

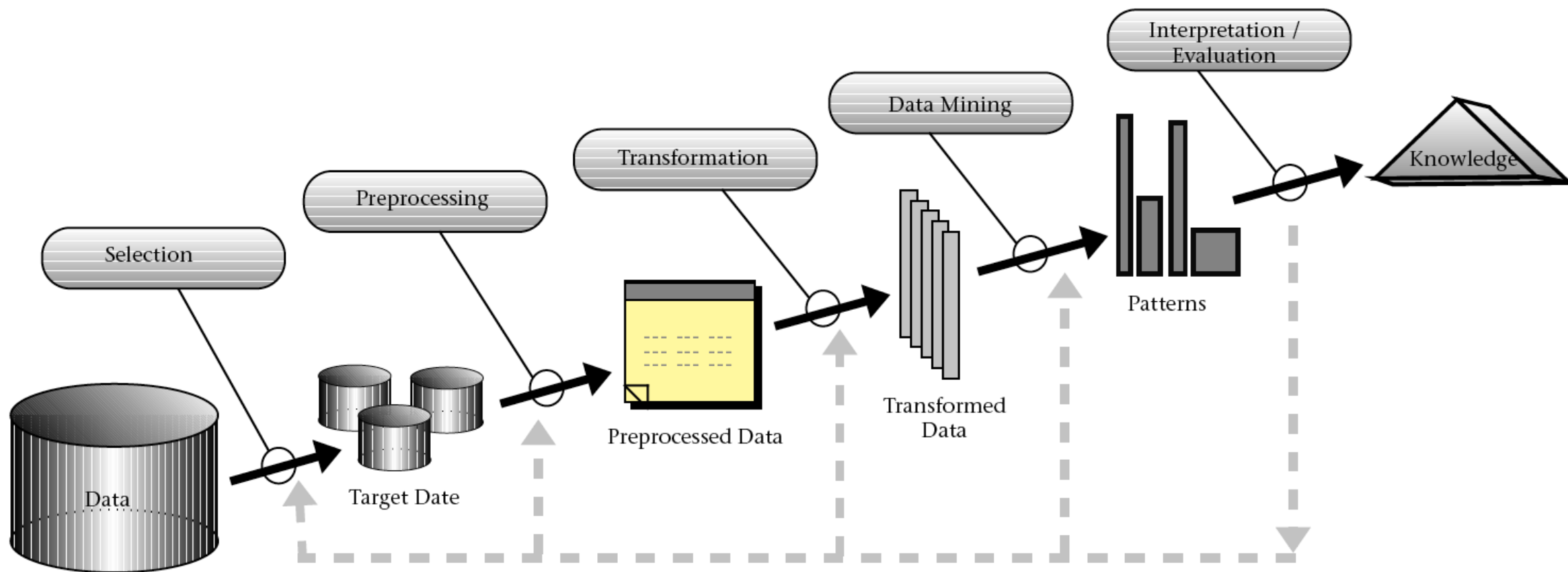
人工智能程序设计

M3 人工智能基础方法

1 数据获取

张 莉



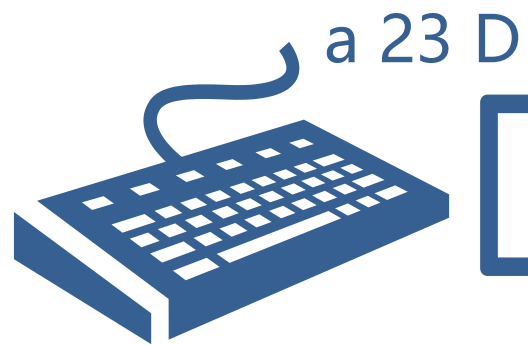




人工智能程序设计

基于内建模块的文件存取

程序中的数据



or



文件基本概念

- 文件：存储在某种介质上的信息集合
- 存储：外部介质
- 识别：文件名
- 分类
 - 存取方式：顺序存取，随机存取
 - 文件内容表示方式：二进制文件，文本文件



二进制文件与文本文件

12345的内存存储形式

00110000	00111001
----------	----------

↓ 转换成ASCII编码形式

00110001	00110010	00110011	00110100	00110101
----------	----------	----------	----------	----------

↓ 以ASCII编码形式写入fp

00110001	00110010	00110011	00110100	00110101
----------	----------	----------	----------	----------

fp 对应的文件

12345的内存存储形式

00110000	00111001
----------	----------

↓ 不进行转换直接写入fp

00110000	00111001
----------	----------

fp 对应的文件

二进制文件与文本文件

- **文本形式输出时**

- 一个字节与一个字符——对应
- 便于对字符进行逐个处理，也便于输出字符；
- 占存储空间较多；
- 要花费转换时间。



- **用二进制形式输出时**

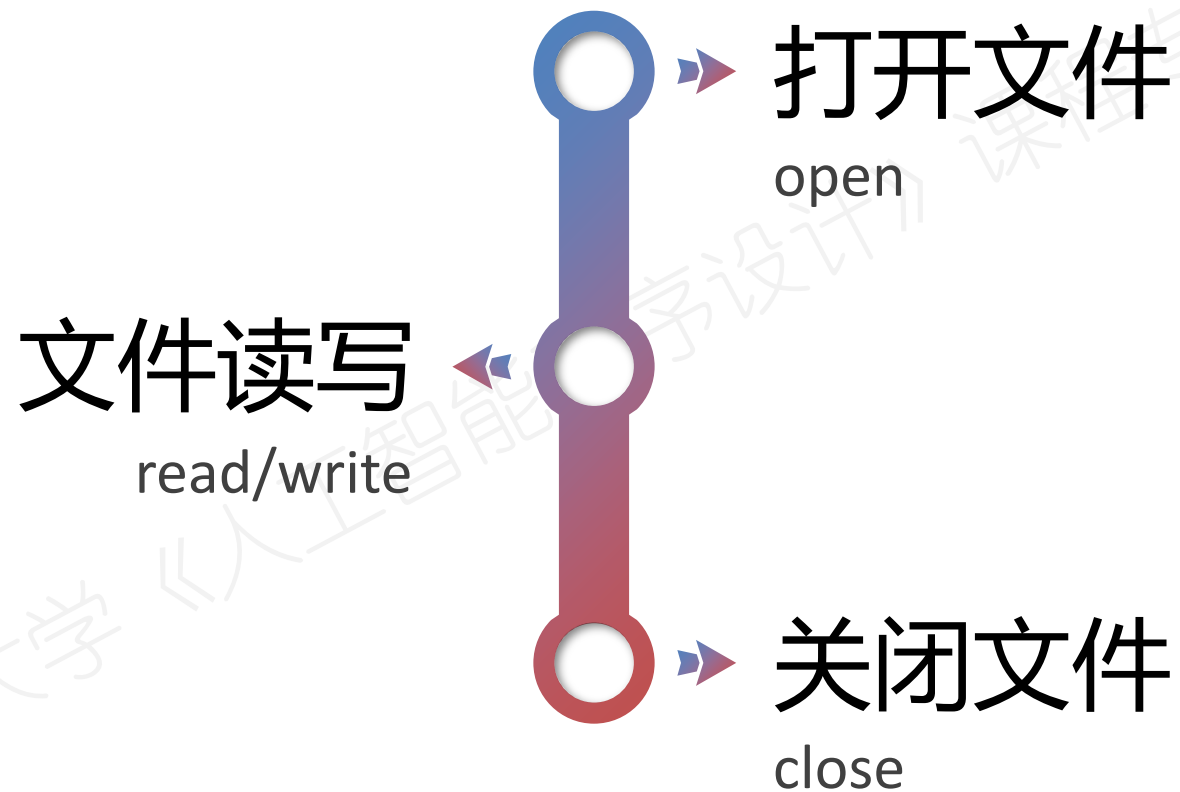
- 可节省外存空间和转换时间
- 一个字节并不对应一个字符，不能直接输出字符形式。
- 可读性差，常用于保存中间结果数据和运行程序。

二进制文件与文本文件

- Python中可以处理二进制文件以及文本文件，对二进制文件的操作可以选择是否使用缓冲区
- 缓冲区是内存中的区域，当程序中需要进行频繁的文件读写操作时，使用缓冲区可以减少I/O操作从而提高效率，也方便管理
- 文本文件均使用缓冲区处理



文件的使用过程



文件的打开

Source

```
>>> f1 = open('d:\\infile.txt')
```

```
>>> f2 = open(r'd:\\infile.txt')
```

```
>>> f3 = open('d:/outfile.txt', 'w')
```

```
>>> f4 = open('frecord.csv', 'ab', 0)
```

open()函数返回一个文件 (file) 对象

file_obj = open(filename, mode='r', buffering=-1)

- mode为可选参数，默认值为r
- buffering也为可选参数，默认值为-1（0代表不缓冲，1或大于1的值表示缓冲一行或指定缓冲区大小）
- 其他常用参数：encoding（指定编码字符集）

open()函数-mode

Mode	Function
r	以读模式打开，文件必须存在
w	以写模式打开，若文件不存在则新建文件，否则清空原内容
x	以写模式打开，若文件已经存在则失败
a	以追加模式打开，若文件存在则向结尾追加内容，否则新建文件
r+	以读写模式打开
w+	以读写模式打开（清空原内容）
a+	以读和追加模式打开
rb	以二进制读模式打开
wb	以二进制写模式打开（参见w）
ab	以二进制追加模式打开（参见a）
rb+	以二进制读写模式打开（参见r+）
wb+	以二进制读写模式打开（参见w+）
ab+	以二进制读写模式打开（参见a+）

关闭文件

- **fp.close()**

- fp为文件对象

- 切断文件对象与外存储器中文
件之间的联系



```
>>> fp = open(r'd:\nfile.txt', 'r')
>>> type(fp)
<class '_io.TextIOWrapper'>
>>> fp.name
'd:\\nfile.txt'
>>> fp.mode
'r'
>>> fp.closed
False
>>> fp.close()
>>> fp.closed
True
```

文件操作

try:

with open(r'd:\自己的文件目录\test.txt') as fp:

... # 各种文件处理

except IOError as err:

print(err)

文件的基本操作

返回值和基本操作

- `open()`函数返回一个文件 (file) 对象
- 文件对象可迭代 (for line in f)
- 有许多读写相关的方法/函数
 - `f.read()`, `f.write()`, `f.readline()`, `f.readlines()`, `f.writelines()`
 - `f.seek()`

读文件方法

- **s = fp.read(size)**
 - 从文件当前位置读取size字节数据，若size为负数或空，则读取到文件结束
 - 返回一个字符串（文本文件）或字节流（二进制文件）
- **s = fp.readline(size= -1)**
 - 从文件当前位置读取本行内size字节数据，若size为默认值或大小超过当前位置到行尾字符长度，则读取到本行结束（包含换行符）
 - 返回读取到的字符串内容
- **lines = fp.readlines(hint= -1)**
 - 从文件当前读写位置开始读取需要的字节数，至少为一行；若hint为默认值或负数，则读取从当前位置到文件末尾的所有行（包含换行符）
 - 返回从文件中读出的行组成的列表

写文件方法

- **fp.write(*s*)**
 - 向文件中写入数据（字符串或字节流）
 - 返回写入的字符数或字节数
- **fp.writelines(*lines*)**
 - 向文件中写入列表数据，多用于文本文件

在 π (前10000位) 中寻找自己的生日

```
pi.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

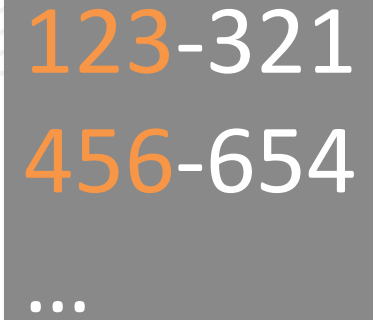
3.14159265358979323846264338327950288419716939937510582097494459230781640628620999682803482534211706798214808651328230664709384460955058223172535940812948111548
02841027019385211055596446229489549303819644228810975665933446128475648233786783163271201909145648566923460348610454326642812339360726024914127372406060606177585
174881520920962829254091715364367892590360011330530548820466521384146951941516094330572703657595919530921861173819326117931051185480744623799627495673518857527
2489122793818301194912983367336244065643086021394946395224737190702137086094370277053921717629317675238467481846766940513200056812714526356082778577134275778960
9173637178721468440901224953430146549585371050792279689258923542019956112129021960864034418159813629774771309960518707211349999998372978049951059731732816096318
595024459455346908302642522308253446850352619311881710100031378387238658753320383142061717766914730339825490429755468731159562863823537875937519577818577805
321712268066130019278766111959092164201989380952572010654858632788659361533818279682303019520353015296899577362259941389124972177528347913151557485724245415069
5950829533116861727855890750983817546374649393192550640092770167113909084882401285836160356370766010471018194295559619894676783744944825379774726847104047534
6462080466842590694912933136770289891521047521620569660240580381501935112533824300355876402474964732639141992726042699227967823547816360093417216412199245863150
3028618297455570674983850549458858692699560992721079750930295532116534498720275596023648006549911988183479775356636980742654252786255181841757467289097777279380
008167060016145249192173217477235014414197358548161361157352552133475741849468438523329073941433345477624168625189935694855620992192218427552524526887671
7904946016534680498862723279178085784338327967976681454100933863786360950680064225125205117392984896084128486269436042419662850222106611863607442786202391949
4504712371378696095636437191728746776465757396241389086583264599581339047802759009946576407895126948398352595709825822620522489407726719478268482601476999090264
01363944374535050682034962524517493996514314298091906592509372216964615157098583874105978859597729754598930161753928468138268683868942774155991855925245953959431
04997252468084598727364469584865383673622266099124608051243884390451244136549762780797715691435997700129616089441694865558484063534220722258284886481584560285
0601684273945226746767889525213852254995466672782398645659611635488623057745649803559363456817432411251507606947945109659609402522887971089314566913686722874894
056010150330861792868092087476091782493858900971490967598526136554978189312978482168299694872268084857564014270477555132379641451523746234364642858444795265867
8210511413547357395231134271961021359895362314429524849371871104576540339027993440374200731057539062198387447808478489833214457138887519435064302154531910484
810053706146806749192781911979399520614196634287544406437451237181921799983910159195618146751426912397489409071864942319615679452080951465502232160388193014209
3762137855956638937787083039069792077346722182562599661501421503068038447734549202605414665925201497442850732518666002132434088190710486331734649651453905796268
561005508106658796998163574736384052571459102897064140110971206280439039759515677157700420378699360072305587631763594218731251471205329281918261861258673215791
984148485291644706095752706957220917567116722910981690915280173506712748583222871835209353965725121083579151369882091444210067510334671103141267113699086585163
96315019701651116851714376576183515650884909989599823873452833163550764791833893226185489631232330898570642046752990709154811654989946137180270981994309
92488957571282890592323260972997120844335732654893823911932597408367305836041428138830320382490375898524374417029132765818093734440307074692112019130203033083
1976211011004492932151608424485963769838952286847831235526821314495768572624334189303968642624341077322697802807318915441101046823252716201052652271116603
966557309254711055783763466820653109896526918620564769312570586356620185581007293606598764861179104534885034611365768675324944166803962579787178556084552965
4126654085306143444318586769751456614068007002378776591344017127494704205622305389945613140711270004078547332699390814546646458807972708266830634328587856983052
338093306575740679545716377525420211495576158149040250126228594130216471550979292309907965473761255176567513575178296646477917450112996148903046399471329621073
4043751893735961458901938971311139042978235647303203198691514028708083900490109412174223197947677428291422845454033213718530614223813758504306332175182978662
23717215916077166925474873898654549494301146540628433663937900397692656721463853067360965712091807638327166416274888800786925602902284721040317211860820419000422
9661711963779213375751149595015606496318629473624523081770367515906735023507283540567040386743513622247715891504953098440893309634087076932593978054193
414473774184263129860809988687413260472156951623965864573021631598193195167358129741677294786724229246543668009806769282382068996400482435403701416314965897
94092432378969070697794223625082216889573837986230015937764716512289357860158816175782973523344604281512627203734314653197774160319906655418763979293344195215
4134189845444734567383162499341913181480927771038638773431720754565453220770921201903166096230490926360197598828161332316636528619326683360627356763035447
782803504807723354710585954870279081435624014517180624648267948127531813407833036254232783944975382437205835311477119926063813346776879695703908339130771409
707408591337464142822772634659470474587847787201927715280731767907707157213444730605700733492436931138350493163128404251219255179806941135280134710347816757
88518529092854520116583934196562134914341595625865855705526904965209858033850722426482939728584783163057777660688764462482465579260395352773480304802900587607
58251047470916439613626760449256274204206320856611906254543721313595845068724602901618766795240616342522571954291629919306453779914037340432888963995
87947572917464263574552540799914513571138941091939352519107602082520281879853188770584297259167781314969909091921169717372784768472686084900337702424916513005
005168323364350389517029893922345172501381286965011784408745116021228599371623130171144484640003890644954440023896074618740766868818385
102283345085048608250390213321971551843063545500766828294930413776527939751754613953984683396383047461199665385815384205685338621867252334028308712328278921
250771262943229563989898935821167456270102183564622013496715188190973038119800497340723961036854066431939509790190699639552453005450580685501956730229219139339
18568034490398205955100226353536192041994745538593810234395544959778379023742161727112364435439478221818528624085140066604150288856986705431547069657474585
503323233421073014269405165537906866273337998511562578432298827372319898757141595781119635833005940873068121602876496286744604774649159950549737425626901040937
81986835938146571458949236485798556145372347867330390486933436346537949864537208723057793174872337203760112302991159793982708943879683016395154133714248828307
22012690147546684765375164773794673200490751355278196536213239364061601363831590074202202030187277605772190055614842551879250345313984425229417562936106455
06390497500865627109535919464589751413103482276930624743536325691607815478181152843667957061108615331504452127473924544945423682886061340841486377670096120715124
914043027258607648236341433462351897576645216413767969031495019108575984423919862916421939949072362346468441173940326591840443780513338945257423995082965912285
08555821572503107125701266830240292952522011827676756220415420516184163484756516999811614100299607838690929160302884002691041407928862150784245167090870006992
821206604183718065355672552352675328612910424877618258297651579598470356222629344600341587229805349996502262917487788202734209222453398562647669149956284250391
```



```
# Filename: find_birth.py
with open('pi.txt') as fp:
    pi_file = fp.read()
if '0912' in pi_file:
    print('66666')
else:
    print('55555')
```

写入回文串

```
lst = []  
for line in open('data.txt'):  
    lineRev = line[::-1]  
    lst.append(line.strip()+'-'+lineRev.strip()+'\n')  
with open('data.txt', 'w') as fp:  
    fp.writelines(lst)
```



123-321
456-654
...

文件读写例子

将文件companies.txt 的字符串前加上序号1、2、3、...后写到另一个文件scompanies.txt中。



Filename: prog1.py

```
with open('companies.txt') as f:
    lines = f.readlines()
    for i in range(len(lines)):
        lines[i] = str(i+1) + ' ' + lines[i]
with open('scompanies.txt', 'w') as f:
    f.writelines(lines)
```

Output:

```
1 GOOGLE Inc.
2 Microsoft Corporation
3 Apple Inc.
4 Facebook, Inc.
```

文件读写例子改写

将文件companies.txt 的字符串前加上序号1、2、3、...
后写回文件中。



```
# Filename: prog2.py
with open('companies.txt', 'r+') as f:
    lines = f.readlines()
    for i in range(len(lines)):
        lines[i] = str(i+1) + ' ' + lines[i]

    f.writelines(lines)
```

Output:

```
1 GOOGLE Inc.
2 Microsoft Corporation
3 Apple Inc.
4 Facebook, Inc.
```

文件的定位-`seek()`方法

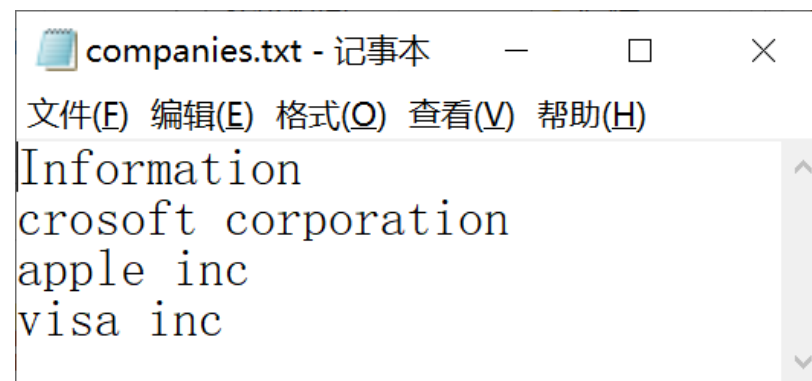
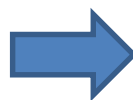
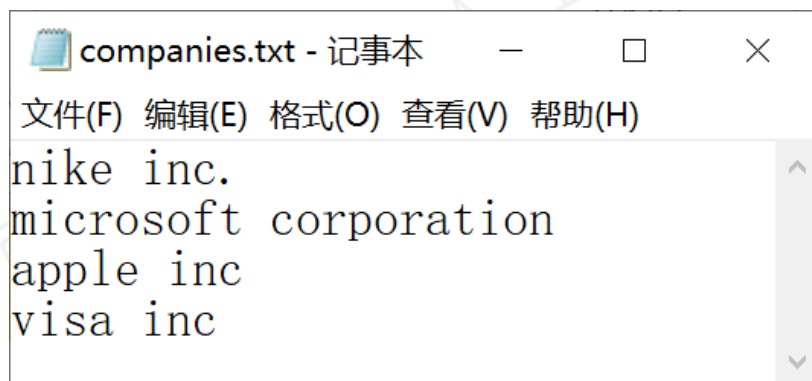
- **`fp.seek(offset, whence=0)`**
 - `fp`打开的文件必须允许随机访问
 - 在文件中移动文件指针，从`whence`（0表示文件头部，1表示当前位置，2表示文件尾部）偏移`offset`个字节
 - 返回当前的读写位置

文件中头部插入一个新行



```
with open('companies.txt', 'r+') as f:  
    lines = f.readlines()  
    f.seek(0)  
    f.write('Information\n')
```

覆盖模式



文件中头部插入一个新行



```
with open('companies.txt', 'r+') as f:
    lines = f.readlines()
    for i in range(0, len(lines)):
        lines[i] = str(i+1) + ' ' + lines[i]
    lines.insert(0, 'Information\n')
    f.seek(0)
    f.writelines(lines)
```

文件的定位-tell()方法

- **fp.tell()**

- 返回文件的当前读写位置



```
>>> fp = open('testseek.dat', 'rb+')
>>> fp.tell()
0
>>> fp.read(5)
b'Hello'
>>> fp.tell()
5
>>> fp.close()
```


读取某目录下的多个文件并进行统计

```
import os
def countLines(fname):
    with open(fname) as f:
        data = f.readlines()
        lens = len(data)
        print(fname + ' has ' + str(lens) + ' lines')
if __name__ == '__main__':
    # files = ['data1.txt', 'data2.txt', 'data3.txt', 'data4.txt']
    path = 'c:/test'
    files = os.listdir(path+'/kkk')
    for fname in files:
        countLines(path+'/kkk/'+fname)
```

```
if os.path.exists('output'):
    shutil.rmtree('output')
os.mkdir('output')
```

```
file_name = os.path.join(path, file)
```