1.1', '192.168.1.1',
'192.168.1.1', '192.168.1.1', '192.168.1.1', '192.168.1.1']
>>> ['192.168.1.%s' % i for i in range(1, 11)]
['192.168.1.1', '192.168.1.2', '192.168.1.3', '192.168.1.4', '192.168.1.5', '192.168.1.6',
'192.168.1.7', '192.168.1.8', '192.168.1.9', '192.168.1.10']
```

九九乘法表：

```
for i in range(1, 10):
    for j in range(1, i + 1):
        print('%sx%s=%s' % (j, i, i * j), end=' ')
    print()
```

文件对象

处理文件的三个步骤：打开、读写、关闭。

文件读取

```
[root@room8pc16 day02]# cp /etc/passwd /tmp/mima
>>> f = open('/tmp/mima')   # 默认以r方式打开
>>> data = f.read()   # 默认read读取全部数据
>>> print(data)

>>> data = f.read()   # 因为文件指针已经到文件结尾，再读数据将是空
>>> print(data)

>>> data
''
>>> f.close()

# 重新打开文件
>>> f = open('/tmp/mima')
>>> f.read(4)   # 读取4字节
'root'
>>> f.read(4)
':x:0'
>>> f.readline()   # 读取到遇到的第一个\n
>>> f.readlines()   # 把文件的第一行作为列表项，保存到列表中
```

```
>>> f.close()
```

最常用的、读取文本文件的方式是for循环遍历：

```
>>> f = open('/tmp/mima')
>>> for line in f:
...     print(line, end='')
>>> f.close()
```

写入文件

```
>>> f = open('/tmp/mima', 'w')   # 文件不存在则创建，存在会清空
>>> f.write('hello world!\n')
13    # 返回写入到文件的字节数
>>> f.writelines(['2nd line.\n', '3rd line.\n'])
[root@room8pc16 day02]# cat /tmp/mima   # 此时文件仍然是空的
>>> f.flush()   # 立即将数据从缓存同步到磁盘
[root@room8pc16 day02]# cat /tmp/mima
hello world!
2nd line.
3rd line.
>>> f.write('end\n')
4
>>> f.close()   # 关闭文件时，数据也会写入磁盘
```

处理非文本文件，与文本文件大体相似，只不过在打开文件时，需要加上b

```
[root@room8pc16 day02]# cp /usr/bin/ls /tmp/
>>> f = open('/tmp/ls')
>>> f.read(10)   # 报错。因为打开时默认以utf8编码打开，但是ls不是文本文件。读取时，Python试图把读出来的10个字节显示为utf8字符。
>>> f.close()

>>> f = open('/tmp/ls', 'rb')   # 以bytes类型打开
>>> f.read(10)
b'\x7fELF\x02\x01\x01\x00\x00\x00'
>>> f.close()

# 文本文件也能以bytes方式打开
>>> f = open('/tmp/mima', 'rb')
>>> f.read()
b'hello world!\n2nd line.\n3rd line.\nend\n'
>>> f.close()
```

文件打开后，如果不关闭，也可能没有问题，当程序结束时，文件自动关闭。但是关闭文件是很好的习惯。

如果使用with语句打开文件，当with语句结束时，文件自动关闭。

```
>>> with open('/tmp/mima') as f:
...     f.readline()
...     f.readline()
...
'hello world!\n'
'2nd line.\n'
>>> f.readline()     # 文件已经关闭，无法再读取数据
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: I/O operation on closed file.
```

移动文件指针：seek(x, y)。y可以取值0，1，2，分别表示文件开头、当前位置和结尾。x是偏移量。

```
>>> f = open('/tmp/mima')
>>> f.tell()   # 显示当前位置，永远从开头算偏移量
0
>>> f.seek(5, 0)
5
>>> f.read(1)
' '
>>> f.close()



>>> f = open('/tmp/mima')
>>> f.read()   # 读取全部内容
>>> f.read()   # 再读就没有数据了
''
>>> f.seek(0, 0)   # 回到文件开头
0
>>> f.read()   # 再次读取文件内容
>>> f.close()  # 关闭文件


>>> f = open('/tmp/mima', 'rb')
>>> f.seek(-5, 2)   # 指针移动到文件结尾前5个字节处
32
>>> f.read()
b'\nend\n'
```

位置参数

python将位置参数保存在sys.argv列表中

```
# vim args.py
import sys
print(sys.argv)

# python3 args.py
['args.py']

# python3 args.py hao 123
['args.py', 'hao', '123']
```

练习：修改拷贝文件的程序，要求使用函数和位置参数。

```
# python copy.py  /etc/hosts  /tmp/zhuji
```

默认参数：给定默认值的参数

```
>>> def pstar(n=30):
...     print('*' * n)
...
>>> pstar(40)  # 传参，使用n=40
****************************************
>>> pstar()   # 不传参，使用默认值n=30
******************************
```

模块

模块实际上就是一个以.py结尾的文件。去除.py之后前面的文件名就是模块名。模块名也是一个名称，变量名也是一个名称，这些名称统一称之为标识符。它们拥有一致的命名规范：

- 首字符必须是字母或下划线
- 其他字符可以是数字、字母、下划线
- 区分大小写

```
[root@room8pc16 day03]# vim star.py
'''
小星星

本程序有一个变量hi
还有一个函数pstar
'''
hi = 'Hello World'

def pstar(n=30):
    "用于打印一行星号"
    print('*' * n)
[root@room8pc16 day03]# ls star.py
star.py
[root@room8pc16 day03]# python3
>>> import star   # 将自己写的程序文件当作模块导入
>>> help(star)    # 查看帮助信息，观察帮助内容与文件内容
```

```
>>> star.hi
'Hello World'
>>> star.pstar()
******************************
>>> star.pstar(40)
****************************************


[root@room8pc16 day03]# vim call_star.py
import star

print(star.hi)
star.pstar()

[root@room8pc16 day03]# python3 call_star.py
Hello World
******************************
```

导入模块的方法

- import os : 常用
- from random import randint, choice：仅导入模块中的某些功能，常用

```
>>> from random import randint, choice
>>> randint(1, 10)
5
>>> choice('abd')
'a'
>>> random.randint(1, 10)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'random' is not defined
```

- import sys, time, getpass ： 不常用
- import getpass as gp：导入模块时，为其创建别名，不常用

```
>>> import getpass as gp
>>> gp.getpass()
Password:
```

第一模块导入（import）时，模块中的代码将会执行（load）一遍；再次导入，代码就不再执行了。

模块导入的特殊属性

每个模块都有一个内部变量叫__name__，__name__值有两种。当程序文件直接运行的时候，它的值是__main__，当它被别的程序导入时，值是模块名。

```
[root@room8pc16 day03]# cat foo.py
print(__name__)
[root@room8pc16 day03]# cat bar.py
import foo
[root@room8pc16 day03]# python3 foo.py
__main__
[root@room8pc16 day03]# python3 bar.py
foo
```