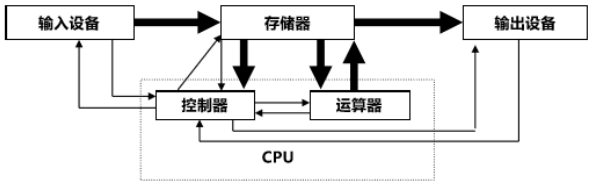
计算机基础



关键字

['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or', 'pass', 'print', 'raise', 'return', 'try', 'while', 'with', 'yield']

基本对象类型

字符串（string）
整数（integer）八进制（025）十六进制（$0x15$）
浮点数（float）
布尔数（boolean）
复数（complex）$3+0.5j$

运算符

运算符优先级从低到高：

lambda	Lambda 表达式
or	布尔“或”
and	布尔“与”
not	布尔“非”
in，not in	成员测试
is，is not	同一性测试
<，<=，>，>=，!=，==	比较
|	按位或
^	按位异或
&	按位与
«，»	移位
+，-	加法与减法
*，/，%	乘法、除法与取余
+x，-x	正负号
x	按位翻转
**	指数
x.attribute	属性参考
x[index]	下标
x[index:index]	寻址段
f(arguments...)	函数调用
(experession,...)	绑定或元组显示
$[expression,...]$	列表显示
key:datum,...	字典显示
'expression,...'	字符串转换

函数

函数中的参数名称为“形参”，调用函数时传递的值为“实参”。
为定义在函数外的全局变量赋值需用 global。
只有在形参表末尾的那些参数可以有默认参数值，即不能在声明函数形参的时候，先声明有默认值的形参而后声明没有默认值的形参。
func(a=1,b=2)～func(**"a":1,"b":2)～func(*(1,2))
函数只能有一个返回值，但是该值可以是一组值, 如返回一个元组。

字符串 str

str.capitalize() Return a copy of the string with its first character capitalized and the rest lowercased.
str.center(width[, fillchar]) Return centered in a string of length width. Padding is done using the specified fillchar (default is a space).
str.count(sub[, start[, end]]) Return the number of non-overlapping occurrences of substring sub in the range [start, end]. Optional arguments start and end are interpreted as in slice notation.
str.find(sub[, start[, end]]) Return the lowest index in the string where substring sub is found, such that sub is contained in the slice s[start:end]. Optional arguments start and end are interpreted as in slice notation. Return -1 if sub is not found.
str.ljust(width[, fillchar]) Return the string left justified in a string of length width. Padding is done using the specified fillchar (default is a space). The original string is returned if width is less than or equal to len(s).
str.center(width[, fillchar]) Return centered in a string of length width. Padding is done using the specified fillchar (default is a space).

字符串格式化

format_spec	[[fill]align][sign][#][0][width][,][.precision][type]
fill	<any character>
align	"<" | ">" | "=" | "^"
sign	"+" | "-" | " "
width	integer
precision	integer
type	"b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | "x" | "X" | "%"

列表 list ([..,..])

L.append(var) 追加元素
L.insert(index,var) 在 index 位置插入 var
L.pop(var) 返回最后一个元素，并从 list 中删除之
L.remove(var) 删除第一次出现的该元素
L.count(var) 该元素在列表中出现的个数
L.index(var) 该元素的位置, 无则抛异常
L.extend(list) 追加 list，即合并 list 到 L 上
L.sort() 排序
L.reverse() 倒序
list 操作符:,+,*，关键字 del
a[1:] 片段操作符，用于子 list 的提取
[1,2]+[3,4] 为 [1,2,3,4]。同 extend()
[2]*4 为 [2,2,2,2]
del L[1] 删除指定下标的元素
del L[1:3] 删除指定下标范围的元素

字典 dict ({:,..})

D.get(key, 0) 同 dict[key]，多了个没有则返回缺省值，0。[] 没有则抛异常
D.has_key(key) 有该键返回 True，否则 False
D.keys() 返回字典键的列表
D.values() 返回字典值的列表
D.items() 返回可遍历的字典列表，每一个键值对为一个 tuple
D.update(dict2) 增加合并字典
D.popitem() 得到一个 pair，并从字典中删除它。已空则抛异常
D.clear() 清空字典，同 del dict
D.copy() 拷贝字典
D.cmp(dict1,dict2) 比较字典，(优先级为元素个数、键大小、键值大小), 第一个大返回 1，小返回-1，一样返回 0

集合 set,frozenset **(** set([..], frozenset([..]))

isdisjoint(other) Return True if the set has no elements in common with other. Sets are disjoint if and only if their intersection is the empty set.

issubset(other) set <= other Test whether every element in the set is in other.

set < other Test whether the set is a proper subset of other, that is, set <= other and set != other.

issuperset(other) set >= other Test whether every element in other is in the set.

set > other Test whether the set is a proper superset of other, that is, set >= other and set != other.

union(other, ...) set | other | ... Return a new set with elements from the set and all others.

intersection(other, ...) set & other & ... Return a new set with elements common to the set and all others.

difference(other, ...) set - other - ... Return a new set with elements in the set that are not in the others.

symmetric_difference(other) set ^ other Return a new set with elements in either the set or other but not both.

copy() Return a new set with a shallow copy of s.

Note, the non-operator versions of union(), intersection(), difference(), and symmetric_difference(), issubset(), and issuperset() methods will accept any iterable as an argument. In contrast, their operator based counterparts require their arguments to be sets. This precludes error-prone constructions like set('abc') & 'cbs' in favor of the more readable set('abc').intersection('cbs').

update(other, ...) set |= other | ... Update the set, adding elements from all others.

intersection_update(other, ...) set &= other & ... Update the set, keeping only elements found in it and all others.

difference_update(other, ...) set -= other | ... Update the set, removing elements found in others.

symmetric_difference_update(other) set ^= other Update the set, keeping only elements found in either set, but not in both.

以下方法 set 支持，frozenset 不支持

add(elem) Add element elem to the set.

remove(elem) Remove element elem from the set. Raises KeyError if elem is not contained in the set.

discard(elem) Remove element elem from the set if it is present.

pop() Remove and return an arbitrary element from the set. Raises KeyError if the set is empty.

数据结构性质总结

	string	list	tuple	set	dict
Mutable	No	Yes	No	Yes	Yes
Sequential	Yes	Yes	Yes	No	No
Sortable	No	Yes	No	No	No
Slicable	Yes	Yes	Yes	No	No
Index/key type	Int	Int	Int	Immut	Immut
Item/value type	Char	Any	Any	No	Any
Search complexity	O(n)	O(n)	O(n)	O(1)	O(1)

正则表达式 re

ptn = re.compile(r'..') #ptn is a Pattern

Pattern 属性:

pattern: 编译时用的表达式字符串。

flags: 编译时用的匹配模式。数字形式。

groups: 表达式中分组的数量。

groupindex: 以表达式中有别名的组的别名为键、以该组对应的编号为值的字典，没有别名的组不包含在内。

Pattern 实例方法 [| re 模块方法]：

match(string[, pos[, endpos]]) | re.match(pattern, string[, flags]) 这个方法将从 string 的 pos 下标处起尝试匹配 pattern；如果 pattern 结束时仍可匹配，则返回一个 Match 对象；如果匹配过程中 pattern 无法匹配，或者匹配未结束就已到达 endpos，则返回 None。pos 和 endpos 的默认值分别为 0 和 len(string)；re.match() 无法指定这两个参数，参数 flags 用于编译 pattern 时指定匹配模式。注意：这个方法并不是完全匹配。当 pattern 结束时若 string 还有剩余字符，仍然视为成功。想要完全匹配，可以在表达式末尾加上边界匹配符'\$'。

search(string[, pos[, endpos]]) | re.search(pattern, string[, flags]) 这个方法用于查找字符串中可以匹配成功的子串。从 string 的 pos 下标处起尝试匹配 pattern，如果 pattern 结束时仍可匹配，则返回一个 Match 对象；若无法匹配，则将 pos 加 1 后重新尝试匹配；直到 pos=endpos 时仍无法匹配则返回 None。pos 和 endpos 的默认值分别为 0 和 len(string))；re.search() 无法指定这两个参数，参数 flags 用于编译 pattern 时指定匹配模式。

split(string[, maxsplit]) | re.split(pattern, string[, maxsplit]) 按照能够匹配的子串将 string 分割后返回列表。maxsplit 用于指定最大分割次数，不指定将全部分割。

findall(string[, pos[, endpos]]) | re.findall(pattern, string[, flags]) 搜索 string，以列表形式返回全部能匹配的子串。

finditer(string[, pos[, endpos]]) | re.finditer(pattern, string[, flags]) 搜索 string，返回一个顺序访问每一个匹配结果（Match 对象）的迭代器。

sub(repl, string[, count]) | re.sub(pattern, repl, string[, count]) 使用 repl 替换 string 中每一个匹配的子串后返回替换后的字符串。当 repl 是一个字符串时，可以使用 \id 或 \g<id>、\g<name> 引用分组，但不能使用编号 0。当 repl 是一个方法时，这个方法应当只接受一个参数（Match 对象），并返回一个字符串用于替换（返回的字符串中不能再引用分组）。count 用于指定最多替换次数，不指定时全部替换。

subn(repl, string[, count]) |re.sub(pattern, repl, string[, count]) 返回 (sub(repl, string[, count]), 替换次数)。

语法	说明	表达式实例	完整匹配的字符串
字符			
一般字符	匹配自身	abc	abc
.	匹配任意除换行符"\n"外的字符。 在DOTALL模式中也能匹配换行符。	a.c	abc
\	转义字符，使后一个字符改变原来的意思。 如果字符串中有字符*需要匹配，可以使用*或者字符集[*]。	a\.c a\\c	a.c a\c
[...]	字符集（字符类）。对应的位置可以是字符集中任意字符。 字符集中的字符可以逐个列出，也可以给出范围，如[abc]或[a-c]。第一个字符如果是^则表示取反，如[^abc]表示不是abc的其他字符。 所有的特殊字符在字符集中都失去其原有的特殊含义。在字符集中如果要使用]、-或^，可以在前面加上反斜杠，或把]、-放在第一个字符，把^放在非第一个字符。	a[bcd]e	abe ace ade
预定义字符集（可以写在字符集[...]中）			
\d	数字：[0-9]	a\dc	a1c
\D	非数字：[^\d]	a\Dc	abc
\s	空白字符：[<空格>\t\r\n\f\v]	a\sc	a c
\S	非空白字符：[^\s]	a\Sc	abc
\w	单词字符：[A-Za-z0-9_]	a\wc	abc
\W	非单词字符：[^\w]	a\Wc	a c
数量词（用在字符或(...)之后）			
*	匹配前一个字符0或无限次。	abc*	ab abccc
+	匹配前一个字符1次或无限次。	abc+	abc abccc
?	匹配前一个字符0次或1次。	abc?	ab abc
{m}	匹配前一个字符m次。	ab{2}c	abbc
{m,n}	匹配前一个字符m至n次。 m和n可以省略：若省略m，则匹配0至n次；若省略n，则匹配m至无限次。	ab{1,2}c	abc abbc
*? +? ?? {m,n}?	使 * + ? {m,n}变成非贪婪模式。	示例将在下文中介绍。	
边界匹配（不消耗待匹配字符串中的字符）			
^	匹配字符串开头。 在多行模式中匹配每一行的开头。	^abc	abc
$	匹配字符串末尾。 在多行模式中匹配每一行的末尾。	abc$	abc
\A	仅匹配字符串开头。	\Aabc	abc
\Z	仅匹配字符串末尾。	abc\Z	abc
\b	匹配\w和\W之间。	a\b!bc	a!bc
\B	[^\b]	a\Bbc	abc
逻辑、分组			
|	|代表左右表达式任意匹配一个。 它总是先尝试匹配左边的表达式，一旦成功匹配则跳过匹配右边的表达式。 如果|没有被包括在()中，则它的范围是整个正则表达式。	abc|def	abc def
(...)	被括起来的表达式将作为分组，从表达式左边开始每遇到一个分组的左括号'('，编号+1。 另外，分组表达式作为一个整体，可以后接数量词。表达式中的|仅在该组中有效。	(abc){2} a(123|456)c	abcabc a456c
(?P<name>...)	分组，除了原有的编号外再指定一个额外的别名。	(?P<id>abc){2}	abcabc
\<number>	引用编号为<number>的分组匹配到的字符串。	(\d)abc\1	1abc1 5abc5
(?P=name)	引用别名为<name>的分组匹配到的字符串。	(?P<id>\d)abc(?P=id)	1abc1 5abc5
特殊构造（不作为分组）			
(?:...)	(...)的不分组版本，用于使用'|'或后接数量词。	(?:abc){2}	abcabc
(?iLmsux)	iLmsux的每个字符代表一个匹配模式，只能用在正则表达式的开头，可选多个。匹配模式将在下文中介绍。	(?i)abc	AbC
(?#...)	#后的内容将作为注释被忽略。	abc(?#comment)123	abc123
(?=...)	之后的字符串内容需要匹配表达式才能成功匹配。 不消耗字符串内容。	a(?=\d)	后面是数字的a
(?!...)	之后的字符串内容需要不匹配表达式才能成功匹配。 不消耗字符串内容。	a(?!\d)	后面不是数字的a
(?<=...)	之前的字符串内容需要匹配表达式才能成功匹配。 不消耗字符串内容。	(?<=\d)a	前面是数字的a
(?<!...)	之前的字符串内容需要不匹配表达式才能成功匹配。 不消耗字符串内容。	(?<!\d)a	前面不是数字的a
(?(id/name) yes-pattern |no-pattern)	如果编号为id/别名为name的组匹配到字符，则需要匹配yes-pattern，否则需要匹配no-pattern。 |no-patern可以省略。	(\d)abc(?(1)\d|abc)	1abc2 abcabc