Table of Contents

“加入条件”：

1. 安装anaconda
2. 自备笔记本
3. 遵循“三位一体“
4. 一颗化繁为简的心

“堂吃”：

1. python language (function of types, control structure,)
2. Text File IO by python (dict)
3. EXCEL File IO by python(openpyxl,xlrd)
4. XML File IO by python (bs4, pandas)
5. PDF File IO by python
6. Data base manipulation (Neo4j and Sqlite3)

“外带”：

1. Challenges.
2. SW. Mindset.
3. Primary examples for your application reference.
4. Tool packages for development.

工作经历

GE中央研究院 2010～2017

GE航空 2017～至今

- 2017.Nov~2018.May, SAMC
- ...

联系方式

Email: jay.cn@outlook.com

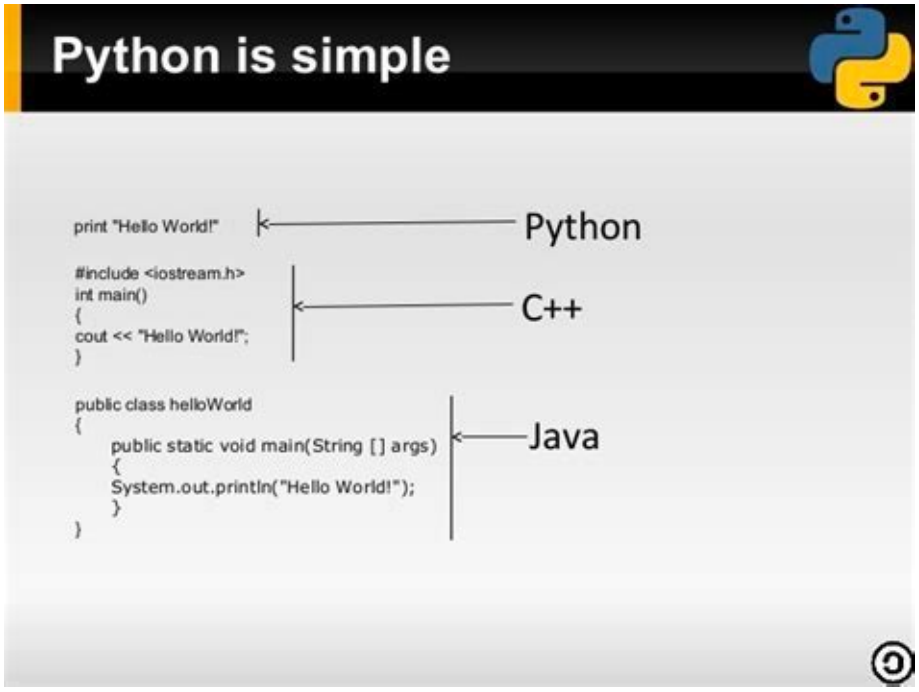
Websit: https://tim9-zero.github.io

技术方向

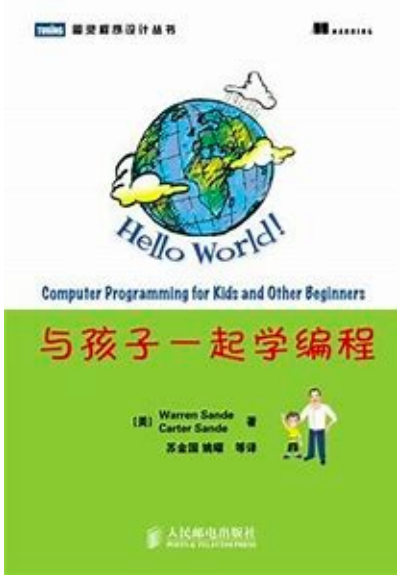
- 无损检测
- 机器视觉
- 制造工艺数字化
- 自适应机器人系统

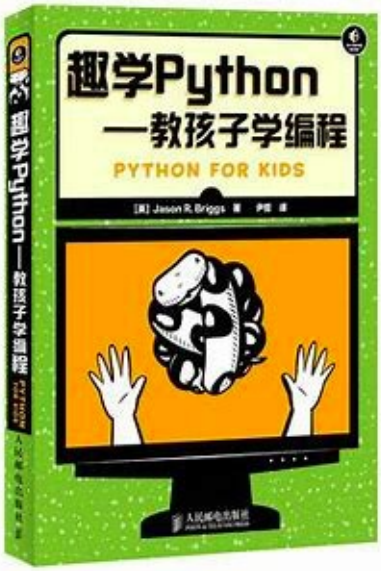
简洁

汇编，C，C++，Java，Javascript，LabView，Matlab，其它设备专用语言



易学





流行

这些年，编程语言的发展进程很快，在商业公司、开源社区两股力量的共同推动下，涌现出诸如Go、Swift这类后起之秀，其中最为耀眼的是Python。

知名开发者网站Stackoverflow撰文指出，从2012至2017年编程语言Python成为开发者使用增长最快的主流编程语言，其中2017年增长率达到了27%，一举超过包括Java、C#、PHP、C++在内的所有同类。另据高盛集团发布的一份《2017调查报告》针对全球数千名高校实习生的调查中，当问到你认为“哪个语言在未来会更重要”时，被调查的**80**、**90**后优秀年轻开发者中**72%**选了**Python**。

语言的使用者是一直被誉为业界上游“源头活水”的开发者，其重要程度从各大科技巨头公司每年例行召开的开发者大会上可见一斑。对于开发者群体而言最重要的事物有两个，一是平台，二就是编程语言。编程语言Python为什么能够获得全球众多开发者的青睐？它的崛起给开发者世界带来了什么变化？

- 成功的一半源于好的开始

在主流编程语言当中，Python并不是一个“新人”，它的历史超过25年，但真正风靡之时却是最近几年，所以“后起之秀”的称呼实至名归。Python的起源是1989年，其发明者荷兰人程序员吉多范罗苏姆受ABC语言的启发计划开发一个新的脚本解释器，由此迈出了Python项目的起点。

Python能够真正风靡的原因之一是有一个好的起点。它的起步很稳，避开了版权纠纷，且搭上了开源运动的顺风车。在那个年代，商业版权一直是热门事件，业界史上第一个软件领域重大官司AT&T和伯克利BSD的Unix版权案打得天昏地暗，该案的结局直接促成了BSD的开源分支、Linux的诞生以及震惊世界的自由软件运动。

Python最初的版权归属是CWI（阿姆斯特丹的国家数学与计算机科研学会），这与吉多早年在该机构工作有关，后来吉多受雇于CNRI（维吉尼亚州的国家创新研究公司），Python权属转移至此。那时自由软件运动已经开始，在CNRI期间发布的1.6至2.1多个版本的Python许可证是一种与GPL并不兼容且类似于BSD的开源许可，CNRI因受到自由软件基金会的压力释放了Python的原许可证，吉多由此掌握了主导权并起草了新的许可证。他改变了原许可证与GPL的不兼容，此举获得了自由软件基金会颁发的自由软件进步奖。再后来吉多和他的团队成立了Python软件基金会，将版权与许可证置于其下。

创始人吉多范罗苏姆的心思缜密与灵活处事为Python最初的发展营造了良好的环境，包括几次权属的转移、起草新的许可证、机智地与自由软件阵营斡旋，最后安全融入开源的大潮。这一切为Python此后十多年里逐渐成长为主流编程语言赢得了契机。

快速成型

- “人生苦短，我用Python”并非一句戏言

Python崛起的原因之二与其本身特点有关，或者说，其长期维护演进形成的独特风格迎合了大多数开发者的口味。在开发者社群流行着一句玩笑“人生苦短，我用Python”（原话为” Life is short, you need Python”），这句看似戏言的话实际上恰恰反映了Python的语言特性与其在开发者心里的价值分量。

除了包涵大多数主流编程语言的优点（面向对象、语法丰富）之外，Python的直观特点是简明优雅、易于开发，用尽量少的代码完成更多工作。尽管Python是一种解释型语言，与传统的编译型语言相比降低了机器执行效率，但是处理器的处理速率与环境速率（比如网络环境）的差异在大多数场景中完全抵消了上述代价；牺牲部分运行效率带来的好处则是提升了开发效率，在跨平台的时候无需移植和重新编译。 所以Python的显著优点在于速成，对于时间短、变化快的需求而言尤为胜任。

Python最强大的地方体现在它的两个外号上，一个叫**“内置电池”**，另一个是**“胶水语言”**。前者的意思是，Python官方本身提供了非常完善的标准代码库，包括针对网络编程、输入输出、文件系统、图形处理、数据库、文本处理等等。代码库相当于已经编写完成打包供开发者使用的代码集合，程序员只需通过加载、调用等操作手段即可实现对库中函数、功能的利用，从而省去了自己编写大量代码的过程，让编程工作看起来更像是在“搭积木”。除了内置库，开源社区和独立开发者长期为Python贡献了丰富大量的第三方库，其数量远超其他主流编程语言，可见Python的语言生态已然相当壮大。

“胶水语言”是Python的另一个亮点。Python本身被设计成具有可扩展性，它提供了丰富的API和工具，以便开发者能够轻松使用包括C、C++等主流编程语言编写的模块来扩充程序。就像使用胶水一样把用其他编程语言编写的模块粘合过来，让整个程序同时兼备其他语言的优点，起到了黏合剂的作用。正是这种多面手的角色让Python近几年在开发者世界中名声鹊起，因为互联网与移动互联时代的需求量急速倍增，大量开发者亟需一种极速、敏捷的工具来助其处理与日俱增的工作，Python发展至今的形态正好满足了他们的愿望。

影响力

- Python的影响

从两个著名编程语言排行网站TIOBE和PYPL的最新数据来看，Java与Python的排名分别位于第1和第5、第1和第2。关于两个网站的排行机制我们不得而知，但从开发者社群的相关评论中可以认为PYPL更能反映编程语言在开发者群体中的流行程度。不论如何，Python的崛起已是毋庸置疑的事实，而它上面的前辈则是常年占据榜单第1，互联网与移动时代的娇子Java。从Stackoverflow和多个开源社区公开的数据来看，Python的用户数量增长很快，在今后两年超过Java成为全球最流行编程语言的可能性非常之高。

值得一提的是，那些颇有影响力的主流编程语言，其背后一般都站着科技巨头公司，比如Java之于甲骨文、C#之于微软、Objective-C之于苹果。Java之所以常年第一是因为其同时还几乎是安卓平台的御用语言，以及受益于Sun时代影响力的眷顾。Python虽曾一度为谷歌使

用，但Go语言问世后随着时间推移或将遇冷。也就是说，**Python**成了没有巨头站队的主流编程语言，那么它的影响力是如何维系的？为什么还能够保持高速成长并形成赶超Java之势？

我们认为这与Python多年来实现较好案例与范用性有关。使用Python开发的知名案例中，包括豆瓣、果壳、知乎、Dropbox、EVE（星战前夜）每一个都是重量级产品，这说明Python语言本身的发展已日臻完善，有着极高的稳定与可靠性保证。第二是Python的应用范围，除了日常工具和脚本之外，还适用于Web程序、GUI开发、操作系统中间件、服务端运维等等，这些年Python的一些第三方库在机器学习、神经网络方面活跃非凡，这也为语言本身的推广和流行加分不少。

最后需要指出的是，Python编程思想包含强烈的黑箱思维，这意味着开发者将愈加重视模块化和流水线式的编程工作，事实上这也是未来主流编程语言的发展趋向。随着计算机语言的演化和开发工具集成功能日趋强大，未来的编程工作将大幅简化。从某种角度看，Python更像是已经“迈入未来”的编程语言，其对开发者群体结构变化，以及新进开发者数量的激增，这些影响都将是深远的。

- VS MATLAB

由于Python语言的简洁性、易读性以及可扩展性，在国外用Python做科学计算的研究机构日益增多，一些知名大学已经采用**Python**来教授程序设计课程。例如卡耐基梅隆大学的编程基础、麻省理工学院的计算机科学及编程导论就使用Python语言讲授。众多开源的科学计算软件包都提供了Python的调用接口，例如著名的计算机视觉库OpenCV、三维可视化库VTK、医学图像处理库ITK。而Python专用的科学计算扩展库就更多了，例如如下3个十分经典的科学计算扩展库：NumPy、SciPy和matplotlib，它们分别为Python提供了快速数组处理、数值运算以及绘图功能。因此Python语言及其众多的扩展库所构成的开发环境十分适合工程技术、科研人员处理实验数据、制作图表，甚至开发科学计算应用程序。

说起科学计算，首先会被提到的可能是MATLAB。然而除了MATLAB的一些专业性很强的工具箱还无法替代之外，MATLAB的大部分常用功能都可以在Python世界中找到相应的扩展库。和MATLAB相比，用Python做科学计算有如下优点：

首先，MATLAB是一款商用软件，并且价格不菲。而Python完全免费，众多开源的科学计算库都提供了Python的调用接口。用户可以在任何计算机上免费安装Python及其绝大多数扩展库。

其次，与MATLAB相比，Python是一门更易学、更严谨的程序设计语言。它能让用户编写出更易读、易维护的代码。

最后，MATLAB主要专注于工程和科学计算。然而即使在计算领域，也经常会遇到文件管理、界面设计、网络通信等各种需求。而Python有着丰富的扩展库，可以轻易完成各种高级任务，开发者可以用Python实现完整应用程序所需的各种功能。

应用举例

- 系统脚本

起源于此

- 游戏

pygame，EVE

- **GUI**

Qt，tk。。

- **web**开发

flask，django

- **IoT**

Raspberry，Arduino，PLC...

- 机器人

ROS

- 大数据分析

Splunk, Dropbox

- 人工智能

人脸识别，字体识别，RL gamer

参考资料

1. python doc
2. pygame
3. Qt
4. flask
5. ROS
6. AI
7. keras
8. 更多可能等你探索

常用函数

- print()
- input()
- len()
- range()
- sorted()
- str()
- int()
- type()

常用类型

- int
- float
- str
- bool

循环控制

- if elif else
- for
- while

常用容器

- list
- tuple
- dict
- set

函数定义

- def
- lambda

类定义

- class something():
- 魔术方法 __*init*__()

第一行代码：

```
print("Hello world")
```

```
name = input("请输入您的名字：")
print("Hello ",name)
```

先上代码，再慢慢聊：

```
info = input("请输入等腰三角形的高度,宽度,表示符号(for example: 40,40,*):").split(",")

try:
    height,width,symbolic = int(info[0]),int(info[1]),info[2]
except :
    print("errors in input parameters!")

dt = (width-1)/(height-1)

for i in range(0,height):

    blank_n =int(dt/2*i)

    line = " "*blank_n+symbolic*int(width-blank_n*2)+" "*blank_n

    print(line)
```

代码先行

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Apr  3 11:27:27 2018

@author: jayhan
"""
poem = '''
Python之禅 by Tim Peters

优美胜于丑陋（Python 以编写优美的代码为目标）。
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）。
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）。
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）。
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）。
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）。
可读性很重要（优美的代码是可读的）。
即便假借特例的实用性之名，也不可违背这些规则（这些规则至高无上）。
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）。
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）。
虽然这并不容易，因为你不是 Python 之父（这里的 Dutch 是指 Guido ）。
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）。
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）。
命名空间是一种绝妙的理念，我们应当多加利用（倡导与号召）！

'''

print(poem)
print(len(poem))

poem_set = set(poem)
print(poem_set)
print(len(poem_set))

en_set = [chr(ord('a')+i) for i in range(26)]+[chr(ord('A')+i) for i in range(26)]+['-'
,'——']
#print(en_set)
poem_set = set([word for word in poem if(word.isalpha() and word not in en_set)])
print(poem_set)
print(len(poem_set))

poem_dic=dict()
for word in poem:
    if word in poem_set:
        val = poem_dic.get(word,0)
        poem_dic[word] = val+1
print(poem_dic)
```

str

- 'capitalize': <method 'capitalize' of 'str' objects>,
- 'casefold': <method 'casefold' of 'str' objects>,
- 'center': <method 'center' of 'str' objects>,
- 'count': <method 'count' of 'str' objects>,
- 'encode': <method 'encode' of 'str' objects>,
- 'endswith': <method 'endswith' of 'str' objects>,
- 'expandtabs': <method 'expandtabs' of 'str' objects>,
- 'find': <method 'find' of 'str' objects>,
- 'format': <method 'format' of 'str' objects>,
- 'format_map': <method 'format_map' of 'str' objects>,
- 'index': <method 'index' of 'str' objects>,
- 'isalnum': <method 'isalnum' of 'str' objects>,
- 'isalpha': <method 'isalpha' of 'str' objects>,
- 'isdecimal': <method 'isdecimal' of 'str' objects>,
- 'isdigit': <method 'isdigit' of 'str' objects>,
- 'isidentifier': <method 'isidentifier' of 'str' objects>,
- 'islower': <method 'islower' of 'str' objects>,
- 'isnumeric': <method 'isnumeric' of 'str' objects>,

-
- 'isprintable': <method 'isprintable' of 'str' objects>,
-
- 'isspace': <method 'isspace' of 'str' objects>,
-
- 'istitle': <method 'istitle' of 'str' objects>,
-
- 'isupper': <method 'isupper' of 'str' objects>,
-
- 'join': <method 'join' of 'str' objects>,
-
- 'ljust': <method 'ljust' of 'str' objects>,
-
- 'lower': <method 'lower' of 'str' objects>,
-
- 'lstrip': <method 'lstrip' of 'str' objects>,
-
- 'maketrans': <staticmethod at 0x109ef7940>,
-
- 'partition': <method 'partition' of 'str' objects>,
-
- 'replace': <method 'replace' of 'str' objects>,
-
- 'rfind': <method 'rfind' of 'str' objects>,
-
- 'rindex': <method 'rindex' of 'str' objects>,
-
- 'rjust': <method 'rjust' of 'str' objects>,
-
- 'rpartition': <method 'rpartition' of 'str' objects>,
-
- 'rsplit': <method 'rsplit' of 'str' objects>,

-
- 'rstrip': <method 'rstrip' of 'str' objects>,
-
- 'split': <method 'split' of 'str' objects>,
-
- 'splitlines': <method 'splitlines' of 'str' objects>,
-
- 'startswith': <method 'startswith' of 'str' objects>,
-
- 'strip': <method 'strip' of 'str' objects>,
-
- 'swapcase': <method 'swapcase' of 'str' objects>,
-
- 'title': <method 'title' of 'str' objects>,
- 'translate': <method 'translate' of 'str' objects>,
- 'upper': <method 'upper' of 'str' objects>,
- 'zfill': <method 'zfill' of 'str' objects>}\)

参考1：https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str

1. read()
2. readline()
3. readlines()
4. write()
5. writelines()
6. Example：info. search

os 模块

This module provides a portable way of using operating system dependent functionality.

这个模块提供了一种方便的使用操作系统函数的方法

- os.remove(‘path/filename’) 删除文件
- os.rename(oldname, newname) 重命名文件
- os.walk() 生成目录树下的所有文件名
- os.chdir('dirname') 改变目录
- os.mkdir/makedirs('dirname')创建目录/多层目录
- os.rmdir/removedirs('dirname') 删除目录/多层目录
- os.listdir('dirname') 列出指定目录的文件
- os.getcwd() 取得当前工作目录
- os.chmod() 改变目录权限
- os.path.basename(‘path/filename’) 去掉目录路径，返回文件名
- os.path.dirname(‘path/filename’) 去掉文件名，返回目录路径
- os.path.join(path1[,path2[,...]]) 将分离的各部分组合成一个路径名
- os.path.split('path') 返回(dirname(), basename())元组
- os.path.splitext() 返回 (filename, extension) 元组
- os.path.getatime\ctime\mtime 分别返回最近访问、创建、修改时间
- os.path.getsize() 返回文件大小
- os.path.exists() 是否存在
- os.path.isabs() 是否为绝对路径
- os.path.isdir() 是否为目录
- os.path.isfile() 是否为文件

文件路径

正是python自带的os模块提供了文件路径相关的所有功能

- file.open(name[,mode[,buffering]])

模式的类型有：

r　　默认只读

w 以写方式打开，如果文件不存在则会先创建，如果文件存在则先把文件内容清空（*truncate the file first*）
a 以追加模式打开 *(*从 *EOF* 开始, 必要时创建新文件*)*用*seek*也无用。打开的文件也是不能读的。
r+ 以读写模式打开，如果文件不存在则报错，文件可读可写，可写到文件的任何位置
w+ 以读写模式打开 *(*参见 *w)*，和*r+*不同的是，它会*truncate the file first*
a+ 以读写模式打开 *(*参见 *a)*，和*r+*不同的是，它只能写到文件末尾
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+)*

- file.read([size]) size未指定则返回整个文件,如果文件大小>2倍内存则有问题.f.read()读到文件尾时返回""(空字串)
- file.readline( )：表示逐行读取,返回字符串
- file.readlines(): 读取所有行,返回字符串列表
- file.readline([size]) 返回包含size行的列表,size 未指定则返回全部行
- file.write() 接收字符串并且写入到文件中
- file.writelines() 接收一个字符串列表作为参数，将他们写入到文件中，换行符不会自动的加入，因此，需要显式的加入换行符
- file.tell() 返回一个整数,表示当前文件指针的位置
- file.seek(偏移量,[起始位置])

用来移动文件指针
偏移量*:*单位*:*比特,可正可负
起始位置*:*

*0-*文件头,默认值

*1-*当前位置

*2-*文件尾

- f.truncate() 清空文件
- file.close() 关闭文件

for line in f: print line #通过迭代器访问

函数介绍

http://www.runoob.com/python/file-methods.html

在指定文件下，列出所有带有特定信息的文件。

功能说明：

path text “someting” --

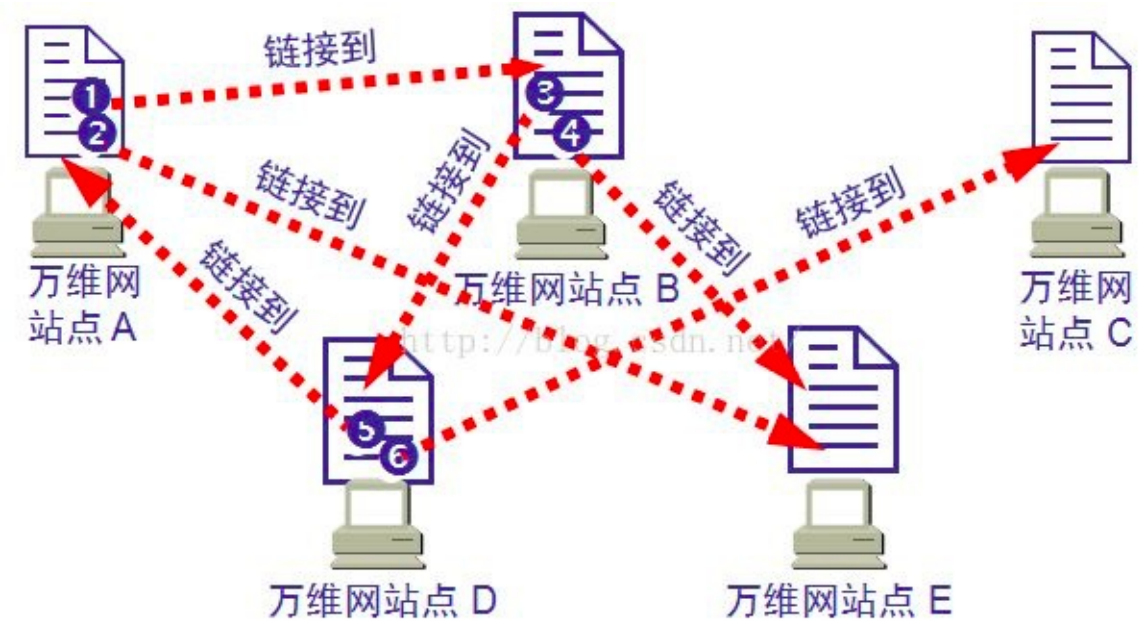
path file ”something“ --

path folder “something“ --

path “something“ --

https://en.wikipedia.org/wiki/World_Wide_Web

https://baike.baidu.com/item/web/150564



微小的需求点亮万亿级的市场："论文分享"

第一个解决方案：

```
<html>
  <head>
    <title></title>
  </head>
  <body>

  </body>
</html>
```

XML是一个理想：结构化数据与结构化文档

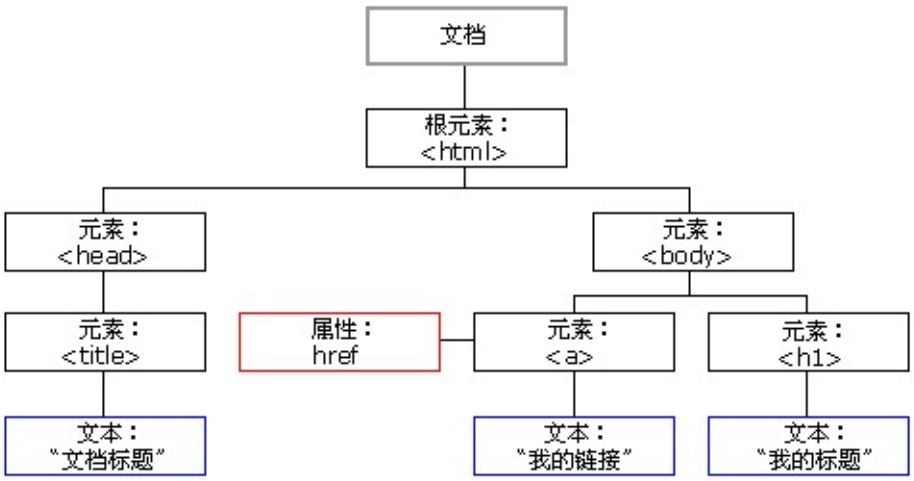
HTML

```
<html>
<head>
<meta charset="utf-8">
<title>文档标题</title>
</head>
<body>

<h1>我的标题</h1>
<a href="https://www.baidu.com">我的链接</a>
<p>我的第一个段落。</p>

</body>
</html>
```

HTML DOM 树



CSS选择器

简单选择器

A:tag

.B:class

#B :id

```
p
{
color:#f00;
}

.class_img
{
background-image:url(img_1.jpg);
}

#id_1
{
border: 1px dash #ff0;
}
```

属性匹配选择器

attr='xx': equal

$= : end with

^=, start with

~=, one of

*=, has this substring

```
p[id='xx']
{
color:#f00;
}
```

组合选择器

A B: and

A,B: or

A>B: and，同时B是A儿子

A+B: 当B是A的下一个兄弟时

A~B:当B有个A兄弟时

```
table td, table th {
  border : 1px solid black;
  padding: 0.5em 0.5em 0.4em;
}

/* All <th>s within <thead>s that are within <table>s */
table thead th {
  color: white;
  background: black;
}

/* All <td>s preceded by another <td>,
   within a <tbody>, within a <table> */
table tbody td + td {
  text-align: center;
}
```

详细内容

BeautifulSoup

1. 元素选择函数

有多种方法可以搜索soup树：

1.标签.children/descendant/parent/parents/next_sibling(s)/previous_sibling(s)/.contents

2.find()/...

3.select()

其中select方法与js比较相似，具有通用性。使用方法举例如下：

soup.select('div') 所有名为 的元素

soup.select('#author') 带有 id 属性为 author 的元素 soup.select('.notice') 所有使用 CSS class 属性名为 notice 的元素

soup.select('div span') 所有在 元素之内的元素 soup.select('div > span') 所有直接在 元素之内的元素，中间没有其他元素

soup.select('input[name]') 所有名为<input>，并有一个 name 属性，其值无所谓的元素

soup.select('input[type="button"]') 所有名为<input>，并有一个 type 属性，其值为 button 的元素

详细信息

2. 属性获取函数

.get()

3. tag对象的dom方法

tag.name,

tag['attr'],

tag.string

Python XML解析

什么是XML？

XML 指可扩展标记语言（e**X**tensible**M**arkup**L**anguage）。 你可以通过本站学习XML教程

XML 被设计用来传输和存储数据。

XML是一套定义语义标记的规则，这些标记将文档分成许多部件并对这些部件加以标识。

它也是元标记语言，即定义了用于定义其他与特定领域有关的、语义的、结构化的标记语言的句法语言。

python对XML的解析

常见的XML编程接口有DOM和SAX，这两种接口处理XML文件的方式不同，当然使用场合也不同。

python有三种方法解析XML，SAX，DOM，以及ElementTree:

1.SAX (simple API for XML)

python 标准库包含SAX解析器，SAX用事件驱动模型，通过在解析XML的过程中触发一个个的事件并调用用户定义的回调函数来处理XML文件。

2.DOM(Document Object Model)

将XML数据在内存中解析成一个树，通过对树的操作来操作XML。

3.ElementTree(元素树)

ElementTree就像一个轻量级的DOM，具有方便友好的API。代码可用性好，速度快，消耗内存少。

注：因DOM需要将XML数据映射到内存中的树，一是比较慢，二是比较耗内存，而SAX流式读取XML文件，比较快，占用内存少，但需要用户实现回调函数（handler）。

参考1: 详细内容

参考2: http://lxml.de/tutorial.html

本教程使用lxml模块的ElementTree类进行入门教学。

PC上，数据组织与处理的常用形式：csv和Excel

- 限定符号

delimiter

quotchar

newline

- 函数

csv.Reader()

csv.Writer()

csv.DictReader

csv.DictWriter()

obj.readrow()

obj.writerow

详细内容

openpxyl(.xlsx)

READ

##get workbook

load_workbook(filename="sample.xlsx")

##get worksheet

wb.sheetnames

wb[ws_names]

##get single data

ws[' ']

##get row

table = ws.values

next(table)

##get table

table = ws.values

list(table)

WRITE

##get workbook

Workbook()

##get worksheet

wb.active

wb.create_sheet()

wb.sheetnames & wb['*sheet_name']*

wb.worksheets[]

##save data in worksheet

ws['xx'] =?

ws.append([,,])

##save excel

wb.save("")

xrld(.xls)

READ

wb = xlrd.open_workbook(fileName) #取得workbook对象

table_by_index = wb.sheets()[by_index] #通过序号获取table对象

table_by_name = wb.sheet_by_name(by_name)#通过表格名获取table对象

cell = table.cell('A1')

nrows = table.nrows #行数

cols = table.ncols #列数

colnames = table.row_values(rowIndex) #整行数据

参考资料

openpyxl

文档结构与操作

- docx.document
- doc.add_heading()
- doc.add_paragraph()
- prg.add_run()
- doc.add_breaker()
- doc.add_picture()

PDF结构

pages()

page.text()

对于PDF，Word，以及一些不易处理与控制系统资源，我们还有一套万能方法-GUI自动化！

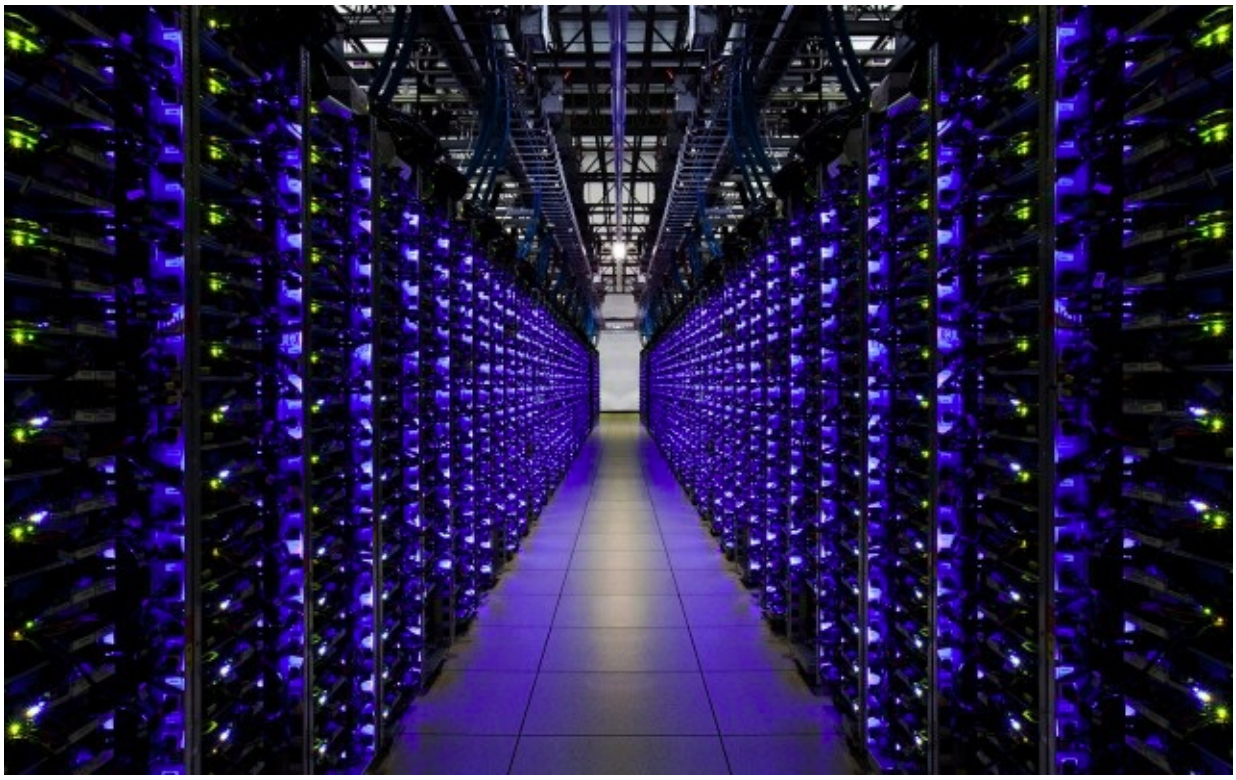
思路如下：

通过记录与控制鼠标键盘，来模拟人工操作，从而直接在系统界面下进行自动化工作。

工具准备：

pyautogui，pyhook

数据库是什么？



SQL

SQL 是用于访问和处理数据库的标准的计算机语言。

什么是 SQL？

- SQL 指结构化查询语言
- SQL 使我们有能力访问数据库
- SQL 是一种 ANSI 的标准计算机语言

编者注：ANSI，美国国家标准化组织

SQL 能做什么？

- SQL 面向数据库执行查询
- SQL 可从数据库取回数据
- SQL 可在数据库中插入新的记录
- SQL 可更新数据库中的数据
- SQL 可从数据库删除记录
- SQL 可创建新数据库
- SQL 可在数据库中创建新表
- SQL 可在数据库中创建存储过程
- SQL 可在数据库中创建视图
- SQL 可以设置表、存储过程和视图的权限

SQL 是一种标准 - 但是...

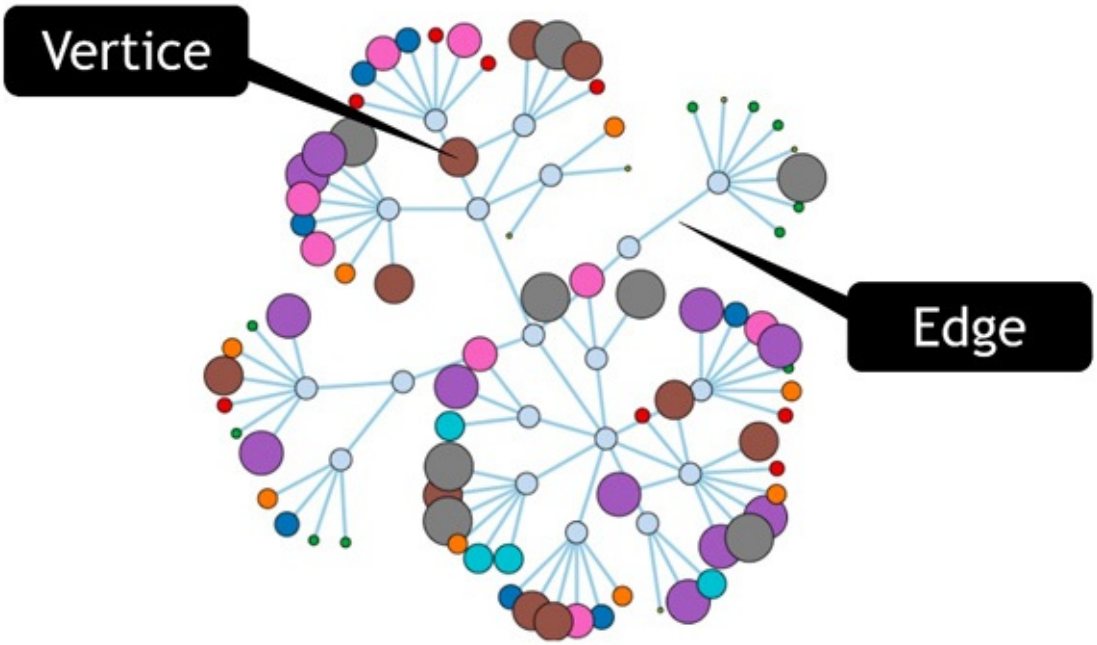
SQL 是一门 ANSI 的标准计算机语言，用来访问和操作数据库系统。SQL 语句用于取回和更新数据库中的数据。SQL 可与数据库程序协同工作，比如 MS Access、DB2、Informix、MS SQL Server、Oracle、Sybase 以及其他数据库系统。

不幸地是，存在着很多不同版本的 SQL 语言，但是为了与 ANSI 标准相兼容，它们必须以相似的方式共同地来支持一些主要的关键词（比如 SELECT、UPDATE、DELETE、INSERT、WHERE 等等）。

注释：除了 SQL 标准之外，大部分 SQL 数据库程序都拥有它们自己的私有扩展！

NonSql(Neo4j)

关系成为第一对象



详细内容

SQL的两个对象：

连接对象(connection): 与外部的文件系统建立关系

游标对象(cursor): 向表格中的指定位置数据

SQL常用数据类型：

text ：可变长度的字符串。最多 2GB 字符数据。

real：从 -3.40E + 38 到 3.40E + 38 的浮动精度数字数据

datetime: 日期和时间的组合。格式：YYYY-MM-DD HH:MM:SS

更多详细内容

SQL常用命令使用方法

(1) 数据记录筛选：
sql="select * from 数据表 **where** 字段名=字段值 **order by** 字段名 **[desc]"**
sql="select * from 数据表 where 字段名 like '%字段值%' order by 字段名 [desc]"
sql="select top 10 * from 数据表 where 字段名 order by 字段名 [desc]"
sql="select * from 数据表 where 字段名 in ('值1','值2','值3')"
sql="select * from 数据表 where 字段名 between 值1 and 值2"
(2) 更新数据记录：
sql="update 数据表 set 字段名=字段值 where 条件表达式"
sql="update 数据表 set 字段1=值1,字段2=值2 …… 字段n=值n where 条件表达式"
(3) 删除数据记录：
sql="delete from 数据表 where 条件表达式"
sql="delete from 数据表" (将数据表所有记录删除)
(4) 添加数据记录：
sql="insert into 数据表 **(**字段**1,**字段**2,**字段**3 …) values (**值**1,**值**2,**值**3 …)"**
sql="insert into 目标数据表 select * from 源数据表" (把源数据表的记录添加到目标数据表)
(5) 数据记录统计函数：
AVG(字段名) 得出一个表格栏平均值
COUNT(*|字段名) 对数据行数的统计或对某一栏有值的数据行数统计
MAX(字段名) 取得一个表格栏最大的值
MIN(字段名) 取得一个表格栏最小的值
SUM(字段名) 把数据栏的值相加

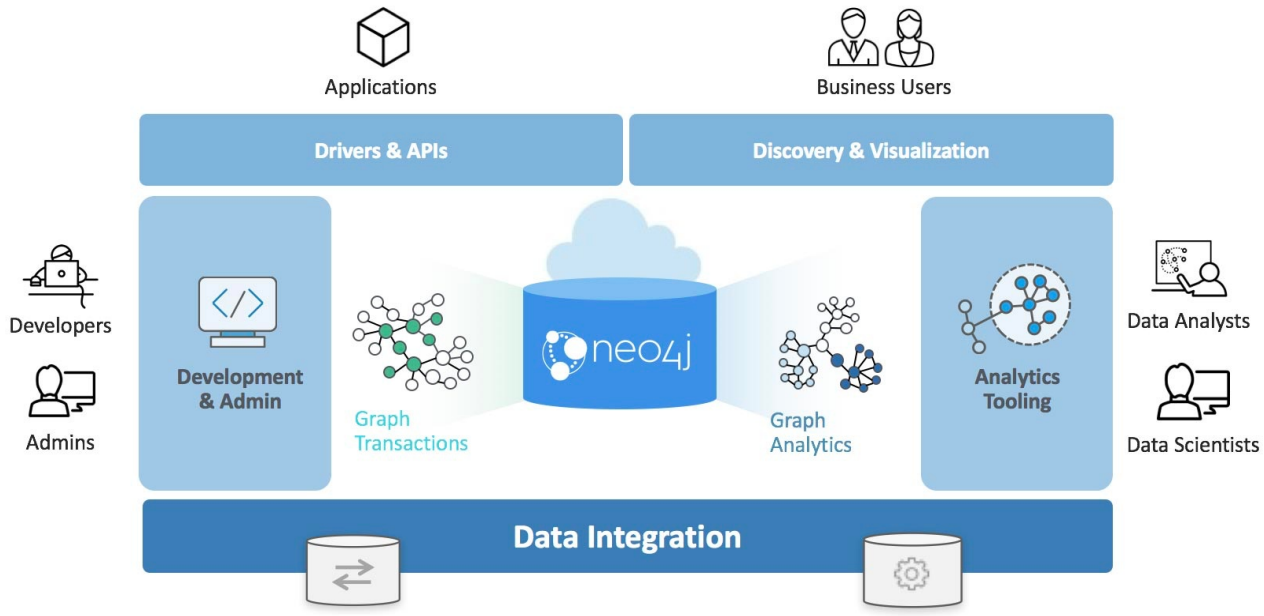
引用以上函数的方法：
sql="select sum(字段名**) as** 别名 **from** 数据表 **where** 条件表达式**"**
set rs=conn.excute(sql)
用 rs("别名") 获取统的计值，其它函数运用同上。
(5) 数据表的建立和删除：
CREATE TABLE 数据表名称**(**字段**1** 类型**1(**长度**),**字段**2** 类型**2(**长度**) ……)**
例：CREATE TABLE tab01(name varchar(50),datetime default now())
DROP TABLE 数据表名称 (永久性删除一个数据表)

1. 记录集对象的方法：
   rs.movenext 将记录指针从当前的位置向下移一行
   rs.moveprevious 将记录指针从当前的位置向上移一行
   rs.movefirst 将记录指针移到数据表第一行
   rs.movelast 将记录指针移到数据表最后一行
   rs.absoluteposition=N 将记录指针移到数据表第N行
   rs.absolutepage=N 将记录指针移到第N页的第一行
   rs.pagesize=N 设置每页为N条记录
   rs.pagecount 根据 pagesize 的设置返回总页数
   rs.recordcount 返回记录总数
   rs.bof 返回记录指针是否超出数据表首端，true表示是，false为否
   rs.eof 返回记录指针是否超出数据表末端，true表示是，false为否
   rs.delete 删除当前记录，但记录指针不会向下移动
   rs.addnew 添加记录到数据表末端
   rs.update 更新数据表记录

Sqlite3

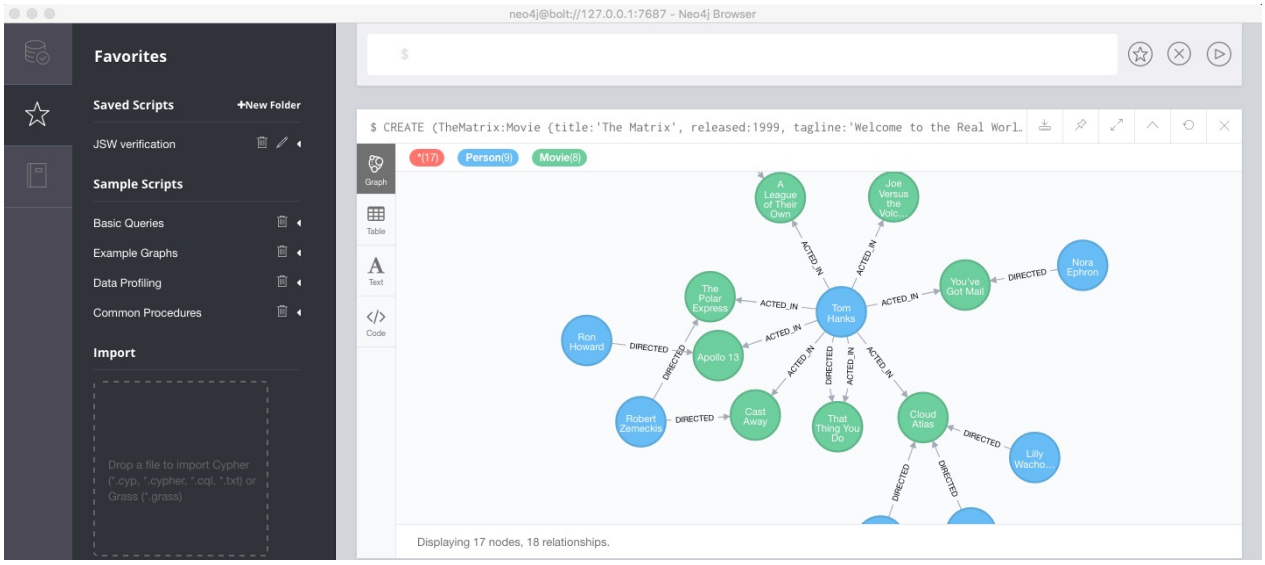
python accompanying

sample codes



neo4j basic

- https://neo4j.com/product/?ref=hro
- learning by practice with Neo4j browser



py2neo - a python API for neo4j

- python code practice

Modules

- beautifulSoup4
- lxml
- csv
- openpyxl
- xlrd
- PyPDF2 （conda install -c conda-forge PyPDF2）
- docx (conda install -c conda-forge python-docx)
- pyhook
- pyautogui
- sqlite3
- py2neo

Websites