Python

数据嗨客

python™

Python语言是一种面向对象、动态数据类型的解释型语言，是数据分析师首选编程语言之一.

符号标记

x | 一个变量，其值为2.　s | 字符串（string）对象.
l,L | 列表（list）对象.　t | 元组（tuple）对象.
e,E | 集合（set）对象.　d | 字典（dict）对象.

基本操作

x = 2 | 定义一个新的变量x，其值为2.
print | 打印输出.
| 行内注释.
"""" """" | 多行注释.
? | 内省，显示对象的通用信息.
?? | 内省，显示出大部分函数的源代码.
help() | 显示一个对象的帮助文档.
%timeit | 魔术命令，计算语句的平均执行时间.

数据类型及相互转换

type(x) | 查看变量x的数据类型.
int(x) | 将变量x的数据类型转换为整型.
float(x) | 将变量x的数据类型转换为浮点型.
str(x) | 将变量x的数据类型转换为字符串.
bool(x) | 将变量x的数据类型转换为布尔型.
isinstance(x, float) | 检测变量x是否为浮点型，返回一个布尔型数值.

算术运算符

x + 5 | 加，计算结果为7.
x - 5 | 减，计算结果为-3.
x * 5 | 乘，计算结果为10.
x / 5 | 除，Python 2.x版本的计算结果为0，Python 3.x版本的计算结果为0.4.
x ** 2 | 幂运算，即 x^2，计算结果为4.

布尔型

False None 0 " " () [] {} | False值.
and | 等价于"&"，表示"且".
or | 等价于"|"，表示"或".
not | 表示"非".

字符串

s = u" " | 定义Unicode字符串.
s = r" " | 定义原始字符串，避免字符串中的字符转义，在正则表达式中经常使用到.
s = "cookdata" | 定义值为"cookdata"的变量s.
len(s) | 返回s的字符个数8.
s.lower() | 将字符串s中的字母全部转换为小写.
s.upper() | 将字符串s中的字母全部转换为大写.
s.capitalize() | 将字符串s中的首个字符转换为大写，其余部分转换为小写.
s.replace('k', 'l') | 使用字符"l"替换掉s中所有的字符"k"，返回结果为cooldata.
s.strip() | 去除掉s最前面和最后面的空格.
s.split("\t") | 使用制表符"\t"分割字符串s.
'%s is No.%d' %(s, 1) | 取出s的值和数值1依次放入字符串%s is No.%d的相应位置，返回结果为cookdata is No.1.
'{} is No.{}'.format(s, 1) | 取出s的值和数值1依次放入{}相应位置，返回结果为cookdata is No.1.

列表

l = ['c', 'o', 'o', 'k', 1] | 创建一个包含字符元素c、o、o、k和整数1的列表.
list() | 创建空列表，或将其他数据结构转换为列表.
l[0] | 返回列表的第一个元素，即字符c.
l[-1] | 返回列表的最后一个元素，即1.
l[1:3] | 列表切片，返回包含原列表的第二个元素和第三个元素的列表['o', 'o'].
len(l) | 返回列表的元素个数.
l[::-1] | 将列表进行逆序排列.
l.reverse() | 将列表进行逆序排列.
l.insert(1, 'b') | 在指定的索引位置插入'b'.
l.append() | 在列表末尾添加元素.
l.extend(L) | 等价于"l+L"，将列表L中的元素依次添加到l的末尾.
l.remove() | 删除列表中的某个元素.
l.pop() | 等价于"del l[]"，删除列表中对应索引位置的元素.
" ".join(['c', 'o', 'o', 'k']) | 将列表中的各个字符串元素用空格连接起来并转换为字符串，返回结果为c o o k.

元组和集合

t = ('c','o','o','k') | 创建一个元组t，包含字符元素c、o、o、k.
e = {'c','o','k'} | 创建一个集合e，包含元素c、o和k.
len(t) | 元组t中元素的个数.
tuple() | 创建一个空的元组，或将其他的数据结构转换为元组.
set() | 创建一个空的集合，或将其他的数据结构转换为集合.
t.index() | 返回元素的索引号.
t.count() | 返回元素在元组中出现的次数.
e.add() | 添加元素.
e.discard() | 删除元素，如果元素不在集合中，则不作任何操作.
e.union(E) | 等价于"e | E"，求并集.
e.intersection(E) | 等价于"e & E"，求交集.
e.issubset(E) | 判断e是否为E的子集，若是则返回True.

字典

d = {'ID':122,'name':'li'}
d = dict('ID'=122,'name'='li')
d = dict([('ID',122),('name','li')])
创建字典d，其中键为'ID'和'name'，对应的值分别122和'li'.
d.items() | 返回d中键值对列表，列表中的每一个元素为(key,value)元组.
d.keys() | 返回包含d中所有键的列表.
d.values() | 返回包含d中所有值的列表.
d.has_key('sex') | 判断字典d中是否包含键'sex'，包含则返回True，否则返回False.
d.get('sex','wrong key') | 返回字典d中键'sex'的值，如果d中没有该键，则返回字符串'wrong key'.

布尔比较运算

x == 1 | 判断x是否等于1.
x != 1 | 判断x是否不等于1.
x == 1 and name == 'li' | 等价于(x == 1) & (name == 'li')，判断x是否等于1并且name等于'li'.
x == 1 or name == 'li' | 等价于(x == 1) | (name == 'li')，判断x是否等于1或者name等于'li'.
'c' in l | 判断c是否在列表中.

条件判断和循环语句

```
if condition1:
    statement1
elif condition2:
    statement2
else:
    statement3
```

if语句，判断条件1是否成立，若成立则执行语句1，若不成立再判断条件2是否成立，若成立则执行语句2，若都不成立，则执行语句3.

```
for item in sequence:
    statement
```

for语句，依次从序列（列表或字符串）中选取一个元素进行循环，每次循环都要执行一次语句.

```
while condition:
    statement
```

while语句，判断条件是否成立，若成立则循环执行语句直到判断条件不成立.

range(5) | 产生一个从0到5且间隔为1的整数列表[0,1,2,3,4].
break | 从最内层for循环或while循环中跳出.
continue | 继续执行下一次循环.
pass | 占位符，不执行任何动作.

enumerate() 和 zip()

for i,item in enumerate(l) | 在每一次循环时取出索引号和相应的值分别赋给i和item.
for id,name in zip(id_l,name_l)
同时循环两个或多个序列，每一次循环从列表id_l取出一个元素赋给id，且取出列表name_l相同索引位置的元素赋给name.

推导式

L = [item**2 for item in l] | 列表推导式，对l中的每一个元素取平方得到新的列表.
S = {item**2 for item in l} | 集合推导式，对l中的每一个元素取平方得到新的集合.
D = {key:value for key,value in zip(l,k)}
字典推导式，通过zip()函数将两个列表l和k中的元素组成键值对并形成字典.

Yan Xiaodong

Python

数据嗨客

文件读写

读取文件

`f = open(filename,mode)` | 返回一个文件对象f，读文件“model = r”，写文件“model = w”.
`f.read(size)` | 返回包含前size个字符的字符串.
`f.readline()` | 每次读取一行，返回该行字符串.
`f.readlines()` | 返回包含整个文件内容的列表，列表的元素为文件的每一行内容所构成的字符串.
`f.close()` | 关闭文件并释放它所占用的系统资源.

```
with open(”cookdata.txt”, ”r”) as f:
    content = f.readlines()
```
在with主体块语句执行完后，自动关闭文件并释放占用的系统资源.

```
import csv
f = open(”cookdata.csv”, ”r”)
csvreader = csv.reader(f)
content_list = list(csvreader)
```
读取csv文件，并把数据存储为一个嵌套列表（列表的元素仍是一个列表）content_list.

写入文件

```
f.write(s)
print(s, file=f)
```
两种等价的方式，将字符串s写入文件对象f中.

函数

```
def sum(a, b=1):
    return a+b
```
定义求和函数sum()，该函数要求输入位置参数a，带默认值的参数b为可选参数，其默认值为1，函数返回结果为a+b的计算结果.

`sum(1, b=10)` | 执行sum()函数，返回结果为11.
`def sum(*args, **kwargs)` | 不定长参数，*args接收包含多个位置参数的元组，**kwargs接收包含多个关键字参数的字典.
`obj.methodname` | 一个方法是一个“属于”对象并被命名为obj.methodname的函数.

map() 和 lambda

`map(func, sequence)` | 将函数依次作用在序列的每个元素上，把结果作为一个新的序列返回.
`lambda a, b:a+b` | 匿名函数，正常函数定义的语法糖，a和b为输入参数，a+b为函数主体和返回的值.

模块

`import module as alias` | 导入模块，并取一个别名，使用alias.func即可调用模块内的函数.
`from module import *` | 导入模块中的所有函数，直接使用函数名即可调用模块内函数.
`from module import func1, func2` | 导入模块中的部分函数，使用func1可直接调用该函数.

面向对象的类

```
class Athlete(object):
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def capitalize_name(self):
        return self.name.capitalize()
```
使用关键字class定义Athlete类，该类继承object类，初始化变量为self.name和self.age，并定义类的capitalize_name方法将self.name的首字母变成大写.

`a = Athlete(’james’, ’23’)` | 创建实例a.
`a.name` | 返回a的name属性，返回‘james’.
`a.age` | 返回a的age属性，返回23.
`a.capitalize_name()` | 调用a的实例方法capitalize_name，返回‘James’.
`isinstance(a, Athlete)` | 判断a是否是Athlete类的实例.

编码和解码

ASCII | 基于拉丁字母的一套电脑编码系统，不包含中文、日文等非英语字符.

GBK | GBK兼容ASCII，同时收录中文、日文等，使用两个字节编码一个汉字.

Unicode | 收录超过十万个字符，统一了所有语言文字的标准编码集，包括UTF-8和UTF-16两种实现方式.

`s = u’中文’` | 定义Unicode字符串‘中文’.
`s.encode(’utf-8’)` | 使用UTF-8编码集将Unicode字符串s编码为str字符串.

```
s = ’中文’
s.decode(’utf-8’)
```
定义str字符串‘中文’，并解码为Unicode字符串.

```
import chardet
chardet.detect(s)
```
检测字符串的编码方式.

异常处理

语法错误 | Syntax Errors，代码编译时检测到的错误.

异常 | Exceptions，代码运行时检测到的错误，如类型错误（TypeError）、数值错误(ValueError)、索引错误（IndexError）和属性错误（AttributeError）等.

```
try:
    statement
except:
    pass
```
先尝试运行try部分的statement语句，如果能够正常运行，则跳过except部分，如果运行出现错误，则跳过错误代码运行except部分的pass语句，“放过”错误.

```
try:
    statement
except Exception, e:
    print ”Error Happened: %s” %e
```
指定要处理的运行时错误类型，如果指定的错误出现，则打印报错信息，e是发生错误时的具体错误信息.

```
try:
    statement1
except (Exception1, Exception2), e:
    statement2 # 发生指定错误时的处理
else:
    statement3 # 正确时运行
finally:
    statement4 # 无论对错都运行
```
try…except…else…finally句式.

抛出异常

`raise Exception(’Oops!’)` | 主动抛出异常Exception，错误提示信息为‘Oops!’.
`assert statement, e` | 若statement语句运行结果为True，则继续运行代码，否则抛出e的错误提示信息.

正则表达式

正则表达式模块 re

```
import re
raw_s = r’\d{17}[\d|x]|\d{15}’
pattern = re.compile(raw_s)
re.search(pattern, s)
```
用于匹配身份证号。
首先使用原始字符串定义正则表达式模式；然后编译原始字符串为正则表达式Pattern对象；最后对整个字符串s进行模式搜索，如果模式匹配，则返回MatchObject的实例，如果该字符串没有模式匹配，则返回None.

`re.search(r’\d{17}[\d|x]|\d{15}’, s)` | 将Pattern编译过程与搜索过程合而为一.
`re.match(pattern, s)` | 从字符串s的起始位置匹配一个模式，如果起始位置匹配不成功，则返回None.
`re.findall(pattern, s)` | 返回一个包含所有满足匹配模式的子串的列表.
`re.sub(pattern, repl, s)` | 使用替换字符串repl替换匹配到的子字符串.
`re.split(pattern, s)` | 利用满足匹配模式的子串将字符串s分隔开，并返回一个列表.

日期处理

```
from datetime import datetime
format = ”%Y-%m-%d %H:%M:%S”
```
指定日期格式，如“2017-10-01 13:40:00”.
```
date_s = datetime.strptime(s,format)
```
将日期字符串按照指定日期格式转换为datetime类.
`date_s.year` | 获取日期字符串中的年份.
`date_s.month` | 获取日期字符串中的月份.
`datetime.now()` | 获取现在的日期和时间.

Yan Xiaodong