**Python1-4**

**• shutil.copyfileobj(fsrc, fdst[, length])将类似文件的对象fsrc的内容复制到类似文件的对象fdst。**

**• shutil.copyfile(src, dst, \*, follow\_symlinks=True)将名为src的文件的内容(无元数据)复制到名为dst的文件,然后返回dst。**

**• shutil.copy(src, dst, \*, follow\_symlinks=True)将文件src复制到文件或目录dst。src和dst应为字符串。如果dst指定目录,则文件将使用src的基本文件名复制到dst中。返回新创建的文件的路径。**

**• shutil.copy2(src, dst, \*, follow\_symlinks=True)与copy()相同,但copy2()也尝试保留所有文件元数据。**

**• shutil.move(src, dst, copy\_function=copy2)递归地将文件或目录(src)移动到另一个位置(dst),并返回目标。**

**• shutil.copytree(src, dst, symlinks=False, ignore=None, copy\_function=copy2, ignore\_dangling\_symlinks=False)递归地复制以src为根的整个目录树,返回目标目录。由dst命名的目标目录不能已经存在。**

**• shutil.rmtree (path, ignore\_errors=False, onerror=None)**

**删除整个目录树; 路径必须指向目录(而不是指向目录的符号链接)。**

**• shutil.copymode(src, dst, \*, follow\_symlinks=True)将权限位从src复制到dst。文件内容,所有者和组不受影响。src和dst是以字符串形式给出的路径名称。**

**• shutil.copystat(src, dst, \*, follow\_symlinks=True)将权限位,最后访问时间,上次修改时间和标志从src复制到dst。**

**• shutil.chown(path, user=None, group=None)更改给定路径的所有者用户和/或组**

**变量赋值**

**• python支持链式多重赋值**

**x = y = 10**

**• 另一种将多个变量同时赋值的方法称为多元赋值,采用这种方式赋值时,等号两边的对象都是元组a, b = 10, 20 相当于 a=10; b=20**

**>>> a,b = b,a ab 的值互换**

**合法标识符 包括 变量 模块类 函数**

**• python标识符字符串规则和其他大部分用C编写的高级语言相似**

**• 第一个字符必须是字母或下划线(\_)**

**• 剩下的字符可以是字母和数字或下划线**

**• 大小写敏感**

**关键字 ：**

**• 和其他的高级语言一样,python也拥有一些被称作关键这字的保留字符**

**• 任何语言的关键字应该保持相对的稳定,但是因为python是一门不断成长和进化的语言,其关键字偶尔会更新**

**• 关键字列表和iskeyword()函数都放入了keyword模块以便查阅**

**>>> keyword.kwlist**

**<https://yiyibooks.cn> python352可以看库参考**

**内建**

**• 除了关键字之外,python还有可以在任何一级代码使用的“内建”的名字集合,这些名字可以由解释器设置或使用**

**• 虽然built-in不是关键字,但是应该把它当作“系统保留字”**

**• 保留的常量如:True、False、None**

**模块结构及布局**

**• 编写程序时,应该建立一种统一且容易阅读的结构,并将它应用到每一个文件中去**

**#!/usr/bin/env python #起始行**

**“this is a test module” #模块文档字符串**

**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()**

**编写程序的步骤**

**1 先想想程序是怎么运行的(交互 非交互)**

**2 交互的 考虑屏幕提示 用户回答**

**3 思考程序功能 将这些功能创建函数 编写程序框架**

**Def**

**....**

**....**

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

**1. 编写一个程序,要求用户输入文件名**

**2. 如果文件已存在,要求用户重新输入**

**3. 提示用户输入数据,每行数据先写到列表中**

**4. 将列表数据写入到用户输入的文件名中**

**import** os  
**def** f\_name():  
 **while True**:  
 x = input(**'输入文件名:'**)  
 **if not** os.path.exists(x):  
 **break** print(**'文件已经存在'**)  
 **return** x  
**def** s\_name():  
 content = []  
 **while True**:  
 y = input(**'请输入数据("end"结束):'**)  
 **if** y == **'end'**:  
 **break** content.append(y + **'\n'**)  
 **return** content  
**def** file(x,content):  
 **with** open(x,**'a'**) **as** f :  
 f.writelines(content)  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 x = f\_name()  
 content = s\_name()  
 *# content = [y + '\n' for y in content]* file(x,content )

**序列类型操作符**

|  |  |
| --- | --- |
| **序列操作符** | **作用** |
| **seq[ind]** | **获得下标为ind的元素** |
| **seq[ind1:ind2]** | **获得下标从ind1到ind2间的元素集合** |
| **seq \* expr** | **序列重复expr次** |
| **seq1 + seq2** | **连接序列seq1和seq2** |
| **obj in seq** | **判断obj元素是否包含在seq中** |
| **obj not in seq** | **判断obj元素是否不包含在seq中** |

**内建函数**

|  |  |
| --- | --- |
| **函数** | **含义** |
| **list(iter)** | **把可迭代对象转换为列表** |
| **str(obj)** | **把obj对象转换成字符串** |
| **tuple(iter)** | **把一个可迭代对象转换成一个元组对象** |

**>>> list('hello')**

**['h', 'e', 'l', 'l', 'o']**

**>>> list(('hello', 'world'))**

**['hello', 'world']**

**• len(seq):返回seq的长度**

**• max(iter,key=None):返回iter中的最大值**

**• enumerate:接受一个可迭代对象作为参数,返回一个enumerate对象**

**• reversed(seq):接受一个序列作为参数,返回一个以逆序访问的迭代器 翻转**

**• sorted(iter):接受一个可迭代对象作为参数,返回一个有序的列表 排序**

**字符串操作符**

**• 比较操作符:字符串大小按ASCII码值大小进行比较**

**• 切片操作符:[ ]、[ : ]、[ : : ]**

**• 成员关系操作符:in、not in**

**1. 程序接受用户输入**

**2. 判断用户输入的标识符是否合法**

**3. 用户输入的标识符不能使用关键字**

**4. 有不合法字符,需要指明第几个字符不合法**

**import** string  
**import** keyword  
*# x = input('请输入标示符：')***def** adc(x):  
 **if** x[0] **not in** string.ascii\_letters+**'\_'**:  
 **return '标示符不合法'  
 if** keyword.iskeyword(x):  
 **return '标示符是关键字'  
 for** i,y **in** enumerate(x[1:]):  
 **if** y **not in** string.ascii\_letters+string.digits:  
 **return'第%s个字符不合法'** % (i +2)  
  
 **return '%s 是合法标示符'** % x  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 x = input(**'请输入标示符：'**)  
 print(adc(x))

**字符串可以使用格式化符号来表示特定含义**

|  |  |  |
| --- | --- | --- |
| **格式化字符** | | **转换方式** |
| **%c** | | **转换成字符** |
| **%s** | | **优先用str()函数进行字符串转换** |
| **%d / %i** | | **转成有符号十进制数** |
| **%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** |
| **辅助指令** | **作用** | |
| **\*** | **定义宽度或者小数点精度** | |
| **-** | **左对齐** | |
| **+** | **在正数前面显示加号** | |
| **<sp>** | **在正数前面显示空格** | |
| **#** | **在八进制数前面显示零0,在十六进制前面显示'0x'或者'0X'** | |
| **0** | **显示的数字前面填充0而不是默认的空格** | |

**format函数**

**• 使用位置参数**

**– 'my name is {} ,age {}'.format('hoho',18)**

**• 使用关键字参数**

**– 'my name is {name},age is**

**{age}'.format({'name':'bob', 'age':23})**

**• 填充与格式化**

**– {:[填充字符][对齐方式 <^>][宽度]}**

**• 使用索引**

**– 'name is {0[0]} age is {0[1]}'.format(['bob', 23])**

**1. 编写一个程序,实现创建用户的功能**

**2. 提示用户输入用户名**

**3. 随机生成8位密码**

**4. 创建用户并设置密码**

**5. 将用户相关信息写入指定文件**

**import** sys  
**import** subprocess  
**import** string  
**import** random  
password = **''**.join([random.choice(string.digits+string.ascii\_letters) **for** i **in** range(8)])**def** adduser(name,password,fname):  
 info = **"""用户信息：  
 用户名 %s  
 密码 %s  
 """** % (name, password)  
 subprocess.call(**'useradd %s'** % name, shell=**True**)  
 subprocess.call(**'echo %s | passwd --stdin %s'** % (password,name),shell=**True**)  
 **with** open(fname,**'a'**) **as** c:  
 c.write(info)  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 fname = **'/tmp/bbb.txt'** adduser(sys.argv[1],password,fname)

**• 原始字符串操作符是为了对付那些在字符串中出现的特殊字符**

**• 在原始字符串里,所有的字符都是直接按照字面的意思来使用,没有转义特殊或不能打印的字符**

**>>> winPath = "c:\windows\temp"**

**>>> print(winPath)**

**c:\windows emp**

**>>> newPath = r"c:\windows\temp"**

**>>> print(newPath)**

**c:\windows\temp**

**1. 提示用户输入(多行)数据**

**2. 假定屏幕的宽度为50,用户输入的多行数据如下显示(文本内容居中):**

**def** asa():  
 *# x = input('请输入多行数据（end结束）：')* content = []  
 *# print('+' + '\*' \* 50 + '+')* **while True**:  
 x = input(**'请输入多行数据（end结束）：'**)  
 *# y = '+'+x.center(50)+'+'* **if** x == **'end'**:  
 **break** content.append(x)  
 print(**'+'** + **'\*'** \* 50 + **'+'**)  
 **for** line **in** content:  
 print(**'+'** + line.center(50) + **'+'**)  
 print(**'+'** + **'\*'** \* 50 + **'+'**)  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 asa()

**• string.capitalize():把字符串的第一个字符大写**

**• string.center(width):返回一个原字符串居中,并使用空格填充至长度width 的新字符串**

**• string.count(str, beg=0,end=len(string)):返回str在string里面出现的次数,如果beg或者end指定则返回指定范围内str出现的次数**

**1. 栈是一个后进先出的结构**

**2. 编写一个程序,用列表实现栈结构**

**3. 需要支持压栈、出栈、查询功能**

content**=[]  
def push\_it():** item **=** input**('数据:')**.strip**()** content.append**(**item**)  
def pop\_it():  
 if** content**:** print **('弹出%s' %** content.pop**())  
 else:** print**('\033[31;1m空栈\033[0m')  
def view\_it():** print**('\033[32;1m%s\033[0m' %** content**)  
def show\_enmu():** cmds**={'0':**push\_it,**'1':**pop\_it,**'2':**view\_it**}** prompt**="""(0)压栈  
 (1)出栈  
 (2)查询  
 (3)退出  
 请输入(0/1/2/3/)"""  
 while True:** choice **=** input**(**prompt**)**.strip**()[**0**]  
 if** choice **not in '0123':** print **('无效输入，请重试')  
 continue  
 if** choice **== '3':** print**('Bye-Bye')  
 break** # if choice == '0':  
 # push\_it()  
 # elif choice =='1':  
 # pop\_it()  
 # else:  
 # view\_it()  
 #  
 cmds**[**choice**]()  
if** \_\_name\_\_ **== '\_\_main\_\_':** show\_enmu**()**