import re

text = "字符串"

ret = re.match('正则表达式',text）

print(ret.group())

**match是从字符串第一个字符开始匹配**

**\d匹配纯数字（仅仅匹配一个字符）：**

text = "12345678"

ret =re.match('\d',text）

print(ret.group())

输出1

**\D匹配非数字（仅匹配一个字符）：**

text = "a2345678"

ret =re.match('\D',text）

print(ret.group())

输出a

**\w匹配字母,下划线,数字（与变量的命名先相同）：**

text = "a2345678"

ret =re.match('\w\w',text）

print(ret.group())

输出a2

**\W与\w的匹配相反：**

text = "+-\*/"

ret =re.match('\W',text）

print(ret.group())

输出 +

**\s的只匹配特殊标识符”\n \r”之类：**

text = "\n"

ret =re.match('\s',text）

print(ret.group())

输出的是 换行的空白

**.匹配任意字符：**

text1 = "123"

text2 = "abc"

ret1 =re.match('.',text）

ret2 =re.match('.',text）

print(ret1.group(),ret2.group())

输出的是 换行的空白

**[]用来组合自己想匹配的内容，只匹配[]内字符：**

text = "abc"

ret =re.match('[abcdefg]',text）

print(ret.group())

输出 a

**用[]来组合成\d,\D,\w,\W:**

**\d: [0-9] (-意思是从0到9的10个数字)**

**\D: [^0-9] (^意思是非，匹配除了0到9的数字)**

**\w: [a-zA-Z0-9\_] (匹配a到z，A到Z，0到9和\_)**

**\W:[^a-zA-Z0-9\_]**

**匹配多个字符**

**\*配合重复\*之前条件，匹配多个任意字符：**

text = "123456789"

ret =re.match('\d\*',text)

print(ret.group())

**输出 123456789**

text = "123456789"

ret =re.match('1\*',text)

print(ret.group())

**只输出1**

text = "111123456789"

ret =re.match('1\*',text)

print(ret.group())

**输出 1111**

**+匹配一个或多个任意字符（与\*相似）：**

text = "abcdefg"

ret =re.match('\w+',text)

print(ret.group())

**输出 abcdefg**

**\*与+的区别：**

**\*与+的匹配模式相同**

text = "12aab12abc12a"

ret =re.findall('ab+',text)

print(ret)

**输出ab ab**

text = "12aab12abc12a"

ret =re.findall('ab\*',text)

print(ret)

**输出a ab ab a**

**同样是匹配ab**

**但是ab\*比ab+多了 a 的匹配**

**？只匹配一次或0次：**

text = "abc"

ret =re.match('\w?',text)

print(ret.group())

**输出 a 只匹配到第一个符合的就停止向下匹配**

**{n} 用{}来选择自己想匹配多少个结果：**

text = "abcdef"

ret =re.match('\w{3}',text)

print(ret.group())

**输出 abc**

**{n，m}匹配n-m个结果：**

text = "abcdef"

ret =re.match('\w{3,5}',text)

print(ret.group())

**输出 abcde**

text = "abc"

ret =re.match('\w{3}',text)

print(ret.group())

**输出 abc**

**开始和结束匹配符：**

**^表示开始：**

text = "abcdef"

ret =re.search('^a\w\w',text)

print(ret.group())

**输出 abc**

text = "bcdef"

ret =re.search('^a\w\w',text)

print(ret.group())

**则无法匹配**

**$表示结束：**

text = "abcdef"

ret =re.search('a\*f$',text)

print(ret.group())

**输出 abcdef**

text = "abcde"

ret =re.search('a\*f$',text)

print(ret.group())

**则无法匹配**

**| 意思是 或 与c语言||用法相同：**

text = "abcde"

ret =re.search('(ab|cd)',text)

print(ret.group())

**输出ab**

**贪婪模式与非贪婪模式**

text = "<h1>标题<h1>"

ret =re.search('<.\*>',text)

print(ret.group())

**此时为贪婪模式，这时候这里的尖括号分别是最前的<和最后的> 返回的是**<h1>标题<h1>

text = "<h1>标题<h1>"

ret =re.search('<.\*>?',text)

print(ret.group())

**而非贪婪模式，则是在最后加?防止向后继续查找，返回的便是**<h1>

**\转义字符：**

text = "\\w"

ret =re.search('\\\w',text)

print(ret.group())

**因为\w是正则表达式的标识符，如果想查找\w的话，就在前加\转义\**

text = "."

ret =re.search('\.',text)

print(ret.group())

**.代表匹配任意字符，但是想匹配.的话就得把.加上\转义**

**\.**

**r” ” 原生字符串：**

**字符串” ”引号前加r，则引号内的标识符不做处理按原样输出**

**通常print(“\n”)输出为空，但是用原生字符串的处理方法print(r“\n”)就能输出\n**

**re中函数：**

**match()：**

**只会从一个字符串的开始位置，进行匹配**

**search():**

**可以从字符串中，任意位置开始匹配**

**group():分组函数：**

**一个括号代表一组**

text = "abcd1234"

ret =re.search('([a-d]+)(\d+)',text)

print(ret.group())

ret.group(1)**为第一组，返回的是abcd**

ret.group(2)**为第二组，返回的是1234**

**findall():**

text = "12aab12abc12a"

ret =re.findall('ab+',text)

print(ret)

**返回的是找到所有符合匹配的结果的列表['ab', 'ab']**

**sub():替换字符串函数：**

text = "abc"

ret =re.sub('abc','123',text) 第一个参数为原字符串，第二个参数为替换

print(ret)

**返回的是一个字符串 结果为123**

**split()分割字符串函数：**text = "a|b|c"

ret =re.split(',',text)

print(ret)

**参数为 以什么分割**

**返回一个列表['a', 'b', 'c']**

**compile():可以把正则表达式先编译起来放在变量当中**

text = "abcd1234"

r =re.compile('\w\*',text)

ret =re.search(r,text)

print(ret.group())

**compile中的正则表达式可以加注释。**