# Notes\_Python

## 1.常用代码

### 1.1代码片段和知识

#### 1.1.1python基础代码

|  |  |  |
| --- | --- | --- |
|  | 代码 | 功能 |
| 基础操作 | help() | 查看函数功能 |
| import os  os.chdir("C:\\Users\\Administrator\\Desktop\\python\_operation")  os.getcwd() | 设置工作路径 |
| total = item\_one + \  item\_two + \  item\_three | 书写多行语句 |
| print | word\_example=’ abc ‘  print(word\_example.rstrip())  print(word\_example.lstrip())  print(word\_example.strip())  print(word\_example.title()) | 删除右侧空白  删除左侧空白  删除两侧空白  第一字符大写 |
| print (‘’’  ‘’’)  print(‘abc’+A[0]+’def’) | 打印多行字符串  打印多个字符 |
| print（”abc\nefg”）  print（”abc\n\tefg”） | 打印换行符  打印换行符并在开头空一行 |
| ‘abc’  ”ab’c” | 打印字符串(双引号可以防止和字符中单引号冲突 ) |
|  | .count('A') | 计数字符串A的数目 |
| 条件格式if | if age < 4:  price = 0  elif age < 18:  price = 5  elif age < 65:  price = 10  elif age >= 65:  price = 5 | 多重if循环 |
| 条件格式while | active=True  while active:  message=input(prompt)  if message==’quit‘:  active=False  else:  print(message)  message = “”  while message != ‘quit’:  message=input(prompt)  print(message)  if message==’quit’:  break | 在输入quit时结束循环 |
| 变量设置 | a, b = b, a | a和b的值互换 |
| (a, b) = readfasta(lines) | 使用函数对a和b复制 |
| x = 3 if (y == 1) else 2 | 如果y等于1就把x赋值给3，否则赋值给2 |
| 输入字符 | A=input（”what is A?”） |  |
| pip模块 | curl [https://bootstrap.pypa.io/get-pip.py -o get-pip.py](https://bootstrap.pypa.io/get-pip.py%20-o%20get-pip.py)  python get-pip.py  pip install -U pip  python -m pip install -U pip | 下载pip安装模块  安装pip  linux更新pip  windows更新pip |
|  |  |  |

1.2.2python 中的是。

所以

#### 1.1.2创建数值格式

|  |  |  |
| --- | --- | --- |
| 列表\_基础 | A=[“abc”,”efg”]  print（A[0].title()）  print(A[-1])  A.append(“hij”)  A.insert(0,”klm”)  del A[0]  A.pop[0]  A.remove(‘abc’) | 设置列表  打印列表的第一个元素并设置首字母大写  打印列表的最后一个元素  在列表末尾添加元素  在列表第0个位置添加元素  删除列表第0个位置的元素  删除列表第0个位置的元素  删除内容为’abc’的元素 |
| 列表\_截取 | print(A[2:])  print(a[i:j]) b  print(A[-3:])  B=A[:]  set(A)  print（A.sort(reverse=true)）  [::-1]  A[::3] = 1 | 取第2个元素之后的所有元素  取a[i]到a[j-1]  输出最后三个元素  复制整个列表  取得列表中不重复的部分  把列表A的顺序由小到大排序并打印列表A  python的slice notation的特殊用法。a[::-1]相当于 a[-1:-len(a)-1:-1]，也就是从最后一个元素到第一个元素复制一遍。所以你看到一个倒序的列表。  列表中每3个元素改为1（以3为步长） |
| seq\_1[i:i + len(seq\_2)] | 截取序列中长度为len(seq\_2)的部分 |
| 产生序列 | range(2,5,1)#= range(2,5)  A=list(range(2,11,2))  range(2,11,2)#≈ xrange(2,11,2) | 生成差为1的等差序列#差为1时1可以省略  生成等差序列并形成列表  生成差为2，最大值为11的等差序列（xrange返回一个可迭代的对象，而range返回的是一个列表，要生成很大的数字序列的时候，用xrange会比range性能优很多，因为不需要一上来就开辟一块很大的内存空间。） |
| 集合 | s = {1,2,3,1,2,3,4,5}  print(list(set(li))) | 创建集合  使用集合的特性去除列表中的重复元素 |
| 布尔表达式 | a=True | 设置a为真 |
| 字典 | C={‘a’:’b’,’c’:’d’}  C[‘a’]  C[‘a’]=’b’  del C[‘a’]=’b’  for key,value in C.item(): | 创建字典  查看字典的值  设置字典的值  删除键值对  遍历字典中的键值对 |
| 函数 | def greet\_user(a,b=”efg”):  print(“hello”+a+b)  greet\_user(a=”abc”,b) | 定义函数，并设置 |
| 类 | class Dog()  def \_\_init\_\_(self,name,age):  #初始化属性name和age  self.name = name  self.age = age  def sit(self):  #模拟小狗被命令时蹲下  print(self.name.title()+”is now sitting.”  def roll\_over(self):  print(self.name.title()+”rolled over!”) | 创建名为Dog的类，\_\_init\_\_()具有三个形参，我们通过实参向Dog()传递变量，而每个与类相关联的方法调用都会自动传递实参self |
|  |  |  |

#### 1.1.3文件读取和批量处理

|  |  |  |
| --- | --- | --- |
| 代码 | 格式 | 备注 |
| import pandas as pd  pd.read\_csv('C:\\test0424\\data.csv') | csv | #read\_csv默认读取用‘，’分隔符的文件，不需要用sep来指定分隔符 |
| pd.read\_table('C:\\test0424\\data.txt') | txt | 直接输入原文件，不按‘，’分隔 |
| pd.read\_csv('C:\\test0424\\data.txt',sep='|')  pd.read\_table('C:\\test0424\\data.txt',sep=’|’) | txt | 设定读取数据，按‘|’分隔 |
| pd.read\_csv('C:\\data1.txt',header=None,  skiprows=[0,3,6]) | txt | 指定读入的行 |
| data.to\_csv('C:\\outdata2.txt',sep=',',index=False,columns=['a','c','d']) | txt | 将‘！’作为分隔符，生成txt文件  不加入标签，并指定输入的列 |
| chunker = pd.read\_csv('C:\\ata1.txt',chunksize=3)  for m in chunker:  print(len(m))  print m | txt | 分块读取 |
| with open(‘example\_file.txt’) as example\_file:  contents = file\_object.read()  print(contents) | txt | 打开文件并输出全部内容 |
| with open(‘example\_file.txt’,’w’) as example\_file:  for lines in example\_file.readlines()  #oneline=’’  for line in lines:  print(line.rstrip())  #oneline += line.strip() | txt | 打开文件并按行输出全部内容  #并将所有的行合并为1行 |
| f = open('rosalind\_subs.txt', 'r')  s = f.read() | txt | 直接读取整个文件 |
| f = open('rosalind\_subs.txt', 'r')  s = f.readlines() | txt | 按行读取文件，把每行作为一个元素 |

|  |  |
| --- | --- |
| r | 以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。 |
| rb | 以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。这是默认模式。 |
| r+ | 打开一个文件用于读写。文件指针将会放在文件的开头。 |
| rb+ | 以二进制格式打开一个文件用于读写。文件指针将会放在文件的开头。 |
| w | 打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。 |
| wb | 以二进制格式打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。 |
| w+ | 打开一个文件用于读写。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。 |
| wb+ | 以二进制格式打开一个文件用于读写。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。 |
| a | 打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。 |
| ab | 以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。 |
| a+ | 打开一个文件用于读写。如果该文件已存在，文件指针将会放在文件的结尾。文件打开时会是追加模式。如果该文件不存在，创建新文件用于读写。 |
| ab+ | 以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。如果该文件不存在，创建新文件用于读写。 |

|  |  |
| --- | --- |
|  |  |
| import glob  import os  os.chdir(r"G:\myPictures")  for file\_name in glob.glob("\*.jpg"):  print file\_name | #指定目录 TODO:支持中文路径  #获取指定目标下所有的jpg文件的文件名  glob模块根据Unix shell使用的规则查找与指定模式匹配的所有路径名，结果以任意顺序返回。 |
| import glob  glob.glob('./[0-9].\*')  glob.glob('\*.gif')  glob.glob('?.gif')  glob.glob('\*\*/\*.txt', recursive=True)  glob.glob('./\*\*/', recursive=True) | ['./1.gif', './2.txt']  ['1.gif', 'card.gif']  ['1.gif']  ['2.txt', 'sub/3.txt']  ['./', './sub/']  <https://docs.python.org/3/library/glob.html> |

#### 1.1.5list数目

list = []

for value in range(1,11):

list.append(value\*\*3)

for i in range(0,10):

print(list[i])

#上下范围均为1-10

#### 1.1.6好的书写习惯

if (age <= 12) and (age >= 3):

while not

#### 1.1.7格式化符号和转义符

|  |  |  |
| --- | --- | --- |
| 判断 | > | 大于 |
| < | 小于 |
| >= | 大于等于 |
| <= | 小于等于 |
| == | 等于。比较两个值是否相等。之所以用两个等号，是为了和变量赋值区分开来。 |
| != | 不等于 |
| += | 左侧加上右侧的值 |
| 格式化符号 | %c | 转换成字符（ASCII 码值，或者长度为一的字符串） |
| %r | 优先用repr()函数进行字符串转换 |
| %s | 优先用str()函数进行字符串转换 |
| %d / %i | 转成有符号十进制数 |
| %u | 转成无符号十进制数 |
| %o | 转成无符号八进制数 |
| %x / %X | 转成无符号十六进制数（x / X 代表转换后的十六进制字符的大小写） |
| %e / %E | 转成科学计数法（e / E控制输出e / E） |
| %f / %F | 转成浮点数（小数部分自然截断） |
| %g / %G | %e和%f / %E和%F 的简写 |
| %% | 输出% （格式化字符串里面包括百分号，那么必须使用%%） |
| 转义字符 | \(在行尾时) | 续行符 |
| \\ | 反斜杠符号 |
| \’ | 单引号 |
| \” | 双引号 |
| \a | 响铃 |
| \b | 退格(Backspace) |
| \e | 转义 |
| \000 | 空 |
| \n | 换行 |
| \v | 纵向制表符，垂直空出几个格 |
| \t | 横向制表符，水平制表符就是水平空出几个格。令 num = |n-8|%8，那么\t对应的空格的个数 spaceNum,有： |
| \r | 回车 |
| \f | 换页 |
| \oyy | 八进制数yy代表的字符，例如：\o12代表换行 |
| \xyy | 十进制数yy代表的字符，例如：\x0a代表换行 |
| \other | 其它的字符以普通格式输出 |

#### 1.1.8格式切换

|  |  |
| --- | --- |
| 函数 | 描述 |
| [int(x [,base])](http://www.runoob.com/python/python-func-int.html) | 将x转换为一个整数 |
| [long(x [,base] )](http://www.runoob.com/python/python-func-long.html) | 将x转换为一个长整数 |
| [float(x)](http://www.runoob.com/python/python-func-float.html) | 将x转换到一个浮点数 |
| [complex(real [,imag])](http://www.runoob.com/python/python-func-complex.html) | 创建一个复数 |
| [str(x)](http://www.runoob.com/python/python-func-str.html) | 将对象 x 转换为字符串 |
| [repr(x)](http://www.runoob.com/python/python-func-repr.html) | 将对象 x 转换为表达式字符串 |
| [eval(str)](http://www.runoob.com/python/python-func-eval.html) | 用来计算在字符串中的有效Python表达式,并返回一个对象 |
| [tuple(s)](http://www.runoob.com/python/att-tuple-tuple.html) | 将序列 s 转换为一个元组 |
| [list(s)](http://www.runoob.com/python/att-list-list.html) | 将序列 s 转换为一个列表 |
| [set(s)](http://www.runoob.com/python/python-func-set.html) | 转换为可变集合 |
| [dict(d)](http://www.runoob.com/python/python-func-dict.html) | 创建一个字典。d 必须是一个序列 (key,value)元组。 |
| [frozenset(s)](http://www.runoob.com/python/python-func-frozenset.html) | 转换为不可变集合 |
| [chr(x)](http://www.runoob.com/python/python-func-chr.html) | 将一个整数转换为一个字符 |
| [unichr(x)](http://www.runoob.com/python/python-func-unichr.html) | 将一个整数转换为Unicode字符 |
| [ord(x)](http://www.runoob.com/python/python-func-ord.html) | 将一个字符转换为它的整数值 |
| [hex(x)](http://www.runoob.com/python/python-func-hex.html) | 将一个整数转换为一个十六进制字符串 |
| [oct(x)](http://www.runoob.com/python/python-func-oct.html) | 将一个整数转换为一个八进制字符串 |

### 1.2实际操作代码

#### 1.2.1.词频统计

filename = input(“enter a filename:”).strip()

infile = open(filename, ”r”)

wordCounts = { }

for line in infile:

processLine(line.lower(), wordCounts)

pairs = list(wordCounts.items())

items = [[x,y] for (y, x) in pairs]

items.sort()

turtle.title(‘词频结果柱状图’)

turtle.setup(900, 750, 0, 0)

t = turtle.Turtle()

t.hideturtle()

t.width(3)

drawGraph(t)

def processLine(line, wordCounts):

line = replacePunctuations(line)

words = line.split()

for word in words:

if word in wordCounts:

word Counts[word] += 1

else:

wordCounts[word] = 1

#空格替换为标点

def replacePunctuation(line):

for ch in line:

if ch in “~!@#$%%^&\*()\_{}|:”<>?/.,;’[]\=-“’”””:

line = line.replace(ch, “ “)

return line

def drawLine(t, x1, y1, x2, y2):

t.penup()

t.goto (x1, y1)

t.pendown()

t.goto (x2, y2)

def drawText(t, x, y, text):

t.penup()

t.goto (x, y)

t.pendown()

t.write(text)

def drawRectangle(t, x, y)

x = x\*xScale

y = y\*yScale

drawLine(t, x-5, 0, x-5, y)

drawLine(t, x-5, y, x+5, y)

drawLine(t, x+5, 0, x+5, y)

drawLine(t, x+5, 0, x-5, y)

def drawBar(t):

for in range(count):

drawRectangle(t, i+1, data[i])

def drawGraph(t):

drawLine (t, 0, 0, 360, 0)

drawLine (t, 0, 300, 0, 0)

for x in range(count):

x = x + 1

drawText(t, x\*xScale-4, -20, (words[x-1]))

drawText(t, x\*xScale-4, data[x-1]\*yScale+10, data[x-1])

drawBar(t)

#### 1.2.2迭代产生的列表生成代码

#通过函数判断变量可否迭代

>>> from collections import Iterable

>>> isinstance('abc', Iterable) # str是否可迭代

True

>>> isinstance([1,2,3], Iterable) # list是否可迭代

True

>>> isinstance(123, Iterable) # 整数是否可迭代

False

#使用迭代直接产生计算结果

>>> [x \* x for x in range(1, 11)]

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

#使用迭代拼接字符

>>> [m + n for m in 'ABC' for n in 'XYZ']

['AX', 'AY', 'AZ', 'BX', 'BY', 'BZ', 'CX', 'CY', 'CZ']

#列出当前目录下的所有文件和目录名

>>> import os # 导入os模块，模块的概念后面讲到

>>> [d for d in os.listdir('.')] # os.listdir可以列出文件和目录

['.emacs.d', '.ssh', '.Trash', 'Adlm', 'Applications', 'Desktop', 'Documents', 'Downloads', 'Library', 'Movies', 'Music', 'Pictures', 'Public', 'VirtualBox VMs', 'Workspace', 'XCode']

#### 1.2.3使默认参数指向不变对象，去除使用append（）中的bug

def add\_end(L=None):

if L is None:

L = []

L.append('END')

return L

1.2.4导入fasta文件

def readfasta(lines):  
 *"""读入并处理FASTA文件的函数"""* seq = []  
 index = []  
 seqplast = ""  
 numlines = 0  
  
 for i in lines:  
 if '>' in i: # 判断是序列行还是说明行  
 index.append(i.replace("\n", "").replace(">", ""))  
 seq.append(seqplast.replace("\n", ""))  
 seqplast = ""  
 numlines += 1  
 else:  
 seqplast = seqplast + i.replace("\n", "") # 把分行的序列拼接成一个字符串  
 numlines += 1  
 if numlines == len(lines):  
 seq.append(seqplast.replace("\n", ""))  
 seq = seq[1:]  
 return index, seq

### 1.3模块介绍

#### 1.3.1常用的模块

list =[“numpy”,”biopython”,”pandas”

|  |  |  |
| --- | --- | --- |
| 模块名 | 特征 | 功能 |
| numpy | 所有科学计算更高层工具的基础 | 1. N维数组。2. 可以不需要使用循环，就对整个数组内的数据行标准数学运算。3. 非常便于传送数据到用低级语言（如C或C++）编写的外部库，也便于外部库以Numpy数组形式返回数据。 |
| Scipy | 科学计算的模块 | 它提供便捷和快速的N维向量数组操作。提供模块用于优化、线性代数、积分以及其它数据科学中的通用任务。 |
| pandas | 包含高级数据结构，以及和让数据分析变得快速、简单的工具。 | 1. 带有坐标轴的数据结构，支持自动或明确的数据对齐。3. 合并流行数据库（如：基于SQL的数据库）中能找到的关系操作。Pandas是进行数据清洗/整理（data munging）的最好工具。 |
| Matlplotlib | Python的一个可视化模块。 | 它可方便地制作线条图、饼图、柱状图以及其它专业图形。 |
| Scikit-learn | 一个用于机器学习的Python模块,建立在Scipy之上。 | Classification、 Regression、 Clustering、 Dimensionality reduction、 Model selection、 Preprocessing。 |
|  |  |  |
|  |  |  |

#### 1.3.2os模块

import os

print(os.getcwd()) # 打印当前工作目录

os.chdir('/Users/<username>/Desktop/') # 将当前工作目录改变为`/Users/<username>/Desktop/`

#### 1.3.3pip模块

python -m pip install --upgrade pip

<https://pip.pypa.io/en/stable/user_guide/#running-pip>

1.3.4re模块

|  |  |  |
| --- | --- | --- |
| 代码 |  |  |
| re.match(regex, content) | 匹配string 开头，成功返回Match object, 失败返回None，只匹配一个。 |  |
| re.search(regex, content) | 在string中进行搜索，成功返回Match object, 失败返回None, 只匹配一个。 |  |
| re.findall(regex, content) | 在string中查找所有 匹配成功的组, 即用括号括起来的部分。返回list对象，每个list item是由每个匹配的所有组组成的list。 |  |
| re.finditer(regex, content) | 在string中查找所有 匹配成功的字符串, 返回iterator，每个item是一个Match object。 |  |
| re.match(r"\d/", info) | 使用Python的r前缀，就不用考虑转义的问题 |  |
| re.sub(r"\d/", "", info) | 使用re替换string中每一个匹配的子串后返回替换后的字符串。 |  |

### 1.4行列处理

添加新的列

import pandas as pd

feature = pd.read\_csv("C://Users//Machenike//Desktop//xzw//lr\_train\_data.txt", delimiter="\t", header=None, usecols=[0, 1])

feature.columns = ["a","b"]

print(feature.head())

feature = feature.reindex(columns=list('cabd'), fill\_value=1)

print(feature.head())

我们添加了c、d两列，结果如下：

a b

0 4.459256 8.225418

1 0.043276 6.307400

2 6.997162 9.313393

3 4.754832 9.260378

4 8.661904 9.767977

c a b d

0 1 4.459256 8.225418 1

1 1 0.043276 6.307400 1

2 1 6.997162 9.313393 1

3 1 4.754832 9.260378 1

4 1 8.661904 9.767977 1

feature = feature.reindex(columns=list('cab'), fill\_value=1)

print(feature.head())

a b

0 4.459256 8.225418

1 0.043276 6.307400

2 6.997162 9.313393

3 4.754832 9.260378

4 8.661904 9.767977

c a b

0 1 4.459256 8.225418

1 1 0.043276 6.307400

2 1 6.997162 9.313393

3 1 4.754832 9.260378

4 1 8.661904 9.767977

### 1.5数学统计代码

#### 1.5.1常用数学代码

|  |  |
| --- | --- |
| 代码 | 功能 |
| map(function,num)  def f(x):  ... return x \* x  r = map(f, [1, 2, 3, 4, 5, 6, 7, 8, 9])  list(r) | map将传入的函数依次作用到序列的每个元素，并把结果作为新的Iterator返回。  [1, 4, 9, 16, 25, 36, 49, 64, 81] |
| reduce（[function,num1,num2]）  from functools import reduce  def fn(x, y):  ... return x \* 10 + y  reduce(fn, [1, 3, 5, 7, 9]) | 把一个函数作用在一个序列[x1, x2, x3, ...]上，这个函数必须接收两个参数，reduce把结果继续和序列的下一个元素做累积计算  13579 |
| filter() | filter()接收一个函数和一个序列，把传入的函数依次作用于每个元素，然后根据返回值是True还是False决定保留还是丢弃该元素。 |
| def not\_empty(s):  return s and s.strip()  list(filter(not\_empty, ['A', '', 'B', None, 'C', ' '])) | 把一个序列中的空字符串删掉。得到['A', 'B', 'C'] |
|  |  |
|  |  |
|  |  |

#### 1.5.2生成器(generator)

生成器是一次生成一个值的特殊类型函数。可以将其视为可恢复函数。调用该函数将返回一个可用于生成连续 x 值的生成【Generator】，简单的说就是在函数的执行过程中，yield语句会把你需要的值返回给调用生成器的地方，然后退出函数，下一次调用生成器函数的时候又从上次中断的地方开始执行，而生成器内的所有变量参数都会被保存下来供下一次使用，进而节约大量内存。

列表元素可以按照某种算法推算出来，在循环的过程中不断推算出后续的元素。生成器和函数的执行流程不一样。函数是顺序执行，遇到return语句或者最后一行函数语句就返回。变成生成器（generator）的函数，在每次调用next()的时候执行，遇到yield语句返回，再次执行时从上次返回的yield语句处继续执行。

def fib(max):

n, a, b = 0, 0, 1

while n < max:

yield b

a, b = b, a + b

n = n + 1

print(b)

return 'done'

f=fib(5)

f.\_\_next\_\_()

f.\_\_next\_\_()

f.\_\_next\_\_()

f.\_\_next\_\_()

f.\_\_next\_\_()

f.\_\_next\_\_()

f.\_\_next\_\_()

#fib函数实际上是定义了斐波拉契数列的推算规则，可以从第一个元素开始，推算出后续任意的元素。但是用for循环调用generator时，发现拿不到generator的return语句的返回值。如果想要拿到返回值，必须捕获StopIteration错误，返回值包含在StopIteration的value中：

>>> g = fib(6)

>>> while True:

... try:

... x = next(g)

... print('g:', x)

... except StopIteration as e:

... print('Generator return value:', e.value)

... break

#### 1.5.3迭代器

可以直接作用于for循环的对象统称为可迭代对象（Iterable），有两种：一、集合数据类型，如list、tuple、dict、set、str等；二、generator，包括生成器和带yield的generator function。

可作用于next()函数的对象都是Iterator类型，它们表示一个惰性计算的序列。集合数据类型如list、dict、str等是Iterable但不是Iterator，不过可以通过iter()函数获得一个Iterator对象。Python的for循环本质上就是通过不断调用next()函数实现的。通过isinstance()来判断一个对象是否是迭代对象。

生成器都是Iterator对象，但list、dict、str虽然是Iterable，却不是Iterator。因为Python的Iterator对象表示的是一个数据流，Iterator对象可以被next()函数调用并不断返回下一个数据，直到没有数据时抛出StopIteration错误。可以把这个数据流看做是一个有序序列，但我们却不能提前知道序列的长度，只能不断通过next()函数实现按需计算下一个数据，所以Iterator的计算是惰性的，只有在需要返回下一个数据时它才会计算。Iterator甚至可以表示一个无限大的数据流，例如全体自然数。而使用list是永远不可能存储全体自然数的。

## 2.知识

### 2.1python基础知识

#### 2.1.1python程序结构

Python程序结构

1. 程序由模块构成，2.模块包含语句，3. 语句包含表达式，4.表达式建立并处理对象

（python中没有类型声明，运行的表达式决定了建立对象的类型，这点有点儿像matlab。）

Python语法实质上是有语句和表达式组成的。表达式处理对象并嵌套在语句中。语句编程实现程序操作中更大的逻辑关系。此外，语句还是对象生成的地方，有些语句会生成新的对象类型（函数、类等）。语句总是存在于模块中，而模块本身则又是由语句来管理的。

**程序设计步骤**

分析问题的可计算部分-确定IPO组分-设计算法-编写程序-测试和升级

#### 2.1.3python2和python3的区别和切换

3里只有input函数，它接收你输入的字符串，不管你输的是什么。 text = input()

像2一样得到用户输入的一个值 value = eval(input())

或者，如果你只是需要一个整数值，也可以：value = int(input())

打开文件不再支持 file 方法，只能用 open

range不再返回列表，而是一个可迭代的range对象

除法 / 不再是整除，而是得到浮点数，整除需要用双斜杠 //

urllib和urllib2合并成了urllib，常用的urllib2.urlopen()变成了urllib.request.urlopen()

字符串及编码相关有大变动，简单来说就是原来的str变成了新的bytes，原来的unicode变成了新的str。

#### 2.1.4pipenv

pipenv 是一种python包管理工具，完美的解决了python的包和版本的管理，并对包之间的依赖关系也管理起来，方便了开发者构建自己的开发运行环境。

* 之前人们使用pip + virtualenv 来管理python依赖包，使用 --python=参数来区分python版本（不再使用pyenv,减少包依赖）。而pipenv的思路简单理解便是把pip和virutalenv 2个工具统一起来，使用 pipenv 来代替。
* pipenv 使用 Pipfile 来代替 requirement.txt 文件记录python包。
* 增加了Pipfile.lock 文件来锁定python软件的包名及版本，以及其依赖关系的列表。
* 它参考了其他语言的包管理工具（bundler, composer, npm, cargo, yarn, etc.），旨在将最好的包管理工具带入python世界。

|  |  |
| --- | --- |
| 命令 | 注释 |
| pipenv install  pipenv --python 3.7  pipenv install [moduel]  pipenv install [moduel] --dev  pipenv uninstall[module]  pipenv uninstall --all  pipenv --rm | 创建虚拟环境  创建python3.7虚拟环境  安装包  安装包到开发环境  卸载包  卸载所有包  删除虚拟环境 |
| which python3 | 查看当前python3的版本 |
| pipenv shell | 激活虚拟环境 |
| Exit | 退出虚拟环境 |
| pipenv graph | 查看包的依赖关系 |
| pipenv pipenv –venv  pipenv pipenv --where  pipenv --py | 列出本地工程路径  列出虚拟环境路径 |
| pipenv run python [pyfile] | 运行python脚本 |

#### 2.1.6尾递归

尾递归是指，在函数返回的时候，调用自身本身，并且，return语句不能包含表达式。这样，编译器或者解释器就可以把尾递归做优化，使递归本身无论调用多少次，都只占用一个栈帧，不会出现栈溢出的情况。把每一步的乘积传入到递归函数中：

def fact(n):

return fact\_iter(n, 1)

def fact\_iter(num, product):

if num == 1:

return product

return fact\_iter(num - 1, num \* product)

### 2.2概念

#### 2.1.1编程概念

**列表**

由多个元素组成的数据结构。列表可以作为堆栈使用。

**元组**

不可修改的列表。

**字典**

一系列无序的键值对，可以使用键访问其关联的值。

**形参**

函数完成其工作所需的一项信息。

**实参**

调用函数时传递给函数的信息。

**位置实参**

调用函数时，python将函数调用中的每个实参按照实参的顺序关联到函数定义中的一个形参（实参和形参的顺序相同）。

**关键字实参**

在实参中将名称和值关联起来，以键值对的形式传递（在输出函数时写清楚哪个变量对应哪条信息）。

**模块**

模块是扩展名为.py的文件，包含要导入到程序中的代码。可以将函数存储在其中，进而一起导入主程序。可以通过as命令给模块设置别名，进而高效调用模块。

**类（class）&实例**

类是抽象的模板，是一种面向对象的程序设计方法，可以定义一大类对象都有的通用行为。实例是类创建出来的一个个具体的“对象”，每个对象都拥有相同的方法，但各自的数据可能不同。

**集合**

集合（set）是一个无序的不重复元素序列

**封装**

抽象对象，打包数据和操作

**多态**

决定对象如何回应消息，使同一函数名启用不同方法

**继承**

写一个新类并使其自动获得另一个类的所有属性和方法，新类还可以定义自己的属性和方法，此时原有的类称为父类，新类称为子类。一个类可以借用另外一个类的行为定义，可以减少重复代码并强化程序复用性。

**物理行&逻辑行**

物理行是在编写程序时所 看见 的。逻辑行是Python“看见”的单个语句。Python假定每个物理行对应一个逻辑行。Python希望每行都只使用一个语句，这样使得代码更加易读。如果想要在一个物理行中使用多于一个逻辑行，那么需要使用分号（;）来特别地标明这种用法。分号表示一个逻辑行/语句的结束。为了让代码更加易读，强烈建议坚持在每个物理行只写一句逻辑行。

网络专用计算机

物联网

脚本语言

索引的索引

in 运算符可判断当有对象是否序列对象成员

lambda可以创建小型匿名函数，可以节省内存并简洁

2递归函数

所有的递归函数都要以基例结尾

有一个或多个基例无需再次递归

每次调用都会引起新函数的开始

递归有本地值的副本，包括该值的参数。

每次函数调用中的相关n值在递归链暂时存储，在函数返回时使用

字符串反转

构造递归函数需要基例。基例不进行递归，否则递归就会无限循环进行。

3unicode和utf8文件编码格式

readlines和writelines是按照列表操作

4序列类型和字典类型存取方式不同

序列键值对的键可以用数字类型之外的类型的数据

排列方式和映射值的方式不同

7对象命名可以采用目标+功能的方法

9LU分解可以将一个矩阵分解为一个单位下三角矩阵和一个上三角矩阵。可用于数值分析。

列表的计算比单个的计算要快

#### 2.1.2函数编写规范

**python之禅**

The Zen of Python, by Tim Peters

|  |  |  |
| --- | --- | --- |
| 原句 | 解释 | 代码 |
| Beautiful is better than ugly. | #优美胜于丑陋（Python 以编写优美的代码为目标）  不要把程序的某一行写得太长 | 一个带有数字参数的list函数，其功能是返回参数中的奇数  halve\_evens\_only = lambda nums: map(lambda i: i/2,\  filter(lambda i: not i%2, nums))  def halve\_evens\_only(nums):  return [i/2 for i in nums if not i % 2]  使用isinstance 而不是type  不要这样做：  if type(s) == type(""): ...  if type(seq) == list or \  ty应该是这样：  if isinstance(s, basestring): ...  if isinstance(seq, (list, tuple)): ...pe(seq) == tuple: ...  https://stackoverflow.com/questions/1549801/what-are-the-differences-between-type-and-isinstance/1549854#1549854  https://www.jb51.net/article/63423.htm |
| Explicit is better than implicit. | #明了胜于晦涩（优美的代码应当是明了的，命名规范，风格相似） |  |
| Simple is better than complex. | #简洁胜于复杂（优美的代码应当是简洁的，不要有复杂的内部实现） |  |
| Complex is better than complicated. | #复杂胜于凌乱。如果复杂不可避免，那代码间也不能有难懂的关系，要保持接口简洁；必要的复杂逻辑是难免的，而繁复啰嗦的代码是不可接受的  # A complex person is like an iPod. That is to say that they are consistent, straightforward and ‘user friendly’ while also being rather sophisticated. | Complicated:  counter = 0  while counter < 5:  print counter  counter += 1  Complex:  for i in xrange(5):  print i |
| Flat is better than nested. | 扁平胜于嵌套（优美的代码应当是扁平的，不能有太多的嵌套） |  |
| Sparse is better than dense. | 间隔胜于紧凑（优美的代码有适当的间隔，不要奢望一行代码解决问题） |  |
| Readability counts. | 可读性很重要 |  |
| Special cases aren't special enough to break the rules.Although practicality beats purity. |  |  |
| Errors should never pass silently. Unless explicitly silenced. | 不要包容所有错误，除非你确定需要这样做（精准地捕获异常，不写 except:pass 风格的代码） |  |
| In the face of ambiguity, refuse the temptation to guess. There should be one-- and preferably only one --obvious way to do it. | 当存在多种可能，不要尝试去猜测  而是尽量找一种，最好是唯一一种明显的解决方案（如果不确定，就用穷举法） |  |
| Although that way may not be obvious at first unless you're Dutch. |  |  |
| Now is better than never.  Although never is often better than \*right\* now. | 做也许好过不做，但不假思索就动手还不如不做（动手之前要细思量） |  |
| If the implementation is hard to explain, it's a bad idea.  If the implementation is easy to explain, it may be a good idea. | 如果你无法向人描述你的方案，那肯定不是一个好方案；反之亦然（方案测评标准） |  |
| Namespaces are one honking great idea -- let's do more of those! |  |  |

**代码行**

代码行的长度应小于等于79以保证编辑器窗口中可以看到整行代码。格式可以为：

def function\_name(

parameter\_0, parameter\_0, parameter\_0,

parameter\_0, parameter\_0, parameter\_0):

function body

#### 2.1.3数学概念

**位置实参：**

调用函数时，必须将函数调用的每个实参关联到函数定义中的一个形参。

**关键字实参：**

传递给名称的名称-值对。

**任意数量的位置实参：**

在最后一个形参加一个\*，创建一个空元组以放置收到的所有值。

**任意数量的关键字实参：**

在参数上加2个\*，创建一个空字典以放置收到的所有名称-值对。

**序列化(Serialization)**:

将对象态信息（如变量）转换为可以存储或传输的形式的过程。在序列化期间，对象将其当前状态写入到临时或持久性存储区。以后，可以通过从存储区中读取或反序列化对象的状态，重新创建该对象。

**反序列化（unpicking）**：

把变量内容从序列化的对象重新读到内存里。

**自顶向下的设计**

从顶层开始分解问题为更小的问题进行求解

**自底向上的设计**

从底层模块开始一个一个进行测试

**敏捷开发**

把软件功能模块化，分成多个相互联系、独立运行的小项目，软件一直处于可使用状态，以人为核心、迭代、循序渐进，自底向上逐步有序。

好的程序员会使用多种程序设计方法：螺旋、分阶段、敏捷开发等

类 某种类型集合的描述

对于整型和浮点型非零值均可视为TRUE，bool类型仅仅是一类特殊的整数

函数可以有两个返回值

**切比雪夫函数:**

如果Λ(n)表示[曼戈尔特函数](https://baike.baidu.com/item/%E6%9B%BC%E6%88%88%E5%B0%94%E7%89%B9%E5%87%BD%E6%95%B0)：



则下面函数：



和



称为切比雪夫函数。它是[切比雪夫](https://baike.baidu.com/item/%E5%88%87%E6%AF%94%E9%9B%AA%E5%A4%AB)为了证明[素数定理](https://baike.baidu.com/item/%E7%B4%A0%E6%95%B0%E5%AE%9A%E7%90%86)而给出的，是重要的数论函数。函数(1)与素数个数函数π(x)有十分密切的联系。事实上，素数定理：



等价于θ(x)～x或Ψ(x)～x(x→∞)，并且可由[算术基本定理](https://baike.baidu.com/item/%E7%AE%97%E6%9C%AF%E5%9F%BA%E6%9C%AC%E5%AE%9A%E7%90%86)推出关于Ψ(x)的一个重要性质，即：



上式使函数Ψ(x)与[对数函数](https://baike.baidu.com/item/%E5%AF%B9%E6%95%B0%E5%87%BD%E6%95%B0)建立了简单的联系，从而为证明素数定理和研究[素数分布](https://baike.baidu.com/item/%E7%B4%A0%E6%95%B0%E5%88%86%E5%B8%83)奠定了基础。

切比雪夫多项式是一个正交多项式序列Ti(x),—个n次多项式可以表示为多个切比雪夫多项式的加权和。在NumPy中，使用Chebyshev类表示由切比雪夫多项式组成的多项式p(x):



Ti(x)多项式可以通过Chebyshev.basis(i)获得。通过多项式类的convert()方法可以在不同类型的多项式之间相互转换，转换的目标类型由kind参数指定。

切比雪夫多项式的根被称为切比雪夫节点，可以用于多项式插值。相应的插值多项式能最大限度地降低龙格现象，并且提供多项式在连续函数的最佳一致逼近。

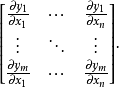
**龙格现象:**

在计算方法中，有利用多项式对某一函数的近似逼近，这样，利用多项式就可以计算相应的函数值。例如，在事先不知道某一函数的具体形式的情况下，只能测量得知某一些分散的函数值。这样，利用已经测的数据，应用待定系数法便可以求得一个多项式函数f（x）。应用此函数就可以计算或者说预测其他日期的气温值。一般情况下，多项式的次数越多，需要的数据就越多，而预测也就越准确。

龙格在研究多项式插值的时候，发现有的情况下，并非取节点（日期数）越多多项式就越精确。著名的例子是f（x）=1/(1+25x^2).它的插值函数在两个端点处发生剧烈的波动，造成较大的误差。究其原因，是舍入误差造成的。

**雅克比矩阵:**

假设某函数从 映到  ， 其雅可比矩阵是从  到  的线性映射，其重要意义在于它表现了一个多变数向量函数的最佳线性逼近。因此，雅可比矩阵类似于单变数函数的导数。 假设  是一个从n维欧氏空间映射到到m维欧氏空间的函数。这个函数由m个实函数组成：  。这些函数的偏导数(如果存在)可以组成一个m行n列的矩阵，这个矩阵就是所谓的雅可比矩阵：



此矩阵用符号表示为：



，或者





这个矩阵的第 i行是由梯度函数的[转置](https://baike.baidu.com/item/%E8%BD%AC%E7%BD%AE)表示的

如果p是



中的一点，F在 p点可微分，根据高等微积分，



是在这点的[导数](https://baike.baidu.com/item/%E5%AF%BC%E6%95%B0)。在此情况下，



这个线性映射即F在点p附近的最优线性逼近，也就是说当x足够靠近点p时，我们有



卷积:

卷积是两个变量在某范围内相乘后求和的结果。如果卷积的变量是序列x(n)和h(n)，则卷积的结果



，

其中星号\*表示卷积。当时序n=0时，序列h(-i)是h(i)的时序i取反的结果；时序取反使得h(i)以纵轴为中心翻转180度，所以这种相乘后求和的计算法称为卷积和，简称卷积。另外，n是使h(-i)位移的量，不同的n对应不同的卷积结果。

如果卷积的变量是函数x(t)和h(t)，则卷积的计算变为



，

其中p是积分变量，积分也是求和，t是使函数h(-p)位移的量，星号\*表示卷积。

**插值**: https://www.cnblogs.com/WhyEngine/p/4020294.html

在离散数据的基础上补插连续函数，使得这条连续曲线通过全部给定的离散数据点。插值是离散函数逼近的重要方法，利用它可通过函数在有限个点处的取值状况，估算出函数在其他点处的近似值。插值可以用于填充图像变换时像素之间的空隙。插值，拟合，逼近是数值分析的三大基础工具，通俗意义上它们的区别在于：插值是已知点列并且完全经过点列；拟合是已知点列，从整体上靠近它们，逼近是已知曲线，或者点列，通过逼近使得构造的函数无限靠近它们。科学和工程问题可以通过诸如采样、实验等方法获得若干离散的数据，根据这些数据，我们往往希望得到一个连续的函数（也就是曲线）或者更加密集的离散方程与已知数据相吻合。

假定[区间](https://baike.baidu.com/item/%E5%8C%BA%E9%97%B4)[a,b]上的[实值函数](https://baike.baidu.com/item/%E5%AE%9E%E5%80%BC%E5%87%BD%E6%95%B0)f(x)在该区间上 n+1个互不相同点x0,x1,……,xn 处的值是f (x0),……f(xn)，要求估算f(x)在[a,b]中某点x\*的值。基本思路是，找到一个函数P(x)，在x0,x1,……,xn的节点上与f(x)函数值相同(有时，甚至一阶导数值也相同)，用P(x\*)的值作为函数f(x\*)的近似。

其通常的做法是：在事先选定的一个由简单函数构成的有n+1个参数C0，C1，……Cn的函数类Φ(C0,C1,……Cn)中求出满足条件P(xi)=f(xi)(i=0,1,…… n)的函数P(x)，并以P()作为f()的估值。此处f(x)称为被插值函数，x0,x1,……,xn称为插值结(节)点，Φ(C0,C1,……Cn)称为插值函数类，上面等式称为插值条件，Φ(C0,C1,……Cn)中满足上式的函数称为插值函数，R(x)= f(x)－P(x)称为插值余项。当估算点属于包含x0,x1,……,xn的最小闭区间时，相应的插值称为内插，否则称为外插。

**样条插值** （https://www.cnblogs.com/WhyEngine/p/4020294.html）

样条方程是一类分段光滑、并且在各段交接处也有一定光滑性的函数。样条一词来源于工程绘图人员为了将一些指定点连接成一条光顺曲线所使用的工具，即富有弹性的细木条或薄钢条。由这样的样条形成的曲线在连接点处具有连续的坡度与曲率。分段低次多项式、在分段处具有一定光滑性的函数插值就是模拟以上原理发展起来的，它克服了高次多项式插值可能出现的振荡现象，具有较好的数值稳定性和收敛性，由这种插值过程产生的函数就是多项式样条函数。

分段插值

将被插值函数f〔x〕的插值节点 由小到大 排序，然后每对相邻的两个节点为端点的区间上用m 次多项式去近似f〔x〕.

**分段线性插值**

将插值区间分成若干个小的区间，在每个小区间进行线性插值，然后相互连接，用连接相邻节点的折线逼近被插函数

外推运算

加密点?

## 3资料与经验

### 3.1经验

先用pycharm而后用vim 或emacs编译。

python学习手册（作业很重要），python编程从入门到实践

国内的视频教育网站不好

要深入源码，了解模块功能。进行有挑战性的实践，把它拆分成若干个小问题。

### 3.2资料

luozixun 《如何成为一名优秀的设计师》

https://www.codecademy.com ，https://www.codewars.com https://www.codeschool.com

花费大量的时间看书、看别人的博客，同时和别人交流刷新知识，多实践。对问题多刨根问底，找到最高的写法效率。

<http://www.runoob.com/python/python-variable-types.html>

python官方教程https://docs.python.org/3/tutorial/index.html

廖雪峰python教程https://www.liaoxuefeng.com/wiki/0014316089557264a6b348958f449949df42a6d3a2e542c000

安装模块网址：https://docs.python.org/3/installing/index.html

https://pypi.org/

安装模块

https://pip.pypa.io/en/stable/reference/pip\_install/#hash-checking-mode

同时安装python2和python3的设定

<https://www.cnblogs.com/zhengyihan1216/p/6011640.html>

#### 安装和更新pip

https://pip.pypa.io/en/stable/installing/