OS 文件/目录方法

os 模块用来处理文件和目录。

- import os:模块导入；
- os.getcwd()作用：获取当前的工作路径；
- os.listdir(path)作用：传入任意一个path路径，返回的是该路径下所有文件和目录，组成的列表；
- os.walk(path)含义 ：传入任意一个path路径，深层次遍历指定路径下的所有子文件夹，返回的是一个由路径、文件夹列表、文件列表组成的元组；
- os.path.exists(path)含义：传入一个path路径，判断指定路径下的目录是否存在。存在返回True，否则返回False；
- os.mkdir(path)含义：传入一个path路径，创建单层(单个)文件夹；
- os.makedirs(path)含义：传入一个path路径，生成一个递归的文件夹；
- os.rmdir(path)含义：传入一个path路径，删除指定路径下的文件夹；
- os.path.join(path1,path2)含义：传入两个path路径，将该路径拼接起来，形成一个新的完整路径；
- os.path.split(path)含义：传入一个完整的path路径，将其拆分为绝对路径和文件名2部分；
- os.path.dirname(path)含义：传入一个完整的文件路径，只获取其绝对路径；
- os.path.basename(path)含义：传入一个完整的文件路径，只获取其文件名;
- os.path.basename(path)含义：传入一个完整的文件路径，只获取其文件名;
- os.path.isdir(path)含义：传入一个完整的文件路径，判断它是否是文件夹；
- os.path.isfile(path)含义：传入一个完整的文件路径，判断它是否是文件;
- os.path.sep含义：返回当前操作系统的路径分隔符；
- os.path.getsize(path)含义：传入一个完整的文件路径，返回该文件的大小。

正则表达式

正则表达式是一个特殊的字符序列，检查一个字符串是否与某种模式匹配。

- re.match函数：从字符串的起始位置匹配一个模式，如果不是起始位置匹配成功的话，match() 就返回 none。
- re.search函数：扫描整个字符串并返回第一个成功的匹配。
- re.sub函数：用于替换字符串中的匹配项。
- re.compile函数：用于编译正则表达式，生成一个正则表达式对象，供 match() 和 search() 这两个函数使用。
- findall函数：在字符串中找到正则表达式所匹配的所有子串，并返回一个列表，如果有多个匹配模式，则返回元组列表，如果没有找到匹配的，则返回空列表。
- re.finditer函数：在字符串中找到正则表达式所匹配的所有子串，并把它们作为一个迭代器返回。
- re.split：split 方法按照能够匹配的子串将字符串分割后返回列表。

正则表达式修饰符 - 可选标志

正则表达式可以包含一些可选标志修饰符来控制匹配的模式。修饰符被指定为一个可选的标志。多个标志可以通过按位 OR(|) 它们来指定。

- re.I：使匹配对大小写不敏感。
- re.L：做本地化识别（locale-aware）匹配。

- re.M：多行匹配，影响 ^ 和 $。
- re.S：使 . 匹配包括换行在内的所有字符。
- re.U：根据Unicode字符集解析字符。这个标志影响 \w, \W, \b, \B。
- re.X：该标志通过给予你更灵活的格式以便你将正则表达式写得更易于理解。

正则表达式模式

- ^：匹配字符串的开头。
- $：匹配字符串的末尾。
- .：匹配任意字符，除了换行符，当re.DOTALL标记被指定时，则可以匹配包括换行符的任意字符。
- [...]：用来表示一组字符,单独列出：[amk] 匹配 'a'，'m'或'k'。
- [^ ...]：不在[]中的字符：[^abc] 匹配除了a,b,c之外的字符。
- re*：匹配0个或多个的表达式。
- re+：匹配1个或多个的表达式。
- re?：匹配0个或1个由前面的正则表达式定义的片段，非贪婪方式。
- re{ n}：精确匹配 n 个前面表达式。例如， o{2} 不能匹配 "Bob" 中的 "o"，但是能匹配 "food" 中的两个 o。
- re{ n,}：匹配 n 个前面表达式。例如， o{2,} 不能匹配"Bob"中的"o"，但能匹配 "fooooood"中的所有 o。"o{1,}" 等价于 "o+"。"o{0,}" 则等价于 "o*"。
- re{ n, m}：匹配 n 到 m 次由前面的正则表达式定义的片段，贪婪方式。
- a| b：匹配a或b。
- (re)：对正则表达式分组并记住匹配的文本。
- (?imx)：正则表达式包含三种可选标志：i, m, 或 x 。只影响括号中的区域。
- (?-imx)：正则表达式关闭 i, m, 或 x 可选标志。只影响括号中的区域。
- (?: re)：类似 (...), 但是不表示一个组。
- (?imx: re)：在括号中使用i, m, 或 x 可选标志。
- (?-imx: re)：在括号中不使用i, m, 或 x 可选标志。
- (?#...)：注释。
- (?= re)：前向肯定界定符。如果所含正则表达式，以 ... 表示，在当前位置成功匹配时成功，否则失败。但一旦所含表达式已经尝试，匹配引擎根本没有提高；模式的剩余部分还要尝试界定符的右边。
- (?! re)：前向否定界定符。与肯定界定符相反；当所含表达式不能在字符串当前位置匹配时成功。
- (?> re)：匹配的独立模式，省去回溯。
- \w：匹配字母数字及下划线。
- \W：匹配非字母数字及下划线。
- \s：匹配任意空白字符，等价于 [\t\n\r\f]。
- \S：匹配任意非空字符。
- \d：匹配任意数字，等价于 [0-9]。
- \D：匹配任意非数字。
- \A：匹配字符串开始。
- \Z：匹配字符串结束，如果是存在换行，只匹配到换行前的结束字符串。
- \z：匹配字符串结束。
- \G：匹配最后匹配完成的位置。
- \b：匹配一个单词边界，也就是指单词和空格间的位置。例如， 'er\b'：可以匹配"never" 中的 'er'，但不能匹配 "verb" 中的 'er'。
- \B：匹配非单词边界。'er\B' 能匹配 "verb" 中的 'er'，但不能匹配 "never" 中的 'er'。
- \n, \t, 等. 匹配一个换行符。匹配一个制表符。等
- \1...\9：匹配第n个分组的内容。
- \10：匹配第n个分组的内容，如果它经匹配。否则指的是八进制字符码的表达式。