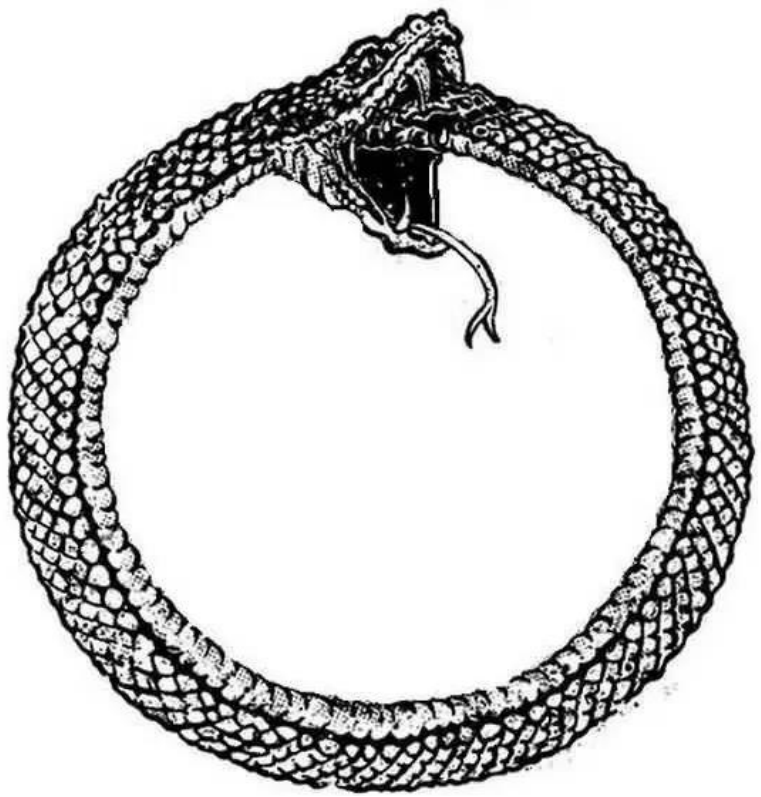
表扬：爬了三层的同学们

包凯宇、丁缊婕、黄家齐、兰轶、向雨馨、叶紫菁

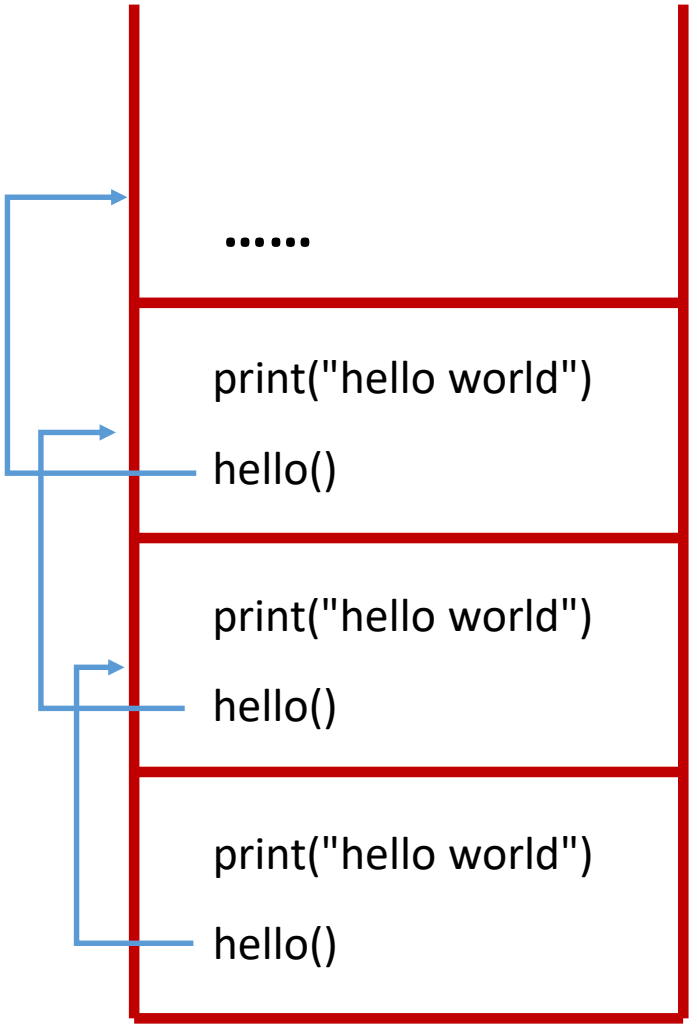
尹保利（用了递归，没有终止条件死循环了，能爬无穷层）

函数可以调用自己么，就像贪吃蛇那样？



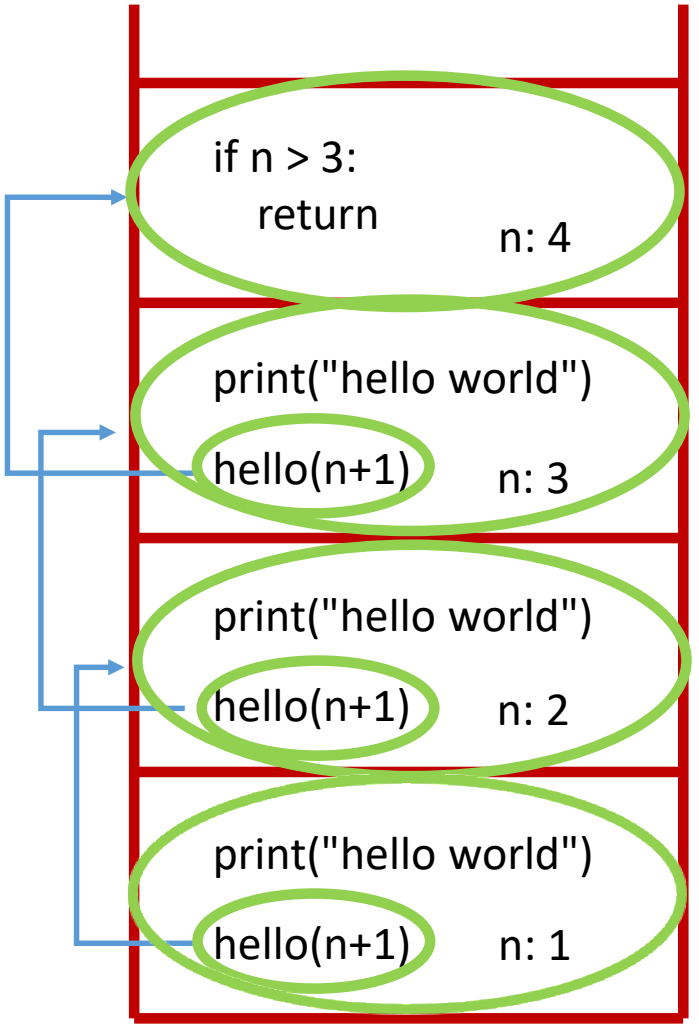
递归调用

```
def hello():
    print("hello world")
    hello()

hello()
```



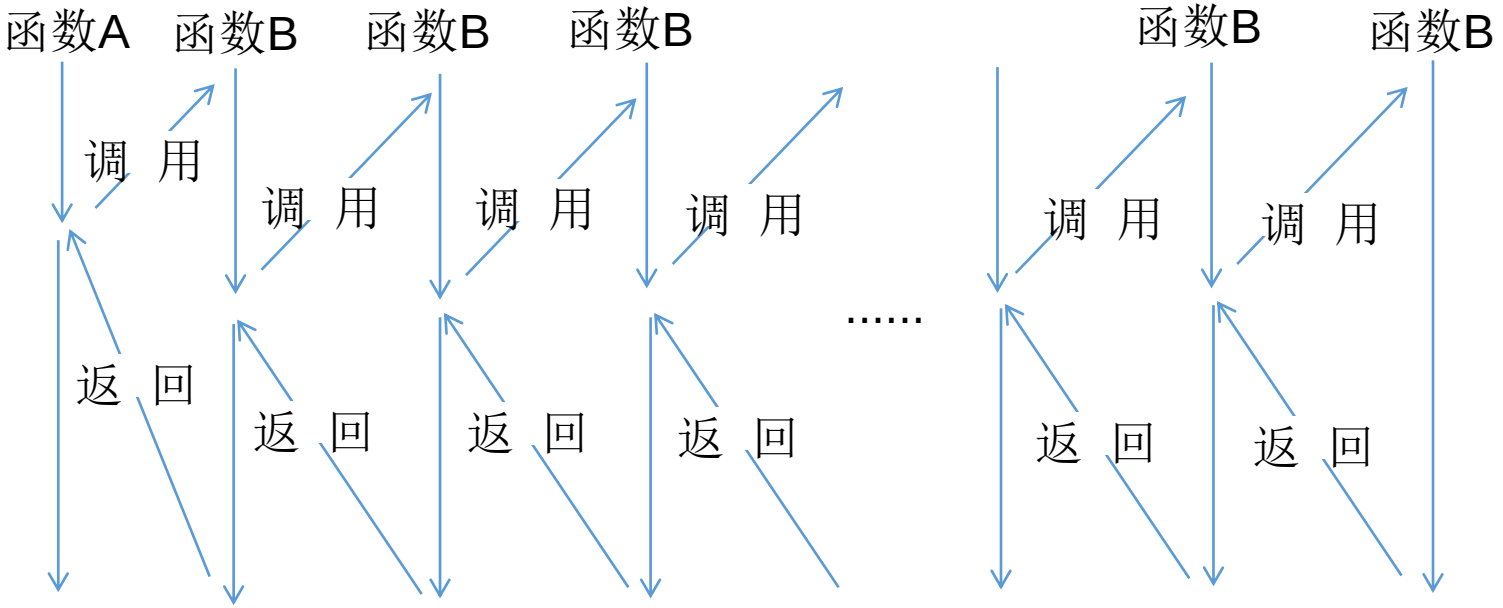
递归调用

```
def hello(n):
    if n > 3:
        return
    print("hello world")
    hello(n + 1)

hello(1)
```



5.1.3 函数递归调用

- 函数的递归调用是函数调用的一种特殊情况，函数调用自己，自己再调用自己，自己再调用自己，...，当某个条件得到满足的时候就不再调用了，然后再一层一层地返回直到该函数第一次调用的位置。



【例】有5个人，第5个人说他比第4个人大2岁，第4个人说他对第3个人大2岁，第3个人说他对第2个人大2岁，第2个人说他比第1个人大2岁，第1个人说他10岁。求第5个人多少岁。

通过分析，设计递归函数如下：

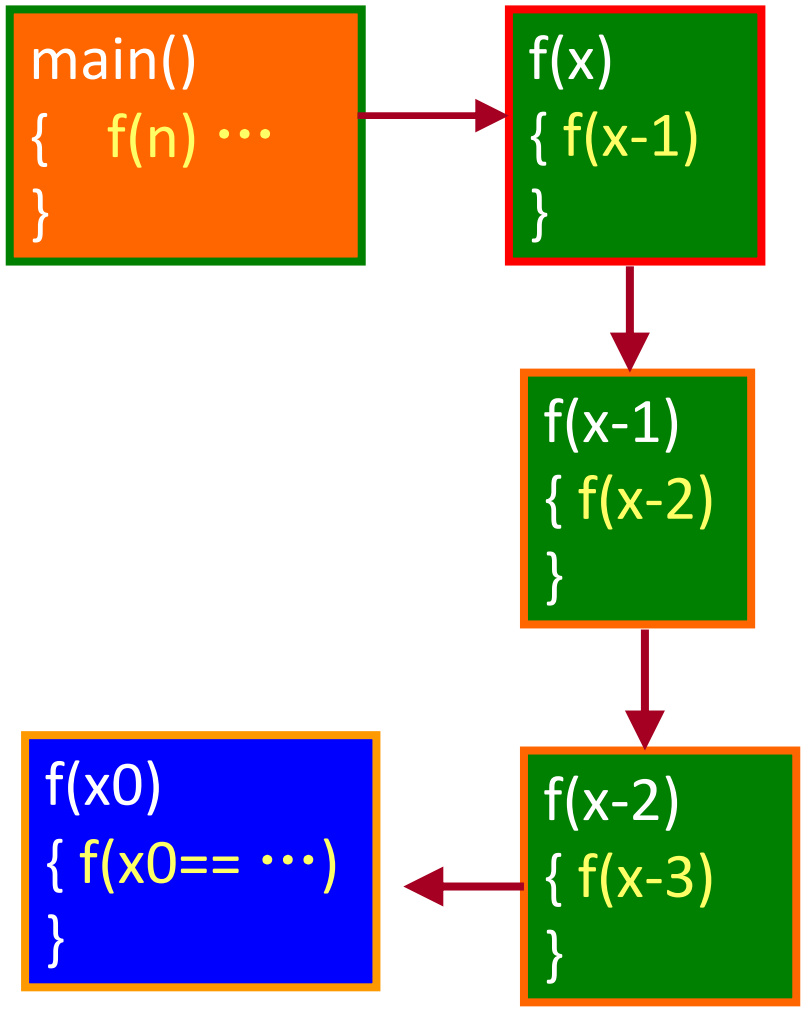
$$age(n)=\begin{cases}10 & (n=1)\\ age(n-1)+2 & (n>1)\end{cases}$$

函数的递归调用

递归算法

其一般形式是：
在函数外用终值n（目标值）调用递归函数，而在递归函数中：

```
递归函数名f(参数x)
{
    if (n==初值)
      结果=…;
    else
      结果=含f(x-1)的表达式;
    返回结果（return）;
}
```



程序如下：

```
def age(n):
    if (n==1):
            c=10
    else:
            c=age(n-1)+2
    return c


age(5)
```

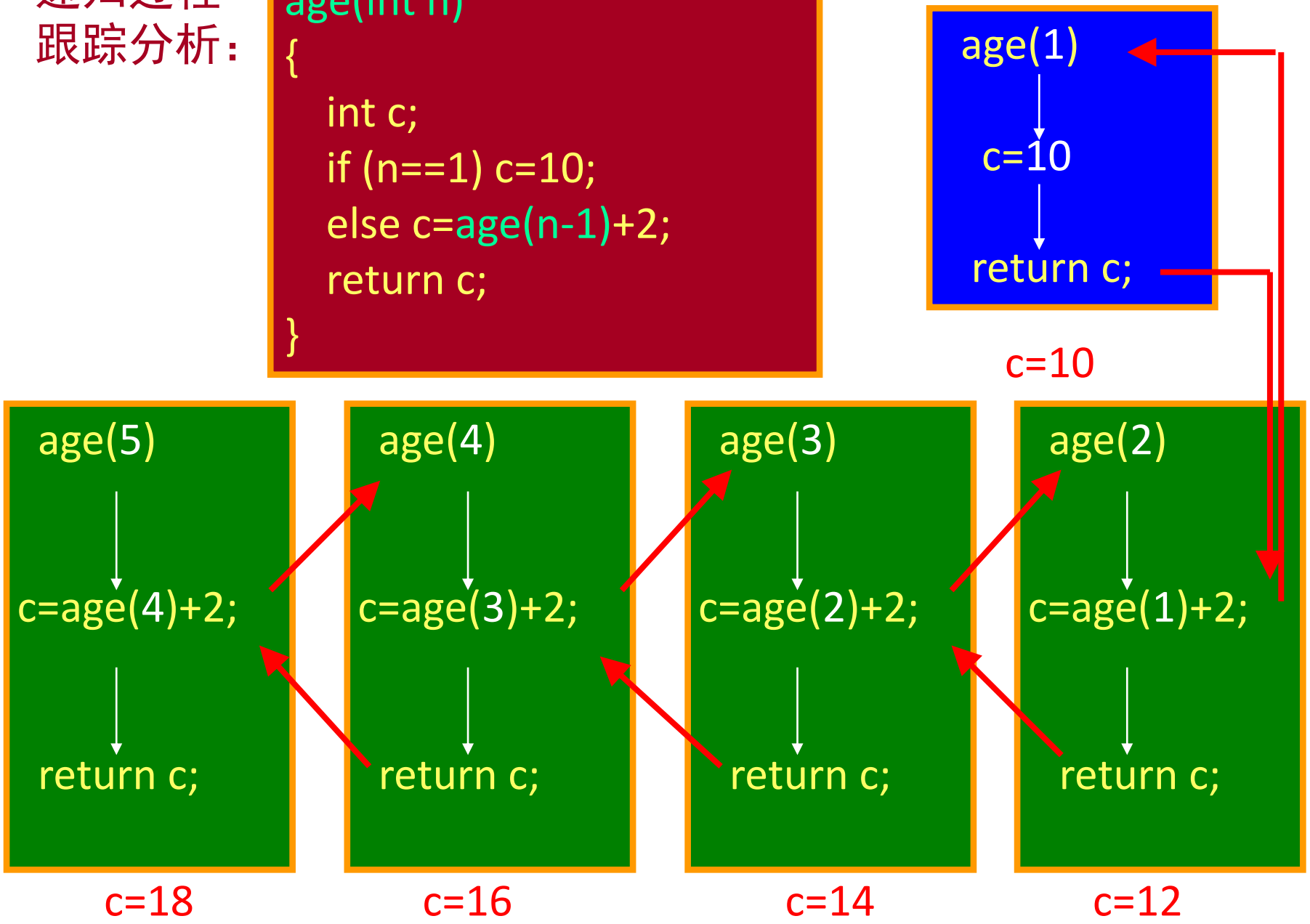
递归函数：

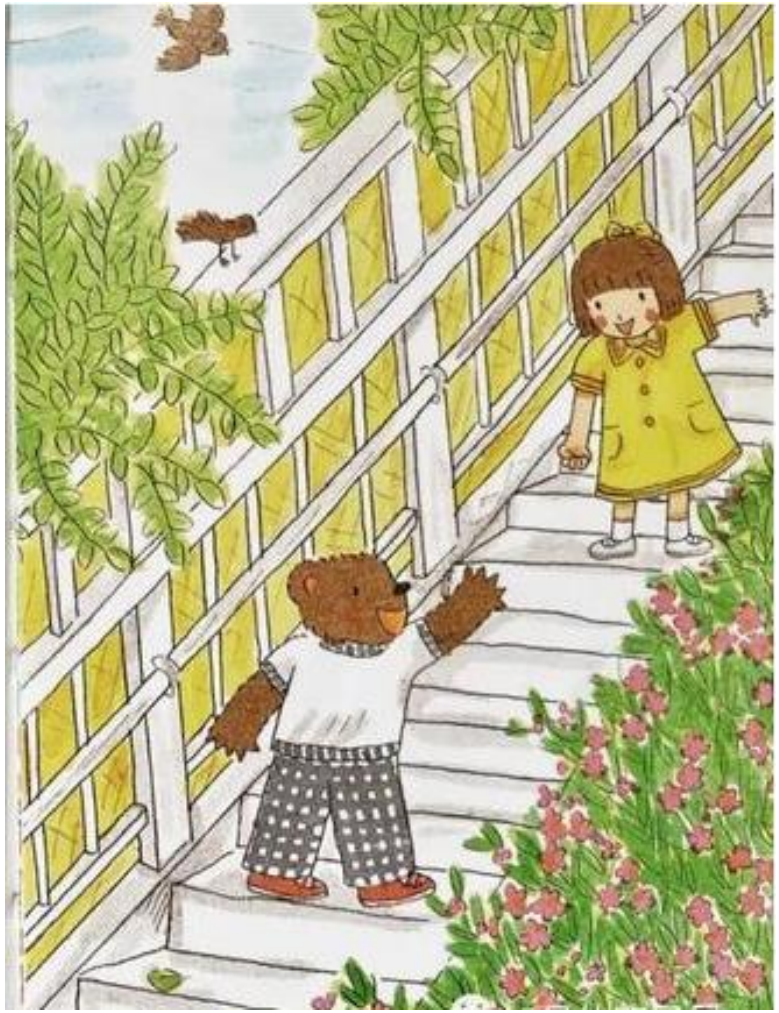
$$age(n)=\begin{cases}10 & (n=1)\\ age(n-1)+2 & (n>1)\end{cases}$$

请看看单步运行的情况……

递归过程
跟踪分析：

```
age(int n)
{
    int c;
    if (n==1) c=10;
    else c=age(n-1)+2;
    return c;
}
```



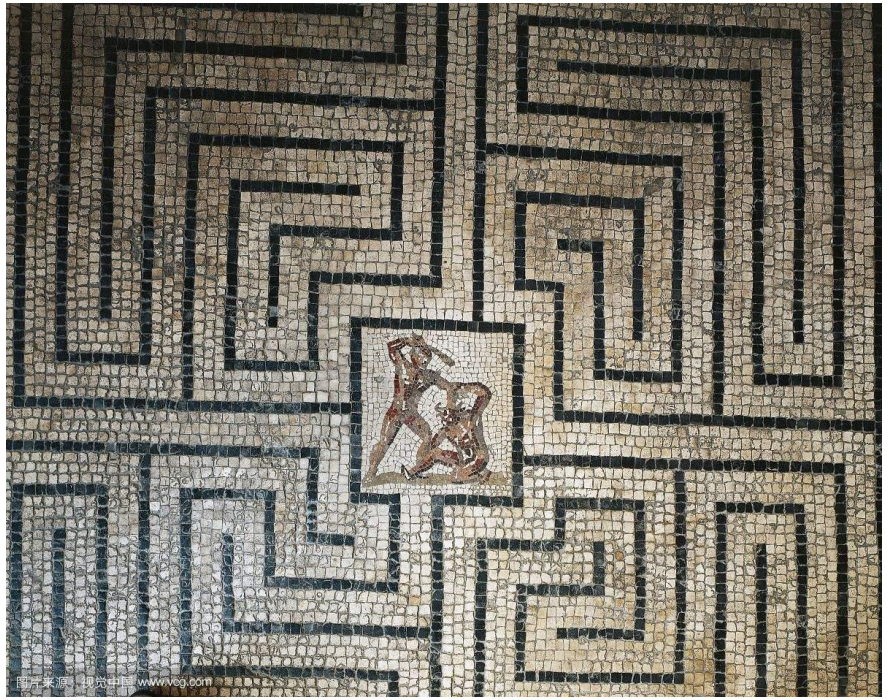


为什么会有这种无聊的设计？

不如问什么是否会出现：

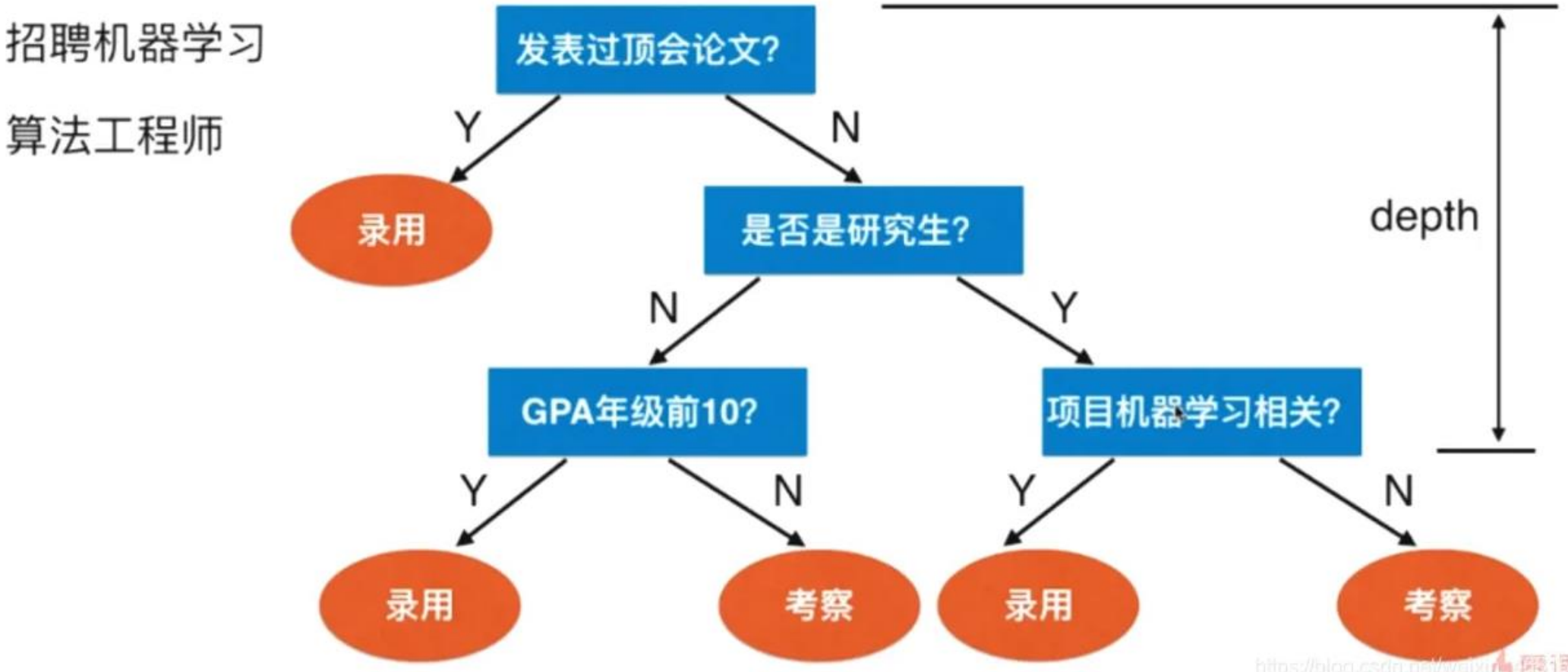
1. 后续每一步依赖上一步的基础 f(n)=f(n-1)
2. 只知道每一步之间的联系，但不知道整体情况 f(n)?

米诺斯的迷宫



米诺斯是宙斯和欧罗巴的儿子，欧罗巴被天后赫拉排挤迫害，来到克里特岛，与当地国王结婚。之后，米诺斯成为克里特国王。米诺斯因为得罪了海神波塞冬，波塞冬便诱使米诺斯的妻子爱上一只公牛，生下了牛首人身的米诺陶洛斯。后来，因为雅典人杀死了米诺斯的另一个儿子，米诺斯求宙斯给雅典施加瘟疫，雅典被迫每年选送7对童男童女去供奉怪物米诺陶洛斯。

管导的决策树，财务的资金优化，人工智能的决策树和神经网络



中国邮递员问题 -> 物流/快递的调度

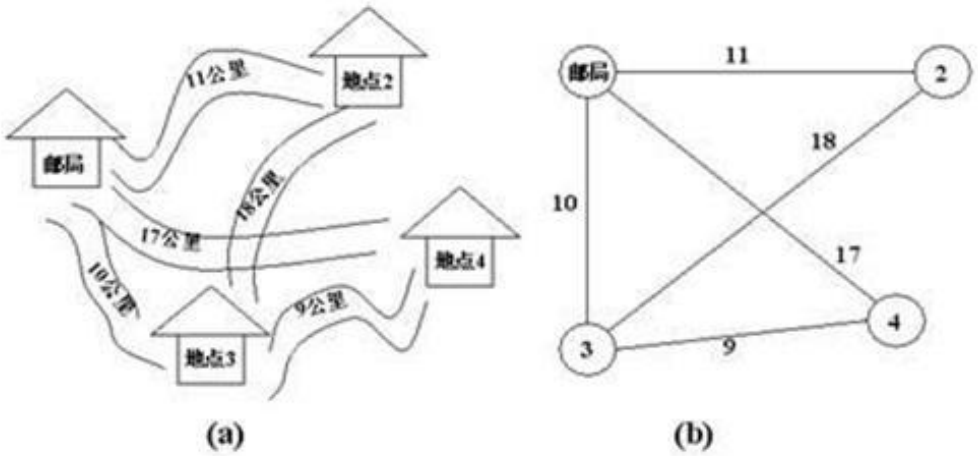


图1.3 邮递员送信问题

表扬

王泽田同学为大家贡献太极包的信息

陶东洋同学为大家贡献逻辑运算示例

刘懿洋同学为大家贡献python多版本导致安装包的问题

https://blog.51cto.com/u_15637561/5291156

薛思远同学为大家贡献正则表达式练习网站

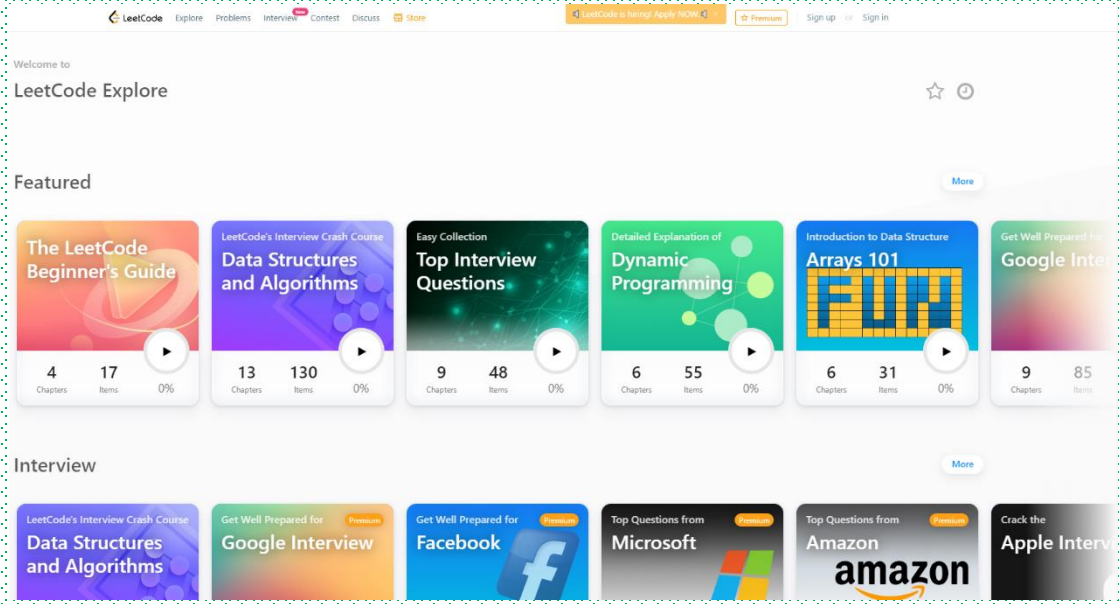
https://regex101.com/

张诚同学为大家贡献刷题网站

https://www.luogu.com.cn （洛谷，信息竞赛题库）

https://leetcode.com （企业笔试真题）

以及众多在作业中用了超前技巧的同学



第6部分　文件和文件夹

书第9章 文件内容操作

- 为了长期保存数据以便重复使用、修改和共享，必须将数据以文件的形式存储到外部存储介质(如磁盘、U盘、光盘或云盘、网盘、快盘等)中。

- 文件操作在各类应用软件的开发中均占有重要的地位：
  - ✓ 管理信息系统是使用数据库来存储数据的，而数据库最终还是要以文件的形式存储到硬盘或其他存储介质上。
  - ✓ 应用程序的配置信息往往也是使用文件来存储的，图形、图像、音频、视频、可执行文件等等也都是以文件的形式存储在磁盘上的。

文件内容操作

- 按文件中数据的组织形式把文件分为文本文件和二进制文件两类。
  - ✓ **文本文件**：文本文件存储的是常规字符串，由若干文本行组成，通常每行以换行符'\n'结尾。常规字符串是指记事本或其他文本编辑器能正常显示、编辑并且人类能够直接阅读和理解的字符串，如英文字母、汉字、数字字符串。文本文件可以使用字处理软件如gedit、记事本进行编辑。
  - ✓ **二进制文件**：二进制文件把对象内容以字节串(bytes)进行存储，无法用记事本或其他普通字处理软件直接进行编辑，通常也无法被人类直接阅读和理解，需要使用专门的软件进行解码后读取、显示、修改或执行。常见的如图形图像文件、音视频文件、可执行文件、资源文件、各种数据库文件、各类office文档等都属于二进制文件。
  - ✓ 无论是文本文件还是二进制文件，其操作流程基本都是一致的，首先打开文件并创建文件对象，然后通过该文件对象对文件内容进行读取、写入、删除、修改等操作，最后关闭并保存文件内容。

1.1 内置函数open()

```
open(file, mode='r', buffering=-1, encoding=None, errors=None,
     newline=None, closefd=True, opener=None)
```

- ✓ file参数指定了被打开的文件名称。
- ✓ mode参数指定了打开文件后的处理方式。
- ✓ buffering参数指定了读写文件的缓存模式。0表示不缓存，1表示缓存，如大于1则表示缓冲区的大小。默认值是缓存模式。
- ✓ encoding参数指定对文本进行编码和解码的方式，只适用于文本模式，可以使用Python支持的任何格式，如CP936(GBK), UTF-8等。

```
f2 = open( 'c:\file2.txt', 'w')
```

- 默认是以文本模式打开文件，如果要以二进制模式打开，那么就给对应模式加上 b 即可，如 rb、wb、ab、rb+、wb+、ab+ 等。

打开模式	说明
r	读模式（**默认模式**，可省略），如果文件不存在则抛出异常
w	写模式，如果文件已存在，先清空原有内容
x	写模式，创建新文件，如果文件已存在则抛出异常
a	追加模式，不覆盖文件中原有内容
b	二进制模式（可与其他模式组合使用）
t	文本模式（**默认模式**，可省略）
+	读、写模式（可与其他模式组合使用）

r:只能读,不能写,读取的文件如果不存在会报错
r+: 可读写,如果操作的文件不存在会报错，默认从文件指针所在位置开始写入
w:只能写,写的时候会清空文件之前的内容,如果写的文件不存在不会报错，会创建新的文件并写入
w+:可读写,会清空文件内容,文件不存在不会报错，会创建新的文件并写入
换句话说：如果文件存在，w+是清空重写， r+是插入；如果文件不存在，r+根本不会继续

a: 只能写（从文件结束位置开始）,文件不存在不报错,不会清空文件内容
a+: 可读写,文件不存在不报错,不会清空文件内容

1.1 内置函数open()

- 如果执行正常，open()函数返回1个文件对象，通过该文件对象可以对文件进行读写操作。如果指定文件不存在、访问权限不够、磁盘空间不足或其他原因导致创建文件对象失败则抛出异常。

```
f1 = open( 'file1.txt', 'r' )      # 以读模式打开文件
f2 = open( 'file2.txt', 'w')       # 以写模式打开文件
```

- 当对文件内容操作完以后，一定要关闭文件对象，这样才能保证所做的任何修改都确实被保存到文件中。

```
f1.close()
```

问1：不关闭会怎样？
问2：示例里面的文件藏在哪？

1.2 文件对象属性与常用方法

方法	功能说明
close()	把缓冲区的内容写入文件，同时关闭文件，并释放文件对象
flush()	把缓冲区的内容写入文件，但不关闭文件
read([size])	从文本文件中读取size个字符（Python 3.x）的内容作为结果返回，或从二进制文件中读取指定数量的字节并返回，如果省略size则表示读取所有内容
readline()	从文本文件中读取一行内容作为结果返回
readlines()	把文本文件中的每行文本作为一个字符串存入列表中，返回该列表
seek(offset[, whence])	把文件指针移动到新的字节位置，offset表示相对于whence的位置。whence为0表示从文件头开始计算，1表示从当前位置开始计算，2表示从文件尾开始计算，默认为0
tell()	返回文件指针的当前位置
write(s)	把s的内容写入文件
writelines(s)	把字符串列表写入文本文件，不添加换行符

1.3 上下文管理语句with

- 在实际开发中，读写文件应优先考虑使用上下文管理语句with，关键字with可以自动管理资源，不论因为什么原因（哪怕是代码引发了异常）跳出with块，总能保证文件被正确关闭(由python进行关闭，省掉自己写close())，并且可以在代码块执行完毕后自动还原进入该代码块时的上下文，常用于文件操作、数据库连接、网络连接、多线程与多进程同步时的锁对象管理等场合。

```
with open(filename, mode, encoding) as fp:
    #这里写通过文件对象fp读写文件内容的语句
```

- 上下文管理语句with还支持下面的用法：

```
with open('test.txt', 'r') as src, open('test_new.txt', 'w') as dst:
    dst.write(src.read())
```

除了逗号，也可以写成两行（见后）

2 文本文件内容操作举例

- **示例9-1** 向文本文件中写入内容，然后再读出。

```
s = 'Hello world\n文本文件的读取方法\n文本文件的写入方法\n'

with open('sample.txt', 'w') as fp:     #默认使用cp936编码
    fp.write(s)

with open('sample.txt') as fp:          #默认使用cp936编码
    print(fp.read())
```

2 文本文件内容操作举例

- **示例9-2** 将一个CP936编码格式的文本文件中的内容全部复制到另一个使用UTF8编码的文本文件中。

```
def fileCopy(src, dst, srcEncoding, dstEncoding):
    with open(src, 'r', encoding=srcEncoding) as srcfp:
        with open(dst, 'w', encoding=dstEncoding) as dstfp:
            dstfp.write(srcfp.read())

fileCopy('sample.txt', 'sample_new.txt', 'cp936', 'utf8')
```

2 文本文件内容操作举例

- **示例9-3** 遍历并输出文本文件的所有行内容。

```
with open('sample.txt') as fp:          #假设文件采用CP936编码
    for line in fp:                     #文件对象可以直接迭代
        print(line)
```

2 文本文件内容操作举例

- **示例9-4** 假设已有一个文本文件sample.txt，将其中第13、14两个字符修改为测试。

```
with open('sample.txt', 'r+') as fp:
    fp.seek(13)
    fp.write('测试')
```

2 文本文件内容操作举例

- **示例9-5** 假设文件data.txt中有若干整数，所有整数之间使用英文逗号分隔，编写程序读取所有整数，将其按升序排序后再写入文本文件data_asc.txt中。

```
with open('data.txt', 'r') as fp:
    data = fp.readlines()                          #读取所有行
data = [line.strip() for line in data]             #删除每行两侧的空白字符
data = ‘,’ .join(data)                              #合并所有行(用逗号合并)
data = data.split(',')                             #分隔得到所有数字字符串
data = [int(item) for item in data]                #转换为数字
data.sort()                                        #升序排序
data = ','.join(map(str,data))                     #将结果转换为字符串
with open('data_asc.txt', 'w') as fp:              #将结果写入文件
    fp.write(data)
```

2 文本文件内容操作举例

- **示例9-6** 统计文本文件中最长行的长度和该行的内容。

```
with open('sample.txt') as fp:
    result = [0, '']
    for line in fp:
        t = len(line)
        if t > result[0]:
            result = [t, line]
print(result)
```

如果是以下三行，大家猜猜哪行最长？
Hello world
文本文件的读取方法
文本文件的写入方法

2 文本文件内容操作举例

- **示例9-7** 使用标准库json进行数据交换。

```
>>> import json
>>> with open('test.txt', 'w') as fp:
    json.dump({'a':1, 'b':2, 'c':3}, fp) #写入文件

>>> with open('test.txt', 'r') as fp:
    print(json.load(fp))                    #从文件中读取

{'a': 1, 'b': 2, 'c': 3}
```

小知识：json就是指把数据按照字典（“key-value”）的方式进行存储读写，比如：“{”name“ : ”zhuxiao5“}”
更复杂点的如右图（多层嵌套）：

```
{
  "animals": {
    "dog": [
      {
        "name": "Rufus",
        "age":15
      },
      {
        "name": "Marty",
        "age": null
      }
    ]
  }
}
```

2 文本文件内容操作举例

- **示例9-8** 使用csv模块读写文件内容。

```
>>> import csv
>>> with open('test.csv', 'w', newline='') as fp:
    test_writer = csv.writer(fp, delimiter=' ', quotechar='"')
    test_writer.writerow(['red', 'blue', 'green'])  #写入一行内容
    test_writer.writerow(['test_string']*5)

>>> with open('test.csv', newline='') as fp:
    test_reader = csv.reader(fp, delimiter=' ', quotechar='"')
    for row in test_reader:                          #遍历所有行
        print(row)                                   #每行作为一个列表返回
['red', 'blue', 'green']
['test_string', 'test_string', 'test_string', 'test_string', 'test_string']
```

3. 其他常见类型二进制文件操作案例

- **示例9-16** 使用扩展库openpyxl读写Excel 2007以及更高版本的文件。

```
import openpyxl
from openpyxl import Workbook
fn = r'c:\test.xlsx'                                    #文件名
wb = Workbook()                                         #创建工作簿
ws = wb.create_sheet(title='你好，世界')                  #创建工作表
ws['A1'] = '这是第一个单元格'                             #单元格赋值
ws['B1'] = 3.1415926
wb.save(fn)                                             #保存Excel文件
wb = openpyxl.load_workbook(fn)                         #打开已有的Excel文件
ws = wb.worksheets[1]                                   #打开指定索引的工作表
print(ws['A1'].value)                                   #读取并输出指定单元格的值
ws.append([1,2,3,4,5])                                  #添加一行数据
ws.merge_cells('F2:F3')                                 #合并单元格
ws['F2'] = "=sum(A2:E2)"                                #写入公式
for r in range(10,15):
    for c in range(3,8):
        ws.cell(row=r, column=c, value=r*c) #写入单元格数据
wb.save(fn)
```

3. 其他常见类型二进制文件操作案例

- **示例9-17** 把记事本文件test.txt转换成Excel 2007+文件。假设test.txt文件中第一行为表头，从第二行开始是实际数据，并且表头和数据行中的不同字段信息都是用逗号分隔。

```
from openpyxl import Workbook

def main(txtFileName):
    new_XlsxFileName = txtFileName[:-3] + 'xlsx'
    wb = Workbook()
    ws = wb.worksheets[0]
    with open(txtFileName) as fp:
        for line in fp:
            line = line.strip().split(',')
            ws.append(line)
    wb.save(new_XlsxFileName)

main('test.txt')
```

5 文件夹操作：os模块

方法	功能说明
chdir(path)	把path设为当前工作目录
curdir	当前文件夹
environ	包含系统环境变量和值的字典
extsep	当前操作系统所使用的文件扩展名分隔符
get_exec_path()	返回可执行文件的搜索路径
getcwd()	返回当前工作目录
listdir(path)	返回path目录下的文件和目录列表

5 文件夹操作：os模块

方法	功能说明
remove(path)	删除指定的文件，要求用户拥有删除文件的权限，并且文件没有只读或其他特殊属性
rename(src, dst)	重命名文件或目录，可以实现文件的移动，若目标文件已存在则抛出异常，不能跨越磁盘或分区
replace(old, new)	重命名文件或目录，若目标文件已存在则直接覆盖，不能跨越磁盘或分区
scandir(path='.')	返回包含指定文件夹中所有DirEntry对象的迭代对象，遍历文件夹时比listdir()更加高效
sep	当前操作系统所使用的路径分隔符
startfile(filepath [, operation])	使用关联的应用程序打开指定文件或启动指定应用程序
system()	启动外部程序

5 文件夹操作： os模块

```
>>> import os
>>> import os.path
>>> os.rename('C:\\dfg.txt', 'D:\\test2.txt') #rename()可以实现文件的改名和移动
>>> [fname for fname in os.listdir('.')
     if fname.endswith(('.pyc', '.py', '.pyw'))]  #结果略
>>> os.getcwd()                                  #返回当前工作目录
'C:\\Python35'
>>> os.mkdir(os.getcwd()+'\\temp')          #创建目录
>>> os.chdir(os.getcwd()+'\\temp')          #改变当前工作目录
>>> os.getcwd()
'C:\\Python35\\temp'
>>> os.mkdir(os.getcwd()+'\\test')
>>> os.listdir('.')
['test']
>>> os.rmdir('test')                        #删除目录
>>> os.listdir('.')
[]
```

5 文件夹操作： os模块

```
>>> os.environ.get('path')                          #获取系统变量path的值
>>> import time
>>> time.strftime('%Y-%m-%d %H:%M:%S',     #查看文件创建时间
                  time.localtime(os.stat('yilaizhuru2.py').st_ctime))
'2016-10-18 15:58:57'
>>> os.startfile('notepad.exe')                     #启动记事本程序
```

os.path模块

方法	功能说明
abspath(path)	返回给定路径的绝对路径
basename(path)	返回指定路径的最后一个组成部分
commonpath(paths)	返回给定的多个路径的最长公共路径
commonprefix(paths)	返回给定的多个路径的最长公共前缀
dirname(p)	返回给定路径的文件夹部分
exists(path)	判断文件是否存在
getatime(filename)	返回文件的最后访问时间
getctime(filename)	返回文件的创建时间
getmtime(filename)	返回文件的最后修改时间
getsize(filename)	返回文件的大小

os.path模块

方法	功能说明
isabs(path)	判断path是否为绝对路径
isdir(path)	判断path是否为文件夹
isfile(path)	判断path是否为文件
join(path, *paths)	连接两个或多个path
realpath(path)	返回给定路径的绝对路径
relpath(path)	返回给定路径的相对路径，不能跨越磁盘驱动器或分区
samefile(f1, f2)	测试f1和f2这两个路径是否引用的同一个文件
split(path)	以路径中的最后一个斜线为分隔符把路径分隔成两部分，以元组形式返回
splitext(path)	从路径中分隔文件的扩展名
splitdrive(path)	从路径中分隔驱动器的名称

os.path模块

```
>>> path='D:\\mypython_exp\\new_test.txt'
>>> os.path.dirname(path)                    #返回路径的文件夹名
'D:\\mypython_exp'
>>> os.path.basename(path)                   #返回路径的最后一个组成部分
'new_test.txt'
>>> os.path.split(path)                      #切分文件路径和文件名
('D:\\mypython_exp', 'new_test.txt')
>>> os.path.split('')                        #切分结果为空字符串
('', '')
>>> os.path.split('C:\\windows')             #以最后一个斜线为分隔符
('C:\\', 'windows')
>>> os.path.split('C:\\windows\\')
('C:\\windows', '')
>>> os.path.splitdrive(path)                 #切分驱动器符号
('D:', '\\mypython_exp\\new_test.txt')
>>> os.path.splitext(path)                   #切分文件扩展名
('D:\\mypython_exp\\new_test', '.txt')
```

os.path模块

```
>>> os.path.commonpath([r'C:\windows\notepad.exe', r'C:\windows\system'])
'C:\\windows'
>>> os.path.commonpath([r'a\b\c\d', r'a\b\c\e'])     #返回路径中的共同部分
'a\\b\\c'
>>> os.path.commonprefix([r'a\b\c\d', r'a\b\c\e'])  #返回字符串的最长公共前缀
'a\\b\\c\\'
>>> os.path.realpath('tttt.py')                      #返回绝对路径
'C:\\Python 3.5\\tttt.py'
>>> os.path.abspath('tttt.py')                       #返回绝对路径
'C:\\Python 3.5\\tttt.py'
>>> os.path.relpath('C:\\windows\\notepad.exe')      #返回相对路径
'..\\windows\\notepad.exe'
>>> os.path.relpath('D:\\windows\\notepad.exe')      #相对路径不能跨越分区
ValueError: path is on mount 'D:', start on mount 'C:'
>>> os.path.relpath('C:\\windows\\notepad.exe','dlls')
                                                     #指定相对路径的基准位置
'..\\..\\windows\\notepad.exe'
```

6. 应用举例

- **示例10-1** 把指定文件夹中的所有文件名批量随机化，保持文件类型不变。

```
from string import ascii_letters
from os import listdir, rename
from os.path import splitext, join
from random import choice, randint

def randomFilename(directory):
    for fn in listdir(directory):
        #切分，得到文件名和扩展名
        name, ext = splitext(fn)
        n = randint(5, 20)
        #生成随机字符串作为新文件名
        newName = ''.join((choice(ascii_letters) for i in range(n)))
        #修改文件名
        rename(join(directory, fn), join(directory, newName+ext))

randomFilename('C:\\test')
```