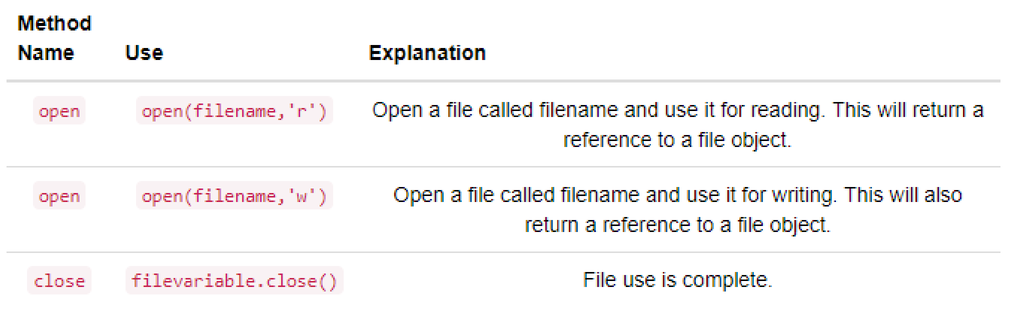
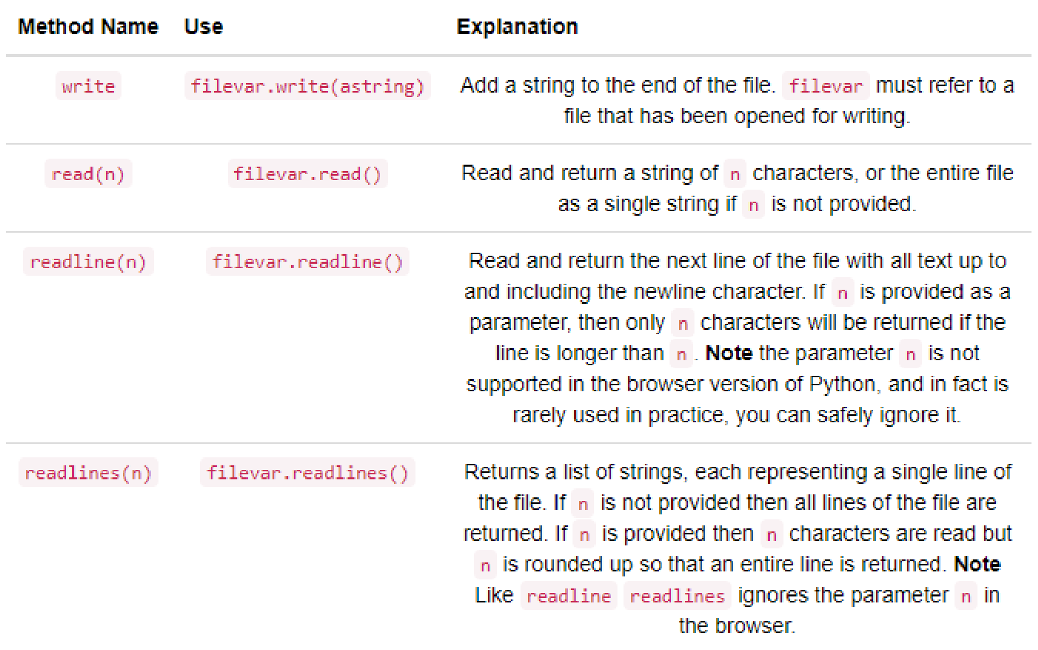
2.1 working with data files

- assume txt files
  - files filled with characters
  - 文件命名不要有空格/\不要有逗号
  - 假设有个文件名字叫olympics.txt
    - open close 开关文件
      - 

Method Name	Use	Explanation
`open`	`open(filename,'r')`	Open a file called filename and use it for reading. This will return a reference to a file object.
`open`	`open(filename,'w')`	Open a file called filename and use it for writing. This will also return a reference to a file object.
`close`	`filevariable.close()`	File use is complete.

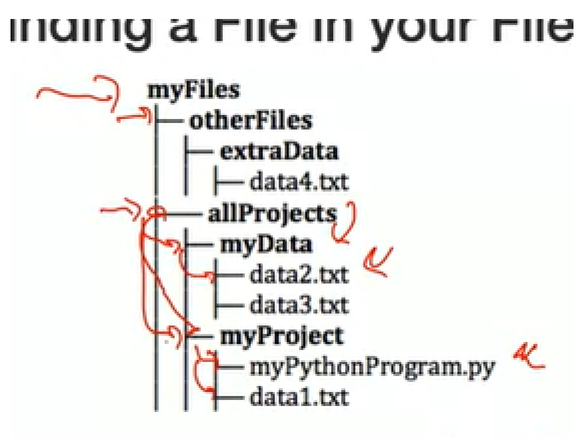
      - fileref = open("olympics.txt", "r")
      - 开的括号里第一个string是文件名和文件形式 逗号 第二个string里是操作 r是read w是write 一旦write了某文件之前的内容都会消失 然后从0开始写
      - fileref.close()
    - 还有另一种open的写法
      - with open("mydata.txt","r") as md:
      - 这句话跟md=open("mydata.txt","r") 一样
      - for num in range(10):
      -     md.write(str(num))
      -     md.write("/n")
    - 有一个很重要的点！ readlines 之后 因为你已经use up 整个file了 所以如果再 call up again就会得到一个空的list
    - read() 看全部内容
      - contents=fileref.read()   return a string and assign it to name contents. 因为一个文件可能有很多数据然后很大 所以.read就比没用了 因为.read就是要看所有的东西



Method Name	Use	Explanation
`write`	`filevar.write(astring)`	Add a string to the end of the file. `filevar` must refer to a file that has been opened for writing.
`read(n)`	`filevar.read()`	Read and return a string of `n` characters, or the entire file as a single string if `n` is not provided.
`readline(n)`	`filevar.readline()`	Read and return the next line of the file with all text up to and including the newline character. If `n` is provided as a parameter, then only `n` characters will be returned if the line is longer than `n`. **Note** the parameter `n` is not supported in the browser version of Python, and in fact is rarely used in practice, you can safely ignore it.
`readlines(n)`	`filevar.readlines()`	Returns a list of strings, each representing a single line of the file. If `n` is not provided then all lines of the file are returned. If `n` is provided then `n` characters are read but `n` is rounded up so that an entire line is returned. **Note** Like `readline` `readlines` ignores the parameter `n` in the browser.

- 如果要print里面的content的话
  - 比如：print(contents[:100]) print前100个characters
- .readline() 会print out list形式[] 然后用\n去隔开每个string 就是new line character
  - fileref=open("olympics.txt", "r")
  - lines=fileref.readlines()
  - print(lines[:4])
  - fileref.close()
  - 输出是一大坨东西然后用\n去隔开就比较丑 如果想要更好看一点的话 可以在中间再加
  - for lin in lines[:4]
  -     print(lin)
  - 就会没有了[]的符号 因为实际上在iterate strings而不是list了 输出一行然后空再一行 因为每一个string好像是会自动告诉你要下一行开始 然后print又默认开始新的一行
  - 如果不想要的话就改成print(lin.strip())就是之前讲过的删去空格的东西 只留有用的了 最后就是紧密的4行
  - file格式的文件不能[:4]take slice 但是可以整个看如果只想对前四行做手脚的话就用.readlines而不是直接iterate整个fileref
  - 例子
    - fileref=open("emotion_words2.txt","r")
    - contents=fileref.read()
    - first_forty=contents[:40]
    - fileref.close()
- 选择.read or .readlines?

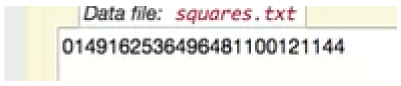
        - 如果想要take slice 用.readlines 或者只想要知道有多少lines in file 一般会搭配使用for loop
            - lines=fileref.readlines()
            - print(len(lines))
        - 如果想知道总共有多少个character在file中
            - contents=fileref.read()
            - print(len(contents))
- fingding a file in your file system
    - 第一个方法就是直接打开
    - 第二种相对path
        - 比如已经打开了一个py文件 然后要打开一个txt文件 先找到py文件的上家allProjects在下去myData里找到data2.txt
        - open("../myData/data2.txt","r")



```
inding a File in your File
myFiles
├─otherFiles
│ ├─extraData
│ │ ├─data4.txt
├─allProjects
│ ├─myData
│ │ ├─data2.txt
│ │ ├─data3.txt
│ ├─myProject
│ │ ├─myPythonProgram.py
│ │ ├─data1.txt
```

        - ..两个点就是返回上家的意思 ../myData就是path
    - 第三种绝对path
        - 在电脑里找文件
        - /user/presnick/myfiles/allProjects/myData/data2.txt
        - 一般不会用这个因为在别的电脑上可能不一样
    - 区分绝对和相对就看最前面有/的就是绝对 无/的是相对
- writing text files
    - 例子
        - for number in range(13):
        -    square=number*number
        -    print(square)
        - 但是想要这个不止在output窗口 而是store在某个file里面 改写
        - 在最前面加上 file_obj=open("square.txt","w")
        - 在最后面加上file_obj.close()

        - 把print那句话改写成 file_obj.write(str(square))
        - 出来的结果是这样的 因为print会默认进行到下一行 但是write进文件里并没有这样的性质

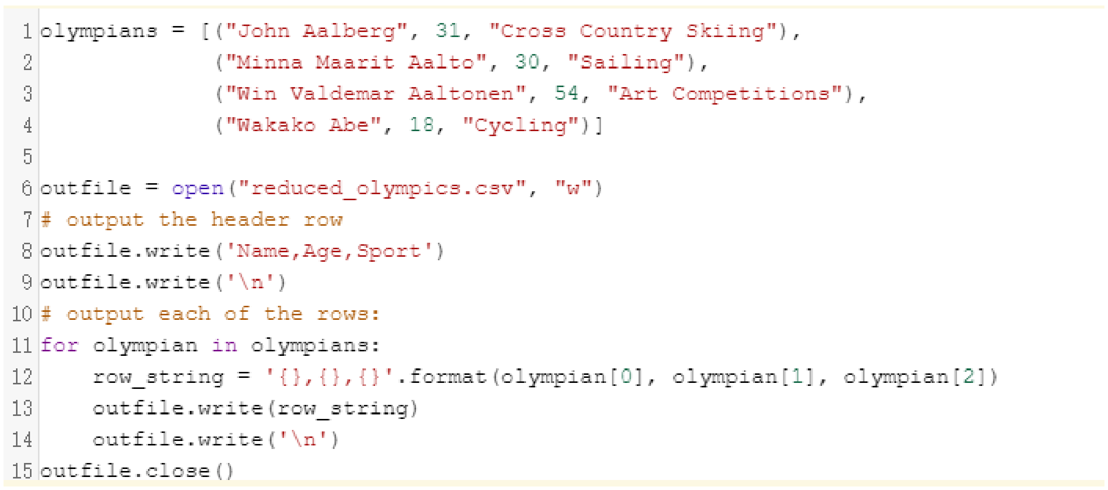


Data file: squares.txt

```
0149162536496481100121144
```

        - 如果想要的话 再写一句 file_obj.write("\n")
        - 或者直接改写成 file_obj.write(str(square)+“\n")
    - recipe
        - #1. Open the file using with and open.
        - #2. Use .readlines() to get a list of the lines of text in the file.
        - #3. Use a for loop to iterate through the strings in the list, each being one line from the file. On each iteration, process that line of text
        - #4. When you are done extracting data from the file, continue writing your code outside of the indentation. Using with will automatically close the file once the program exits the with block.
- csv format
    - every line of the file will have the same structure
    - 用逗号隔开
    - read a csv file
        - 第一行就是header 标题
        - header=lines[0]
        - field_names=header.strip(",")
        - print(field_names)
        - for row in lines[1:]:
        -     vals=row.strip().spit(",")   如果这里只是split()的话 就只能get到一个list包含所有word 但不是分开的word 是word当作一个整体
        -     if vals[5] !="NA":
        -         print("{};{};{}".format(
        -                     vals[0],
        -                     vals[4],
        -                     vals[5]))
    - write data to a csv file
        - 两种把东西带入的格式
            - for olympian in olympians:
            -     row_string = '{},{},{}'.format(olympian[0], olympian[1], olympian[2])
            -      outfile.write(row_string)

- outfile.write('\n')
- 或者用 row_string=",".join(olympian[0], olympian[1], olympian[2])但是join只认识2个objects 所以要给后面的东西再创一个list row_string=",".join([olympian[0], olympian[1], olympian[2]])
- 如果一堆全是string的话 用join function直接join（olympian）即可
-



```python
olympians = [("John Aalberg", 31, "Cross Country Skiing"),
             ("Minna Maarit Aalto", 30, "Sailing"),
             ("Win Valdemar Aaltonen", 54, "Art Competitions"),
             ("Wakako Abe", 18, "Cycling")]

outfile = open("reduced_olympics.csv", "w")
# output the header row
outfile.write('Name,Age,Sport')
outfile.write('\n')
# output each of the rows:
for olympian in olympians:
    row_string = '{},{},{}'.format(olympian[0], olympian[1], olympian[2])
    outfile.write(row_string)
    outfile.write('\n')
outfile.close()
```

  - 写入标题
  - 因为write method 不会create new line 所以要写入一个\n