nsd1905_py01_day04

系统管理模块

shutil模块

```
# 拷贝文件对象
>>> import shutil
>>> f1 = open('/bin/touch', 'rb')
>>> f2 = open('/tmp/touch', 'wb')
>>> shutil.copyfileobj(f1, f2)
>>> f1.close()
>>> f2.close()

# 拷贝文件内容
>>> shutil.copyfile('/etc/motd', '/tmp/motd')

# 拷贝文件到目标，目标可以是目录，记住
>>> shutil.copy('/etc/issue', '/tmp')

# 拷贝文件到目标，目标可以是目录，同时拷贝元数据
>>> shutil.copy2('/etc/issue', '/tmp/issue.txt')

# 拷贝目录，记住
>>> shutil.copytree('/etc/security', '/var/tmp/anquan')

# 删除目录，记住
>>> shutil.rmtree('/var/tmp/anquan')

# 改变属主、属组，记住
>>> shutil.chown('/etc/motd', user='bob', group='bob')

# 移动，记住
>>> shutil.move('/tmp/motd', '/var/tmp/')
```

subprocess模块

- 用于执行系统命令

```
>>> import subprocess
# 在shell环境中执行ls ~
>>> subprocess.run('ls ~', shell=True)
>>> subprocess.run('id root', shell=True)

# 将输出和错误保存到result中
>>> result = subprocess.run('id root', shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
```

```
>>> result.returncode   # 相当于$?
0
>>> result.stdout  # 查看标准输出
b'uid=0(root) gid=0(root) \xe7\xbb\x84=0(root)\n'
>>> result.stdout.decode()  # 将bytes转为str
'uid=0(root) gid=0(root) 组=0(root)\n'

>>> result = subprocess.run('id zhaoliu', shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
>>> result.stderr
b'id: zhaoliu: no such user\n'
>>> result.returncode
1
```

python语法风格及模块布局

语法

```
# 链式多重赋值
>>> a = b = 10

# 多元赋值
>>> a, b = 10, 20
>>> c, d = (10, 20)
>>> e, f = [100, 200]
>>> g, h = 'mn'
```

合法标识符

- 各种各样的名字就是标识符
- 名称包括：变量、函数、模块、类
- 标识符有统一的命名约定
    - 首字符必须是字母或下划线
    - 其他字符必须是字母、数字、下划线
    - 区分大小写

关键字

- 构成python语法的那些保留字

```
>>> 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
- 可以被覆盖，但是不应该覆盖
- 内建函数：https://docs.python.org/zh-cn/3/library/index.html

```
>>> len('abcd')
4
>>> len = 10
>>> len('abcd')   # 报错
```

模块布局

```
#!/root/nsd1905/bin/python     # 解释器位置
"""文档字符串

在help查看时，能够看到的模块说明
"""

# 模块导入
import os
import string

# 全局变量定义
all_chs = string.ascii_letters + string.digits

# 类的定义
class MyClass:
    pass

# 函数定义
def func1():
    pass

# 主程序代码
if __name__ == '__main__':
    mc = MyClass()
    func1()
```

编程思路

1. 发呆。思考程序的运作方式。交互？非交互？如果是交互的，程序有什么输出，要求用户有什么输入。

```
# python mkfile.py
filename: /etc/hosts
文件已存在，请重试。
filename: /etc
文件已存在，请重试。
filename: /tmp/myfile.txt
输请入内容，在单独的一行上输入end结束。
(end to quit)> hello world!
(end to quit)> ni hao
(end to quit)> how are you?
```

```
(end to quit)> end
# cat /tmp/myfile.txt
hello world!
ni hao
how are you?
```

2. 思考程序有哪些功能，将功能编写成函数。这样，将大的、复杂的问题化简为一个个的小的问题。

```
def get_fname():
    '返回文件名'

def get_content():
    '返回内容'

def wfile(fname, content):
    '将内容content写入文件fname'
```

3. 书写主程序代码，按顺序调用函数

```
def get_fname():
    '返回文件名'

def get_content():
    '返回内容'

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

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

4. 填写每个具体的函数主体代码

序列对象操作

```
# list函数将对象转成列表
>>> list('abc')
['a', 'b', 'c']
>>> list(range(5))
[0, 1, 2, 3, 4]

# tuple函数将对象转成元组
>>> tuple('abc')
('a', 'b', 'c')
>>> tuple(range(5))
(0, 1, 2, 3, 4)
>>> tuple(['bob', 'alice'])
('bob', 'alice')
```

```
# str将对象转成字符串
>>> str(100)
'100'
>>> str([1, 2, 3])
'[1, 2, 3]'

# reversed函数用反转序列对象
>>> from random import randint
>>> nums = [randint(1, 100) for i in range(10)]
>>> nums
[1, 30, 79, 99, 39, 61, 96, 36, 48, 42]
>>> reversed(nums)
<list_reverseiterator object at 0x7f4b34134c50>
>>> for i in reversed(nums):
...   print(i)
>>> list(reversed(nums))
[42, 48, 36, 96, 61, 39, 99, 79, 30, 1]
>>> nums   # nums本身不会改变
[1, 30, 79, 99, 39, 61, 96, 36, 48, 42]

# sorted用于排序
>>> sorted(nums)  # 默认升序排列
[1, 30, 36, 39, 42, 48, 61, 79, 96, 99]
>>> sorted(nums, reverse=True)  # 降序排列
[99, 96, 79, 61, 48, 42, 39, 36, 30, 1]

# enumerate函数可以同时得到序列对象的下标和值
for ind, num in enumerate(nums):
    print(ind, num)
```

字符串

编码

- 将一串2进制的0/1组合表示某个字符
- 美国常用的编方案是ASCII
- 西方国家常用的编码方案是ISO – 8859 – 1，即Latin-1
- 中国常用的编码方案：GB2312 / GBK / GB18030
- ISO推出了万国码：Unicode，其中应用最多的形式是utf8
- utf8是变长编码解决方案。英文字符占1个字节，汉字占3个字节

字符串比较大小，是看字符的码值大小

```
>>> ord('a')   # 查看字母a的码值
97
>>> ord('A')
65
>>> 'a' > 'A'
True
>>> ord('中')
20013
>>> ord('上')
19978
>>> '中' > '上'
True
```

字符串格式化操作符

```
>>> '%s is %s years old' % ('bob', 20)
'bob is 20 years old'
>>> '%s is %d years old' % ('bob', 20)
'bob is 20 years old'
>>> '5 / 3 = %d' % (5 / 3)  # %d: 10进制整数
'5 / 3 = 1'
>>> '5 / 3 = %f' % (5 / 3)  # %f: 浮点数
'5 / 3 = 1.666667'
>>> '5 / 3 = %5.2f' % (5 / 3)  # %5.2f：共5列，小数点2位
'5 / 3 =  1.67'

>>> '%8s%5s' % ('name', 'age')  # %8s: 占8列，右对齐
'    name  age'
>>> '%8s%5s' % ('bob', 20)
'     bob   20'
>>> '%-8s%-5s' % ('name', 'age')  # 负数表示左对齐
'name    age  '
>>> '%-8s%-5s' % ('bob', 20)
'bob     20   '

# 不常用的方法，了解
>>> '%#o' % 10  # 以8进制表示
'0o12'
>>> '%#x' % 10  # 以16进制表示
'0xa'
>>> '%e' % 10000   # 科学计数法
'1.000000e+04'
>>> '%05d' % 10   # 宽度不够5，用0填充，而不是空格填充
'00010'
```

format方法

与%s等格式化方法一样，实现字符串格式化

```
>>> '{} is {} years old'.format('bob', 20)
'bob is 20 years old'
>>> '{} is {} years old'.format(20, 'bob')
'20 is bob years old'
>>> '{1} is {0} years old'.format(20, 'bob')
'bob is 20 years old'
```

原始字符串

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

字符串方法

```
# 没有使用字符串方法，判断一个字符串是否完全由数字构成
>>> s1 = '1234'
>>> s2 = '12a34'

>>> def is_digit(s):
...   for ch in s:
...     if ch not in '0123456789':
...       return False
...   return True #注意，函数就算有多个return也只会执行一个，类似于循环的break
...
>>> is_digit(s1)
True
>>> is_digit(s2)
False

# 使用字符串方法判断字符串是否完全由数字构成
>>> s1.isdigit()
True
>>> s2.isdigit()
False

# 字符串方法
>>> s1.center(30)  # 居中
'             1234             '
>>> s1.center(30, '*')   # 居中，两边用*号填充
'*************1234*************'
>>> s1.ljust(30, '#')  # 左对齐
'1234##########################'
```

```
>>> s1.rjust(30, '@')  # 右对齐
'@@@@@@@@@@@@@@@@@@@@@@@@@@1234'
>>> '  \thello world.\n'.strip()  # 移除两端空白字符
'hello world.'
>>> '  \thello world.\n'.lstrip()  # 移除左端空白字符
'hello world.\n'
>>> '  \thello world.\n'.rstrip()  # 移除右端空白字符
'  \thello world.'
>>> s3 = 'abc'
>>> s3.upper()  # 转成大写
'ABC'
>>> 'HELLO WORLD'.lower()  # 转成小写
'hello world'

>>> s1
'1234'
>>> s1.islower()  # 有字母，并且是小写为True
False
>>> s2
'12a34'
>>> s2.islower()  # 有字母，并且是小写为True
True
>>> 'HAO123'.isupper()  # 字母是大写的
True
>>> s1.isdigit()  # 所有字符全部为数字
True
>>> s2.isdigit()
False
>>> 'Hao'.isalpha()  # 字符全为字母
True
>>> 'Hao123'.isalpha()
False
>>> 'Hao123'.isalnum()  # 字符全部为字母或数字
True
>>> 'Hao123'.startswith('Hao')  # 字符串以Hao开头吗？
True
>>> 'Hao123'.endswith('ab')  # 字符串以ab结尾吗？
False
>>> 'hao 123 abc'.split()   # 切分字符串
['hao', '123', 'abc']
>>> 'hao-123-abc'.split()
['hao-123-abc']
>>> 'hao-123-abc'.split('-')  # 以-为分隔符，切分字符串
['hao', '123', 'abc']
>>> s1.replace('12', 'abcd')  # 替换
'abcd34'
```