[杂项](#杂项)、[列表](#列表)、[元组](#元组)、[字典](#字典)、[集合](#集合)、[字符串](#字符串)、[函数](#函数)、[类](#类)、[模块](#模块)、[异常](#异常)、[文件](#文件)、[目录（文件夹）：](#目录（文件夹）：)、[时间：](#时间：)

[虚拟环境](#虚拟环境)

[Decimal模块（精准的小数计算）](#Decimal模块（精准的小数计算）)

[sys模块](#sys模块)

[itertools模块](#itertools模块)

[正则表达式.docx](file:///F:\1学习资料备份\Python\正则表达式.docx)

[GUI.docx](file:///F:\1学习资料备份\Python\笔记\GUI.docx)

[Database.docx](file:///F:\1学习资料备份\Python\笔记\Database.docx)

1. 杂项：
   1. 路径分隔符“\”在字符串中用转义反斜杠“\\”或斜杠“/”代替。或使用原始字符串
   2. . 当前所在目录
   3. .. 当前所在目录的父目录
   4. ./ 当前所在目录下的某个文件夹或文件
   5. ../ 当前所在目录的父目录下的某个文件夹或文件
   6. dir(对象) 查看对象可用属性
   7. is与==：is比较id，==比较值
2. 列表：
   1. listname = list(range(10, 20, 2))
   2. index, item = **enumerate**(listname) 返回列表中元素索引值和元素值。与for循环连用。
   3. list.**append**() list.**extend**() list.**insert**(index, item) 在下标为index元素左边插入item。
   4. list.**remove**(item) list.**count**(item) list.**index**(item) 返回下标，不存在时抛出异常。
   5. **sum**(list) list.**sort**([key=None], [reverse=False]) 从小到大排序，直接改变列表，无返回
   6. **sorted**(list,[],[]) 返回改变后列表的复制
      1. [key = lambda x : x.data] key会在每次比较前调用，用于对象列表按某参数排序，key的值是一个输入为列表元素，输出为某值的函数
   7. list = [2\*I for I in range(10) if I%2==1] 对0到9取值，如果为奇数，输出它的两倍。
3. 元组：
   1. 创建时可加括号可不加。
   2. 元组内元素不可修改，但可以将整个元组重新赋值。元组也能用+相连接。
   3. 元组通过推导式会生成一个生成器对象。要用tuple()将生成器对象再转成元组对象。
   4. zip([迭代器, ..,]) 将迭代器依次打包成元组，返回元组列表
4. 字典：
   1. 创建：{ key1 : value1, key2 : value2…} dict(key1 = value1, key2 = value2…) dict(**zip**(list1, list2))
   2. dic = {}.fromkeys(‘陈李王赵钱孙’) 取单个字符为键，创建值为空的字典
   3. 遍历：for item in dictionary.items(): item为（键，值）的元组。
   4. del dictionary[key]
   5. dict.setdefault(key, default=None) 获取key对应的值，若key不在字典中，添加key并设值为default
   6. dict .get(key[, default=’xx’]) 获取key对应的值，若键不存在则返回xx
   7. 有序字典：
      1. from collections import OrderedDict
      2. dict = OrderedDict({......}) 遍历时输出会按插入顺序
5. 集合：
   1. 集合无序，每次输出的顺序都不确定。集合会合并相同元素。
   2. setname = {element1, element2…} setname = set(iteration) 将可迭代对象转为集合。
   3. setname.**add**(element) setname.**remove**() setname.**pop**() setname.**clear**()
   4. 交集运算 & 并集运算 | 差集运算 –
6. 字符串：
   1. length = **len**(str1.encode()) 计算字符串所占字节数。UTF-8中文占3字节，GBK/GB2312中文占2字节。
   2. strlist = str.**split**([sep[, maxsplit = -1]]) 将字符串按sep分隔符分隔后返回列表。
      1. 若分隔符之间无元素，会出现‘’。若指定分隔符为空格，而存在多个空格时，会出现’’。
      2. 若不指定分隔符，默认按空格和空白字符(\n \r \t)分隔，且存在多个空格时，不出现上述情况
   3. strnew = string.**join**(iterable) 把迭代对象以string为分隔符连接为字符串。
   4. str.count(sub[, start[, end]]) 返回str中sub出现的次数。
   5. str.find(sub[, start[, end]]) 返回str中sub首次出现的索引。不存在返回-1。
   6. str.rfind(…) 从右边开始的find。
   7. str.index(…) 返回str中sub首次出现的索引。不存在抛出异常。
   8. str.startwith(…) str.endwith(…) 返回True或False。
   9. str.lower() str.upper()
   10. str.strip([chars]) 去除左右的char内所有字符 str.lstrip(…) str.rstrip(…)
       1. 若不指定，默认空格、tab、回车、换行。
   11. ‘%09d’%(num) 格式化输出字符串
   12. str.format(args) 格式化输出字符串，推荐。
       1. {[index][:[[fill] align][sign][#][width][ , ][.precision][type]]}
          1. index : 索引值
          2. fill : 空白处填充字符
          3. align : < 左对齐 > 右对齐 = 数字右对齐 ^ 数字居中
          4. sign : + 数字显示符号 - 负数加负号 空格 正数加空格，负数加负号
          5. # : 加上后二、八、十六进制数显示0b、0o、0x前缀
          6. width : 所占宽度
          7. , :每隔三位数加，
          8. .precision : 保留小数位数
          9. type : 指定类型。s：字符 d：整型 f：浮点数 详见P143
   13. bytes类型：二进制数据。表现为带b前缀的字符串。b’mr’
   14. str.encode([encoding = “utf-8”][,errors = “strict”]) 将str按utf-8编码为bytes。
       1. 若不写异常项，可忽略encoding=。
   15. str.decode(…)
   16. str.center(int) 将字符串用空格填充至int长度，字符串居中
   17. str.ljust(int) str.rjust(int) 将字符串用空格填充至int长度，字符串居左/右
   18. print（‘\ 033 [显示方式;字体色;背景色m 内容 [\ 033 [0m’）指定输出颜色格式
       1. 显示方式：可不写。1：高亮显示 4：下划线 5：闪烁 7：反白显示 8：不可见
       2. 字体色30-37背景色40-47：黑、红、绿、黄、蓝、紫红、青蓝、白
   19. str .zfill(int) 在数字左侧加0直至int长度
7. 函数：
   1. 可变对象（指针传递）：列表、字典
   2. 不可变对象（值传递）：字符串、整数、浮点数、元组……
   3. 默认参数指向可变对象时，只在第一次调用赋值，之后会改变可变对象。
   4. 可变参数：
      1. \*parameter : 接收任意多个实际参数，以元组形式赋值给parameter。
         1. fun(“张三”，“李四”) 或
         2. fun(\*[“张三”，“李四”])
      2. \*\*parameter : 接收任意多个赋值形式的实际参数，以字典形式赋值给parameter。
         1. fun(张三 = 5，李四 = 8) 或
         2. fun(\*\*{‘张三’：5，’李四’：8})
   5. 函数体内可以访问函数体外的变量（全局变量）。
   6. 函数体内可用global声明全局变量
   7. lambda [参数1，参数2。。。] : 表达式 返回表达式的值
      1. 将函数赋给变量：add = lambda x, y: x+y add(5, 9)
      2. 屏蔽函数: time.sleep = lambda x: None
      3. 作为返回值: return lambda x, y: x+y
   8. 闭包：将函数名作为返回值
      1. def nth\_power(exponent):
         1. def exponent\_of(base):
            1. return base \*\* exponent
         2. return exponent\_of
      2. square = nth\_power(2)
      3. print(square(3))
      4. 输出9
   9. 以函数为参数的函数：
      1. max(可迭代对象，key=函数)
      2. sorted([1, 2, 3, 4, 5], key=lambda x: abs(2 - x))
      3. list.sort(可迭代对象，key=函数)
      4. filter(lambda x: x % 3 == 0, [1, 2, 3]) 筛选能与3整除的项
      5. map(lambda x: x \* 2, [1, 2, 3]) 使所有项×2
8. 类：
   1. 类属性、实例属性
   2. 私有方法和属性：
      1. 属性或方法前加双下划线\_ \_
      2. 只允许类内部进行访问，若要在外部进行访问，要用 \_类名\_ \_变量名或方法名 的形式
   3. @property：
      1. 在方法定义前一行出现，将方法转化为属性，调用时不需括号，并且无法被重新赋值。
      2. 可实现私有变量的只读特性：
         1. @property
         2. def age(self):
            1. return self.\_ \_ age
      3. @函数名.setter
      4. def 函数名(self, value)
         1. 进行 函数名 = xx 赋值时，将xx作为value参数传给函数，并执行。
   4. 继承、方法重写
   5. 子类通过super().\_\_init\_\_()调用父类的初始化方法
9. 模块：
   1. from 模块名 import \* 导入全部定义
   2. print(dir()) 打印导入的全部定义
   3. 包（含有模块文件的文件夹）：
      1. 包内必须有“\_ \_init\_ \_.py”文件
      2. 导入：
         1. import 包.模块
            1. 使用：包.模块.定义名
         2. from 包 import 模块
            1. 使用：模块.定义名
         3. from 包.模块 import 定义名
            1. 使用：定义名
10. 异常：
    1. int(小数)抛出ValueError
       1. try:
       2. except[EcxeptionName [as alias]]:
          * 1. 默认捕获全部异常，可指定别名，可将多个异常用括号括起。
            2. 若要输出异常内容，需用alias变量输出。
       3. else:
          * 1. 无捕获异常时执行
       4. finally:
          * 1. 任何情况执行
    2. raise [ ExceptionName [ ( reason) ] ]
       1. 抛出ExceptionName类型的异常，内容为reason
    3. assert expression [ , reason ]
       1. 若expression表达式不满足，抛出AssertionError，内容为reason
11. 文件：
    1. file = open( filename[ , mode[ , buffering] ][, encoding = ‘GBK’] )
       1. mode: 打开模式 r rb r+ rb+ w wb w+ wb+ a ab a+ ab+
       2. buffering: 缓冲模式 0表不缓存 1表缓存 大于1表缓冲区大小
    2. with expression as target: #将expression的结果保存到target，若为打开文件，with执行完自动关闭
       1. with-body
    3. file.write(string) #当打开模式为w或a时，写入string。
       1. file.writelines(string) 写入后不添加换行符
    4. file.close() 将缓存区中的内容写入文件并关闭文件
    5. file.flush() 将缓存区中的内容写入文件
    6. file.read( [ size ] ) 当打开模式为r或r+时，从指针开始读取size个字符，默认全部读取。
    7. file.seek( offset[ , whence] ) 移动指针
       1. offset: 移动offset个**字节**，GBK：2字节/中文字符。UTF-8：3字节/中文字符。
       2. whence: 0表从文件头开始，1表从当前位置开始，2表从文件尾开始。默认为0，不使用b模式只能为0
    8. file.readline() 在r、r+模式下，读取一行
    9. file.readlines() 在r、r+模式下，读取全部，每行作为元素，返回一个字符串列表。
12. 目录（文件夹）：
    1. os模块：
       1. os.name 返回操作系统类型 nt : windows posix : Linux、Unix、Mac OS
       2. os.linesep 返回当前操作系统的换行符
       3. os.sep 返回当前操作系统的路径分隔符
       4. os.getcwd() 返回当前目录
       5. os.listdir( path ) 返回路径内文件和文件夹名字的列表
       6. os.mkdir( path, mode = 0o777 )
          1. 创建一级目录path，要求path上一级目录存在，否则抛出FileNotFoundError。
          2. mode在非UNIX系统可忽略。
          3. 若path路径已存在，抛出FileExistsError。
       7. os.makedirs( name, mode = 0o777 ) 采用递归形式创建多级目录name，已存在则同上。
       8. os.rmdir( path ) 删除**空**目录，不为空抛出OSError，不存在抛出FileNotFoundError。
          1. shutil.rmtree( path ) 删除目录，可以不为空
       9. os.walk( top[ , topdown][ , onerror][ , followlinks] ) 遍历目录及其子目录
          1. top : 要遍历的根目录
          2. topdown : True : 自上而下遍历（先根目录）。False : 自下而上遍历（先最后的子目录）。默认True。
          3. onerror : 错误处理方式
          4. followlinks : True：会转向快捷方式所在目录。False：不会。默认False
          5. 返回：（当前遍历路径的字符串，当前路径所包含子目录的列表，当前路径所包含文件的列表）的元组------生成器对象
       10. os.remove( path ) 删除文件，不存在则抛出FileNotFoundError
       11. os.rename( src, dst ) 将src重命名为dst，两个都为路径格式。不存在同上。
           1. 重命名目录时，若目录不是最后一级，抛出FileNotFoundError异常。
       12. os.stat( path ) 获取文件基本信息，返回一个对象。
           1. 对象.st\_size 文件大小（Byte） 对象.st\_atime 最后访问时间
           2. 对象.st\_mtime 最后修改时间 对象.st\_ctime 最后状态变化时间（windows返回创建时间）
           3. 返回的时间为秒数
       13. os.startfile(path) 通过系统默认程序打开文件
    2. os.path模块：
       1. os.path.abspath(相对path) 根据相对路径返回绝对路径。
       2. os.path.join( path1[ , path2[ , ……] ] ) 拼接路径，返回路径字符串。
          1. os.path.join( r“E:\program”, r“demo\message.txt”)
          2. 若有多个绝对路径，只取最后出现的。
          3. 若无绝对路径，返回值为相对路径。
       3. os.path.exists( path ) 若路径存在，返回True，否则返回False
       4. os.path.splitext(完整文件名) 返回[文件名，后缀]，如[‘123’, ‘.png’]
13. 时间：
    1. time模块：
       1. time.strftime( format[ , t ] ) 以format格式转化struct\_time对象，默认输出当地时间。
          1. time.strftime(‘%Y-%m-%d %H:%M:%S’,time.localtime(sec))
       2. time.localtime( [ sec ] ) 将秒数转化为struct\_time对象，默认输出当地时间。
14. 虚拟环境：
    1. cd venv/Scripts
    2. activate
    3. cd ..
    4. cd ..
    5. main.py
15. Decimal模块（精准的小数计算）：
    1. from decimal import \*
    2. a = Decimal( ‘4.2’ )
    3. b = Decimal( 3 )
       1. 将整数或字符串转化为decimal对象。
    4. getcontext().prec = 4
       1. 保留4位有效数字
    5. Decimal(‘1.23456789’).quantize(Decimal(‘0.00’))
       1. 保留两位小数
    6. Decimal对象可以直接用加减乘除运算符
16. sys模块：
    1. sys.argv 获取当前正在执行的命令行参数的参数列表
    2. sys.path 获取指定模块路径的字符串集合
    3. sys.exit() 退出程序，参数非0时引发一个SystemExit异常
    4. sys.platform 获取当前系统平台
    5. sys.modules 加载模块字典，每次导入新模块时，自动记录该模块。相同模块第二次导入时Python自动从该--------------字典中查询，从而加快运行速度
    6. sys.getdefaultencoding() 获取当前系统编码方式
    7. sys.stdout.write(‘string’) 输出字符串，不自动换行
    8. sys.stdout.flush() 缓冲区数据全部输出
17. itertools模块：
    1. cycle（可迭代对象） 返回迭代器，迭代至最后一位时自动返回第一位
18. math模块：
    1. .ceil(float) 向上取整，返回整数