Python官方文档中文版：https://yiyibooks.cn

1. shell相关模块

**1、shutil模块**

>>> import shutil

>>> help(shutil)

1）对象的操作

>>> shutil.copyfileobj(fsrc, fdst[, length])

#fsrc、fdst为open()对象

2）文件的操作

>>> shutil.copyfile('/etc/hosts', '/tmp/zhuji')

#src、dst都是文件

>>> shutil.copy('/etc/hosts', '/tmp/zj.txt')

>>> shutil.copy('/etc/hosts', '/tmp')

#将文件src复制到文件或目录dst

>>> shutil.copy2('/etc/hosts', '/tmp')

#与copy相同，同时拷贝文件属性

>>> shutil.move('/tmp/zhuji', '/var/tmp')

#递归移动文件或目录

3）目录的操作

>>> shutil.copytree('/etc/security', '/tmp/anquan')

>>> shutil.rmtree('/tmp/anquan')

#Tips：本模块没有删除单个文件的方法。

1. **subprocess模块**

import subprocess

#缺点：只能使用Linux命令行，不能跨平台，推荐使用OS模块

subprocess.call('id %s &> /dev/null' % username, shell=True)

subprocess.call('useradd %s' % username, shell=True)

1. **语法风格**
2. 变量赋值

>>> a, b = 10, 20

>>> a, b = b, a

1. 关键字(不能重新赋值)

>>> import keyword

#keyword模块用于查看关键字列表

>>> keyword.kwlist

['False', 'None', 'True', 'def', 'return', 'for', 'class', 'continue',... ...]

>>> keyword.iskeyword('pass')

#查看'pass'是否是关键字

3）内建(内建名称可以重新赋值)

内建函数+内建变量

1. 编程结构其布局

#!/usr/bin/env python #起始行

“this is a test module” #模块文档字符串(可以不写)

import sys #导入模块

import os

debug = True #全局变量声明

class FooClass(object): #类定义

'Foo class'

pass

def test(): #函数定义

"test funcDon"

foo = FooClass()

if \_\_name\_\_ == ‘\_\_main\_\_’: #程序主体

test()

**3、案例：创建文本文件**

# Tips：

# import os

# os.path.exists(fname): #检查文件是否存在

# with open(fname, 'w') as fobj:

# fobj.writelines(content) #使用writelines将列表写入

if \_\_name\_\_ == '\_\_main\_\_':

fname = get\_fname()

content = get\_content()

content = ['%s\n' % line for line in content]

#input写入的字符串不带'\n'，需要列表解析

wfile(fname, content)

1. 字符串详解

**1、序列的内建函数**

序列类型对象包括：字符串、列表、元组

1）内建函数

>>>ord('z') #返回ASCII码

>>> str([10, 20, 30]) #转换成字符串

'[10, 20, 30]'

>>> list('hello') #转换成列表

>>> tuple('hello') #转换成元组

>>> len('hello')

>>> alist = [random.randint(1, 100) for i in range(10)]

>>> max(alist)

>>> min(alist)

for item in enumerate(alist): #返回一个enumerate对象

print('%s:%s' % item) #item返回格式为(0, 'bob')

for ind, val in enumerate(alist):

print('%s:%s' % (ind, val))

for item in reversed(alist): #返回逆序访问的迭代器(reversed)

print(item)

print(sorted(alist)) #返回一个有序的列表，元素类型必须一致

**2、字符串**

**1）案例1：检查标识符合法性**

2）格式化操作符：

|  |  |
| --- | --- |
| **%s** | 用str()函数进行字符串转换 |
| **%d** / %i | 转成有符号十进制数 |
| %c | 转换成字符(ascii) |
| %o | 转成无符号八进制数 |
| %e / %E | 转成科学计数法 |
| %f / %F | 转成浮点数 |

>>> '%s is %s years old' % ('bob', 20)

'bob is 20 years old'

>>> '%10s%8s' % ('name', 'bob') #设置%s的总宽度

' name bob'

>>> '%-10s%-8s' % ('name', 'bob') #左对齐

'name bob '

1. format函数(与%..可相互替换)

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

'bob is 20 years old'

>>> '{name} is {age} years old'.format({'name':'bob', 'age':23})

'bob is 20 years old'

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

'bob is 20 years old'

1. **案例2：创建用户**

import subprocess

subprocess.call('id %s &> /dev/null' % username, shell=True)

subprocess.call('useradd %s' % username, shell=True)

5）保留原始字符

newPath = r"c:\windows\temp"

1. 内建函数(原字符串不变)

|  |  |
| --- | --- |
| py\_str.center(10) | 居中对齐(共10字节) |
| 'Hao123'.ljust(20, '#')  'Hao123'.rjust(20, '#') | 左对齐，使用'#'填充  右对齐 |
| py\_str.**strip('xxx')**  py\_str.lstrip()  py\_str.rstrip | 默认去除两端空白，也可以自己指定  去除左端空白  去除右端空白 |
| py\_str.upper()  py\_str.lower() | 转大写  转小写 |
| py\_str.isidentifier() | 有效标识符返回True |
| 'hao123'.islower() | 字母全小写返回True |
| 'hao123'.isdigit() | 全是数字返回True |
| py\_str.startswith('hao')  py\_str.endswith('!') | 以'hao'开头返回True  以'!'结尾返回True |
| py\_str.replace(old,new) | 将old字符替换成new字符 |
| 'hello world.tar.gz'.split('.') | 以'.'作为分隔符生成列表 |

1. 列表和元组

**1、列表**

1）列表的更新

>>> alist = [1, 2, 3, 4]

>>> alist[1:3] = [50, 60]

[1, 50, 60, 4]

>>> alist[1:1] = [20, 30, 40, 50] #在alist[1]后插入元素

[1, 20, 30, 40, 50, 50, 60, 4]

1. 内建函数

>>> alist = [10, 20, 30, 40, 50]

>>> alist.index(50) #返回下标

|  |  |
| --- | --- |
| alist.**append([1, 2])** | 向列表添加一个元素[1, 2] |
| alist.extend([1, 2]) | 向列表添加两个元素1, 2 |
| alist.index(70) | 取出70的下标 |
| alist.insert(alist.index(70)+1, 80)  alist.insert(1, 100) | 在指定的下标位置插入新值  返回[0, 100, 1, 2] |
| alist.**pop()**  alist.pop(4) | 弹出下标-1的元素  弹出下标4的元素 |
| alist.remove(70) | 删除第一个值为60的元素 |
| alist.reverse() | 翻转 |
| alist.sort(reverse=True) | 排序(默认升序、reverse=True降序) |
| blist = alist.**copy()**  blist = alist | alist的值拷贝给blist  alist、blist指向内存的同一空间 |
| alist.clear() | 清空列表 |
| alist.count(3) | 值3出现的次数 |

**2、元组**

1）创建单元素元组

atuple = ('hello',) #加一个逗号，否则视为str

1. "更新"元组

>>> atuple = ('bob', ['boy', 23])

>>> atuple[-1] = ['boy', 22]

#元组中的可变序列元素可以更新

1. 内建函数

atuple.count(10) #统计相同元素值在元组中的数量

atuple.index(10) #返回下标

**3、案例：用列表构建栈结构**

#!/usr/bin/env python3

"This is a push\_pop script"

allist = []

def push\_it():

data = input('push >> ').strip()

if data: #字符串非空才追加到列表

allist.append(data)

def pop\_it():

if not allist:

print('empty list!')

else:

print('From list popped: %s' % allist.pop())

def view\_it():

print(allist)

def show\_menu():

info = """\033[5;32m[0] push it

[1] pop it

[2] view it

[3] quit

please input >>>: \033[0m"""

cmds = {'0':push\_it, '1':pop\_it, '2':view\_it}

#使用字典产生与shell case一样的效果

#注意此处函数不能加()，否则会在字典定义时就调用函数

while True:

choice = input(info).strip()

if not choice:

continue

myin = choice[0]

if myin not in '0123':

print('\033[5;31mIvalid input. Try again.\033[0m')

continue

if myin == '3':

print('Bye-Bye')

break

cmds[myin]**()**

if \_\_name\_\_ == '\_\_main\_\_':

show\_menu()

四、字典

**1、创建字典**

adict = {'name':'bob', 'age':30}

bdict = dict(['ab', ('name', 'bob'), ['age', 18]])

> {'a': 'b', 'name': 'bob', 'age': 18}

cdict = {}.fromkeys(['tom', 'jerry'], 10)

> {'tom': 10, 'jerry': 10}

1. **访问字典**

for key in bdict: print('%s:%s' % (key, bdict[key]))

for key in bdict.keys(): print(key)

for val in bdict.values(): print(val)

for key, val in bdict.items(): print('%s:%s' % (key, val))

1. **更新与删除字典**

Tips：字典没有顺序可言

adict['age'] = 22 #更新

del adict['name'] #删除

bdict.pop('age')

cdict.clear() #清空

**4、作用于字典的函数**

len(adict)

hash() #查看对象是否可哈希(作为字典的key)，必须是不可变类型

**5、字典内建函数**

hdict = bdict.copy() #创建副本

bdict.get('name')

bdict.get('work', 'not found')

#key存在，返回val；

#key不存在，默认返回None，也可自定义返回值

bdict.setdefault('work', 'teacher')

#'work'已存在，不改变val

#'work'不存在，向字典添加新键值对

adict.update({'work': 'teacher'})

#adict与{'work': 'teacher'}合并，已存在的key也会更新val

1. **案例**

**1：模拟用户登陆信息系统**

1）支持新用户注册,新用户名和密码注册到字典中

2）支持老用户登陆,用户名和密码正确提示登陆成功

3）主程序通过循环询问进行何种操作,根据用户的选择,执行注册或是登陆操作

**2：编写unix2dos的程序**

1）Windows文本文件的行结束标志是\r\n

2）类unix文本文件的行结束标志是\n

3）编写程序,将unix文本文件格式转换为windows文本文件的格式

注意：\r表示回车不换行，回到行首以替换形式输入

**3：编写类进度条程序**

1）在屏幕上打印20个#号

2）符号@从20个#号穿过

3）当@符号到达尾部,再从头开始

import time

print('#' \* 20, end='')

n = 0

while True:

time.sleep(0.3)

print('\r%s@%s' % ('#' \* n, '#' \* (19 - n)), end = '')

n += 1

if n == 20:

n = 0

#拷贝文件进度条显示：(参考Python百例)

[root@room9pc01 ~]# pip install tqdm

>>> import tqdm

>>> for i in tqdm.tqdm(range(10))

五、集合

由**不同的不可变**元素构成，它像是没有val的字典。

1. **集合的操作符**

>>> aset = set('abc') #创建可变集合

{'a', 'b', 'c'}

>>> bset = set('bcd')

>>> len(aset)

>>> for ch in aset: print(ch)

>>> aset & bset #交集{'b', 'c'}，还是set类型

>>> aset | bset #并集{'a', 'b', 'c', 'd'}

>>> aset - bset #差补，只在aset中存在的元素{'a'}

**2、集合内建函数**

>>> cset.add('new') #添加元素'new'

>>> cset.update('new') #添加元素'n' 'e' 'w'

>>> cset.update(['abc', 'hello']) #添加元素'abc' 'hello'

>>> dset = set('hello')

>>> dset.issubset(cset) #dset是cset的子集吗？

>>> cset.issuperset(dset) #cset是dset的超集吗？

>>> aset.intersection(bset) #aset & bset

>>> aset.difference(bset) #aset - bset

>>> aset.union(bset) #aset | bset

1. **案例：file\_diff.py**

要求：对比两天的web日志，返回在昨天没被访问而在今天被访问的网站。

# cp /etc/passwd /tmp/mima

# 修改/tmp/mima，使之有些行与/etc/passwd不一样

with open('/etc/passwd') as fobj:

aset = set(fobj) #把文件对象转换成集合

with open('/tmp/mima') as fobj:

bset = set(fobj)

with open('/tmp/myfile', 'w') as fobj:

fobj.writelines(bset - aset)