# 打开pydoc

(1)Cmd:

Python -m pydoc -p 12345

(2)浏览器中输入：<http://localhost:12345/>

# 字符串操作

## 判断一个字符串是否有子字符串

1. 方法一：

if childStr in str:

（2）方法二：

string.index(’world‘) ！= -1 #-1代表没有

## 拼接

例子：

li=[`kang`, `chi`,`ni`,`hao`]

str=` `.join(li)

#str是：kang chi ni hao

## 拆分

例子：

str=` kang chi ni hao `

li=str.split(` `)

# li=[`kang`, `chi`,`ni`,`hao`]

# Openpyxl 操作EXCEL

## 打开excel wb= load\_workbook(filename, read\_only=False, keep\_vba=KEEP\_VBA, data\_only=False, keep\_links=True)

返回rtype : class: openpyxl.workbook.Workbook

例子:wb=openpyxl.load\_workbook(r‘D:\test.txt’)

## 取得excel所有sheet名字 wb.get\_sheet\_names()

返回：一个包含所有sheet名字的List

例子：for sheetName in wb.get\_sheet\_names():

## 返回某个sheet wb[sheetName]

返回：一个sheet对象

例子：ws=wb[‘sheet1’]

## 删除某个sheet wb.remove（sheet对象）

例子：wb.remove(wb[‘sheet1’])

## 获取一个sheet的最大行 sheet.max\_row

## 获取sheet中的一个cell sheet.**cell**(self, row, column, value=None)或者sheet[‘D5’]

返回：一个cell对象

例子：sheet.cell(row=i+1,column=13).value='haha'

或者：sheet[‘D5’]

## 取得或者设置一个单元格的值 cell.**value**

例子：sheet.cell(row=i+1,column=13).value='haha'

## 修改完成后保存到文件 wb.save(filename)

## 合并格单元说明

对于合并格子单元，读取的时候，只有左上角那个会有值，其它的都是None

# Os 操作文件和目录

## [os.chdir(path)](https://www.runoob.com/python/os-chdir.html) 改变当前工作目录

## os.listdir(rootPath) 取得一个目录下所有文件和目录名字

## os.path.isdir(filename) 判断是否是目录

## os.walk(dir) 生成目录树

轮巡目录dir下面包含的所有目录和文件，最后会生成一系列元组（dirpath,dirnames,filenames）

其中，dirpath是目录，dirnames是该目录下的子目录（不包含孙目录）列表，filenames是该目录下所有子文件列表.

常用用法：

(1)

for root,dirs,files in os.walk(rootdir):  
 print(root)  
 print(dirs)  
 print(files)

1. 查询目录下所有文件

def getAllFiles(dir,filesList):  
 for root,dirs,files in os.walk(dir):  
 for file in files :  
 filesList.append(os.path.join(root,file))

# 文件读写

## open(filename，mode=’r’，encoding=’’) 读取一个文件

参数mode：'r' open for reading (default)  
'w' open for writing, truncating the file first  
'x' create a new file and open it for writing  
'a' open for writing, appending to the end of the file if it exists  
'b' binary mode  
't' text mode (default)  
'+' open a disk file for updating (reading and writing)  
'U' universal newline mode (deprecated)

参数encoding: 可取值“utf-8”“ gb18030”等等。

返回：一个file对象

**例子：helloFile = open('C:\\Users\\*your\_home\_folder*\\hello.txt')**

## read() 将文件中的内容读入内存

例子：**helloFile = open('C:\\Users\\*your\_home\_folder*\\hello.txt')**

**StrRead=helloFile.read()**

## readlines() 一行一行读取

返回：所有行组成的List.

## write() 写入文件

# 日期和时间

## 日期时间 datetime

包含**datetime类，date类，time类.**

### 比较大小

\_\_eq\_\_(self, value,)

返回：self==value.

**\_**\_ge\_\_(self, value)

返回：self>=value.

\_gt\_\_(self, value, /)

返回：self>value.

### 两个datetime相减

返回：timedelta对象，该对象包含：

days

Number of days.

microseconds

Number of microseconds (>= 0 and less than 1 second).

seconds

Number of seconds (>= 0 and less than 1 day).

### 间隔时间转化为秒 timedelta.total\_seconds(...)

返回：秒。可以为负数。

### 根据时间戳返回datetime utcfromtimestamp()

返回：datetime.datetime

例子：datetime.datetime.utcfromtimestamp(time.time())

返回：datetime.datetime(2017, 3, 22, 8, 29, 7, 654272)

### 格式化输入datetime.datetime.strptime（）

返回：datetime.datetime

例子：datetime.datetime.strptime('2017-3-22 15:25','%Y-%m-%d %H:%M')

返回：datetime.datetime(2017, 3, 22, 15, 25)

## 方便的时间格式化工具 time

每个时间戳都以自从1970年1月1日午夜（历元）经过了多长时间来表示。

Python 的 time 模块下有很多函数可以转换常见日期格式。如函数time.time()用于获取当前时间戳,

### time.time()

返回：当前时间的时间戳(秒)

例子ticks = time.time()

print "当前时间戳为:", ticks

返回：当前时间戳为: 1459994552.51

### time.localtime(second)

参数second: 时间戳，秒。不带参数表示当前时间

返回：一个struct\_time元组

|  |  |  |
| --- | --- | --- |
| **序号** | **属性** | **值** |
| 0 | tm\_year | 2008 |
| 1 | tm\_mon | 1 到 12 |
| 2 | tm\_mday | 1 到 31 |
| 3 | tm\_hour | 0 到 23 |
| 4 | tm\_min | 0 到 59 |
| 5 | tm\_sec | 0 到 61 (60或61 是闰秒) |
| 6 | tm\_wday | 0到6 (0是周一) |
| 7 | tm\_yday | 1 到 366(儒略历) |
| 8 | tm\_isdst | -1, 0, 1, -1是决定是否为夏令时的旗帜 |

例子：

import time

localtime = time.localtime(time.time())

print "本地时间为 :", localtime

以上实例输出结果：

本地时间为 : time.struct\_time(tm\_year=2016, tm\_mon=4, tm\_mday=7, tm\_hour=10, tm\_min=3, tm\_sec=27, tm\_wday=3, tm\_yday=98, tm\_isdst=0)

### 格式化输出日期 time.strftime(format[, t])

格式化日期

例子：

# 格式化成2016-03-20 11:45:39形式

print time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())

# 格式化成Sat Mar 28 22:24:24 2016形式

print time.strftime("%a %b %d %H:%M:%S %Y", time.localtime())

# 将格式字符串转换为时间戳

a = "Sat Mar 28 22:24:24 2016" print time.mktime(time.strptime(a,"%a %b %d %H:%M:%S %Y"))

格式：

%Y  Year with century as a decimal number.  
%m  Month as a decimal number [01,12].  
%d  Day of the month as a decimal number [01,31].  
%H  Hour (24-hour clock) as a decimal number [00,23].  
%M  Minute as a decimal number [00,59].  
%S  Second as a decimal number [00,61].  
%z  Time zone offset from UTC.  
%a  Locale's abbreviated weekday name.  
%A  Locale's full weekday name.  
%b  Locale's abbreviated month name.  
%B  Locale's full month name.  
%c  Locale's appropriate date and time representation.  
%I  Hour (12-hour clock) as a decimal number [01,12].  
%p  Locale's equivalent of either AM or PM.

### 格式化输入日期 time. [strptime](http://localhost:12345/time.html#-strptime)(string, format)

# requests, bs4

## res=requests.get(net,timeout=35) 获取网站

例子：res= requests.get(‘http://www.baidu.com’,timeout=35)

Print(res.text) #打印出网站内容

## soup=bs4.BeautifulSoup(res.text) 根据网站内容生成解析对象

## tablist=soup.select('css语言') 根据css过滤出所有标签的集合

返回：所有标签内容的集合。

## str=tab.get(‘href’) 取得该标签名为href的属性

返回：标签名为href的属性的内容。为字符串

例子：str=soup.select('table#archiveResult')[0].get(‘href’)

## str=tab.getText() 取得该标签的内容

返回：标签的内容。为字符串。

# 正则表达式

## compile()函数 （可有可无），制造一个正则表达式对象

返回：一个pattern对象，即正则表达式

## findall( **pattern， string [，flags]** ) 寻找string中符合pattern的所有内容

返回：一个list，包含所有符合的内容

参数：flags:一般不用

例子：

patern=re.compile('k\dk')

res=re.findall(patern,“kangchi is k1k,k2k is kangchi”)

#res=[‘k1k’,‘k2k’]

## **match(**pattern， string， flag=0**),search(**pattern， string， flag=0**) 查找**

参数：flags:一般不用

返回：匹配成功：一个Match对象，失败：None

两者区别：match是从string的第一个字符开始匹配，若失败则直接返回；search也是从第一个字符开始，但是如果失败，会接着从下一个字符继续匹配，一直到成功或者到尾为止。

例子：

patern=re.compile('k\dk')

res\_match=re.match(patern,“kangchi is k1k,k2k is kangchi”)

res\_search=re. search (patern,“kangchi is k1k,k2k is kangchi”)

# res\_match=None

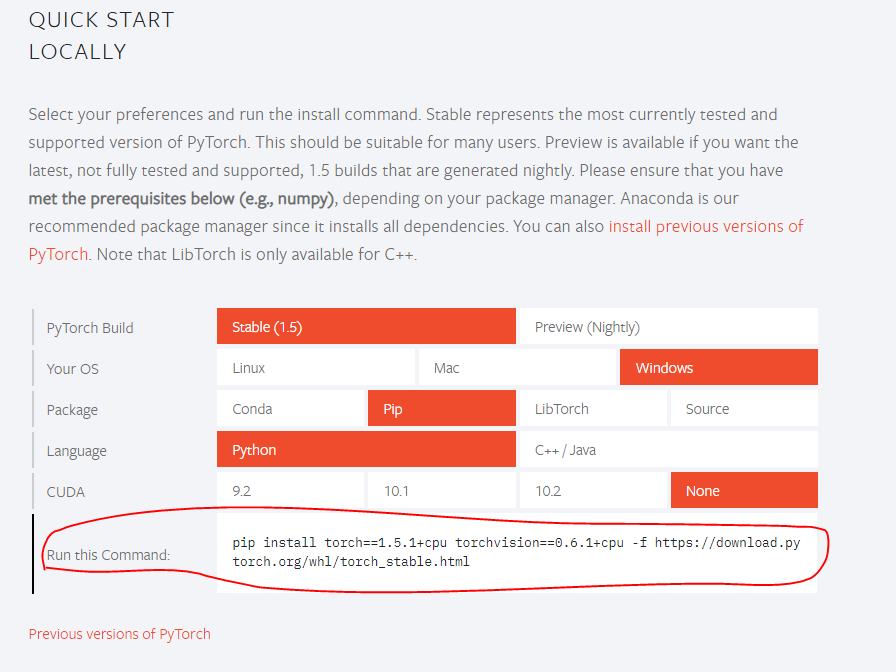
#res\_search是一个Match 对象，该对象包含

1. group():返回整个的匹配对象
2. group(0)/group(1)/… :如果pattern中包含小括号（），则group(n)表示第n个小括号内的内容。
3. groups(): 参考（2），返回所有小括号的内容，组成元组。

# Pytorch

## 安装

方法一：官网上<https://pytorch.org/>



红圈处有命令。有pip、conda等方式。

方法二：或者直接在Anaconda上使用图形界面安装。

