nsd1912-py01-day04

变量命名解决方案：https://unbug.github.io/codelf/

shutil模块

- 用于实现一部分系统操作功能，如复制、移动

```
>>> import shutil
>>> f1 = open('/bin/touch', 'rb')
>>> f2 = open('/tmp/tch', 'wb')
>>> shutil.copyfileobj(f1, f2)
>>> f1.close()
>>> f2.close()

# cp /etc/passwd /tmp/mima
>>> shutil.copy('/etc/passwd', '/tmp/mima')
'/tmp/mima'
# cp -r
>>> shutil.copytree('/etc/security', '/tmp/security')
'/tmp/security'
# mv
>>> shutil.move('/tmp/security', '/var/tmp')
'/var/tmp/security'
>>> shutil.rmtree('/var/tmp/security')  # rm -rf
>>> shutil.chown('/tmp/tch', user='student', group='student')   # chown
```

subprocess模块

- subprocess用于执行系统命令

```
>>> import subprocess
>>> result = subprocess.run('id root', shell=True, stdout=subprocess.PIPE)
>>> result
CompletedProcess(args='id root', returncode=0, stdout=b'uid=0(root) gid=0(root)
\xe7\xbb\x84=0(root)\n')
>>> result.args
'id root'
>>> result.returncode   # $?
0
>>> result.stdout
b'uid=0(root) gid=0(root) \xe7\xbb\x84=0(root)\n'
>>> result.stdout.decode()  # bytes转str类型
'uid=0(root) gid=0(root) 组=0(root)\n'

>>> result = subprocess.run('id root; id zhangsan', shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
>>> result
CompletedProcess(args='id root; id zhangsan', returncode=1, stdout=b'uid=0(root)
gid=0(root) \xe7\xbb\x84=0(root)\n', stderr=b'id: zhangsan: no such user\n')
>>> result.stderr
b'id: zhangsan: no such user\n'
```

语法风格

```
# 链式多重赋值
>>> a = b = 10
>>> id(a)
9360768
>>> id(b)
9360768
>>> b = 100
>>> id(b)
9363648
>>> a
10

# 可变对象，注意，变量指向同一地址空间，改一个，另一个也跟着改
>>> l1 = l2 = [1, 2]
>>> id(l1)
140176998238728
>>> id(l2)
140176998238728
>>> l1[1] = 100
>>> l1
[1, 100]
>>> l2
[1, 100]


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

# 交换两个变量的值
# 其他语言的写法
>>> t = a
>>> a = b
>>> b = t
# python的写法
>>> c, d = d, c
```

- 标识符：就是各种各样的名字，如变量、函数、模块。它们的命名遵守相同的命名约定
- 关键字：不能被覆盖的保留字

```
>>> import keyword
>>> keyword.kwlist
```

- 内建：系统提前定义好的、但是可以被覆盖的函数或变量

```
>>> len('abc')
3
>>> len = 100
>>> len('abc')   # 报错，len现在是数字100，数字不能调用
```

https://docs.python.org/zh-cn/3/library/functions.html

模块布局

```
#!/usr/local/bin/python3   # 指定解释器
"""模块文件的文档字符串

可以包含很多行，在查看 help帮助时显示
"""
import string              # 模块导入
import os

all_chs = string.asicc_letters + digits  # 全局变量
debug = True

class MyClass:             # 声明类
    pass

def func1():               # 声明函数
    pass

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

编程思路

1. 发呆。考虑程序的运行方式。交互？非交互？

```
[root@localhost day04]# python3 mk_file.py
文件名： /
文件已存在，请重试。
文件名： /etc/hosts
文件已存在，请重试。
文件名： /tmp/myfile.txt
请输入内容，单独一行输入 end结束。
(end to quit)> How are you?
(end to quit)> 睡醒了吗？
(end to quit)> the end
(end to quit)> end
[root@localhost day04]# ls /tmp/myfile.txt
[root@localhost day04]# cat /tmp/myfile.txt
How are you?
睡醒了吗？
the end
```

2. 思考程序有哪些功能。将这些功能写成功能函数。

```
def get_fname():
    "用于获取一个不存在的文件名 "

def get_content():
    "用于获取内容"

def wfile(fname, content):
    "用于将内容content写入到文件fname中"
```

3. 编写主程序代码，按照一定的逻辑，调用函数

```
def get_fname():
    "用于获取一个不存在的文件名 "

def get_content():
    "用于获取内容"

def wfile(fname, content):
    "用于将内容content写入到文件fname中"

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

4. 完成每一个功能函数

```
import os

def get_fname():
    "用于获取一个不存在的文件名 "
    while 1:
        fname = input('文件名： ')
        if not os.path.exists(fname):  # 如果文件不存在
            break
        print('文件已存在，请重试。 ')

    return fname

def get_content():
    "用于获取内容 "
    content = []  # 定义用于保存内容的列表

    print('请输入内容，单独一行输入 end结束。 ')
    while 1:
        line = input('(end to quit)> ')
        if line == 'end':
            break
        content.append(line)

    return content

def wfile(fname, content):
    "用于将内容content写入到文件fname中"
    with open(fname, 'w') as fobj:
        fobj.writelines(content)

if __name__ == '__main__':
    fname = get_fname()
    content = get_content()
    content = ['%s\n' % line for line in content]
    wfile(fname, content)
```

> 格式化代码快捷键：ctrl + alt + L

序列对象

```
# list将对象转成列表
>>> list(10)   # 报错
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list('hello world')
['h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd']

# str将对象转成字符串
>>> str(100)
'100'

# tuple将对象转成元组
>>> tuple(range(1, 11))
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

>>> from random import randint
>>> nums = [randint(1, 100) for i in range(10)]
>>> nums
[61, 4, 29, 42, 21, 14, 42, 63, 73, 82]
>>> sorted(nums)  # 升序排列，不会改变列表本身
[4, 14, 21, 29, 42, 42, 61, 63, 73, 82]
>>> nums
[61, 4, 29, 42, 21, 14, 42, 63, 73, 82]
>>> reversed(nums)  # 翻转，返回翻转对象
<list_reverseiterator object at 0x7f4cee77fa90>
>>> list(reversed(nums))
[82, 73, 63, 42, 14, 21, 42, 29, 4, 61]
>>> for i in reversed(nums):
...   print(i)

# enumerate可以同时得到下标和值
>>> enumerate(nums)
<enumerate object at 0x7f4ce4a81828>
>>> list(enumerate(nums))
[(0, 61), (1, 4), (2, 29), (3, 42), (4, 21), (5, 14), (6, 42), (7, 63), (8, 73), (9,
82)]
>>> for data in enumerate(nums):
...   print(data)
...
>>> for data in enumerate(nums):
...   print(data[0], data[1])
...
>>> for i, n in enumerate(nums):
...   print(i, n)
...
```

字符串

- 属于标量、顺序、不可变类型
- 字符串比较是按编码值大小比较
- 字符串格式化

```
# 基本形式：
'' % ()

>>> '%s is %s years old' % ('牛老师', 20)
'牛老师 is 20 years old'
# %d表示10进制整数
>>> '%s is %d years old' % ('牛老师', 20)
'牛老师 is 20 years old'
>>> '%d is %d years old' % ('牛老师', 20)  # 报错
>>> '%s is %d years old' % ('牛老师', 20.6)
'牛老师 is 20 years old'
>>> '%f' % (5 / 3)    # %f是浮点数
'1.666667'
>>> '%.2f' % (5 / 3)  # 小数位2位
'1.67'
>>> '%6.2f' % (5 / 3) # 总宽度为6，不足的用空格补全
'  1.67'

>>> '%10s%8s' % ('name', 'age')  # 指定字段宽度
'      name     age'
>>> '%10s%8s' % ('tom', 20)
'       tom      20'
>>> '%-10s%-8s' % ('name', 'age')  # 负数实现左对齐
'name      age     '
>>> '%-10s%-8s' % ('tom', 20)
'tom       20      '


# 不常用的格式化操作
>>> '%#x' % 10   # 以16进制表示
'0xa'
>>> '%#o' % 10   # 以8进制表示
'0o12'
>>> '%e' % 1230000  # 科学计数法
'1.230000e+06'
```

- 字符串格式化还可以用format方法

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

- 原始字符串、真实字符串。指的是字符串中的每个字符都表示它本身的含义。

```
>>> win_path = 'c:\temp'
>>> print(win_path)  # \t被解释为tab
c:      emp
>>> wpath = r'c:\temp'  # 真实字符串
>>> print(wpath)
c:\temp
>>> wpath
'c:\\temp'
>>> win_path = 'c:\\temp'
>>> print(win_path)
c:\temp
```

字符串方法

- 如果字符串没有方法，所有功能都需要自己实现
- 掌握了字符串方法，可以节省自己的实间和精力

```
>>> s1 = '  \thello world.\n'
>>> s1.strip()   # 去除两端空白字符
'hello world.'
>>> s2 = s1.strip()
>>> s2
'hello world.'
>>> s1.lstrip()   # 去除左端空白字符
'hello world.\n'
>>> s2.rstrip()   # 去除右端空白字符
'hello world.'
>>> s2.upper()    # 字母转大写
'HELLO WORLD.'
>>> s3 = s2.upper()
>>> s3
'HELLO WORLD.'
>>> s3.lower()    # 字母转小写
'hello world.'
>>> s2.center(50)  # 居中
'                  hello world.                  '
>>> s2.center(50, '#')  # 居中，#填充
'##################hello world.###################'
>>> s2.ljust(50, '*')  # 左对齐
'hello world.**************************************'
>>> s2.rjust(50)   # 右对齐
'                                     hello world.'
>>> 'hello-world-ni-hao'.split('-')  # 切分
['hello', 'world', 'ni', 'hao']
>>> s2.islower()   # 字母全是小写的吗？
True
>>> s2.isdigit()   # 所有字符都是数字吗？
False
```