



# Python与金融数据挖掘(6)

文欣秀

[wenxinxiu@ecust.edu.cn](mailto:wenxinxiu@ecust.edu.cn)

# 案例分析



# HTML

```
<html>
```

```
<head>
```

```
<title>My Homepage</title>
```

```
</head>
```

```
<body>
```

```
Welcome everyone.
```

```
</body>
```

```
</html>
```

# Head

◆ **<title>**

◆ **<meta>**

◆ **<link>**

◆ **<script>**

# <meta>标签

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<meta name="author" content="Mike">
```

```
<title>My Homepage</title>
```

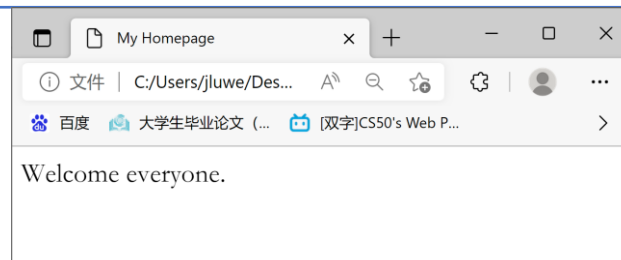
```
</head>
```

```
<body>
```

```
Welcome everyone.
```

```
</body>
```

```
</html>
```



# Body

◆ **<p>**

◆ **<div>**

◆ **<ul>**

◆ **<ol>**

◆ **<img>**

◆ **<a>**

# <p>标记

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

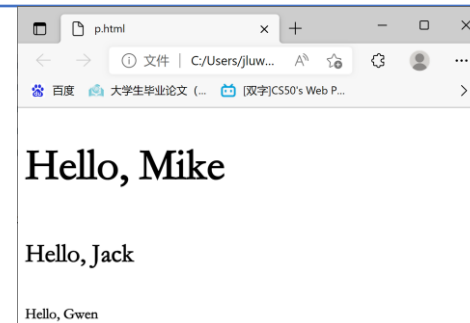
```
<p><h1>Hello, Mike<h1></p>
```

```
<p><h3>Hello, Jack<h3></p>
```

```
<p><h6>Hello, Gwen<h6></p>
```

```
</body>
```

```
</html>
```



# <div>标记

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p>Hello, Mike</p>
```

```
<div style="color:#0000FF">
```

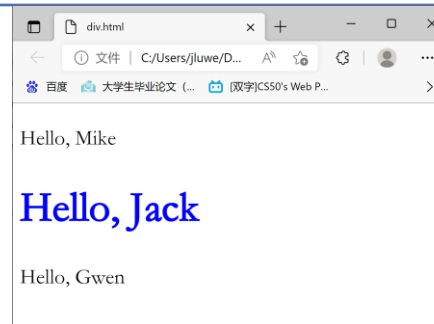
```
<p><h1>Hello, Jack</h1></p>
```

```
</div>
```

```
<p>Hello, Gwen</p>
```

```
</body>
```

```
</html>
```





# <ul>标记

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<ul>
```

```
<li>Mike</li>
```

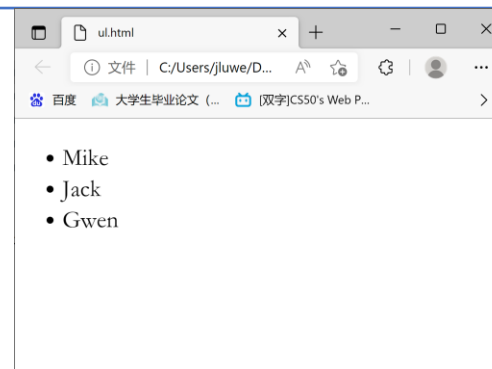
```
<li>Jack</li>
```

```
<li>Gwen</li>
```

```
</ul>
```

```
</body>
```

```
</html>
```



# <ol> 标记

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<ol>
```

```
<li>Mike</li>
```

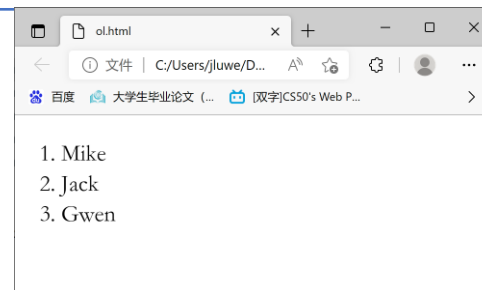
```
<li>Jack</li>
```

```
<li>Gwen</li>
```

```
</ol>
```

```
</body>
```

```
</html>
```



# <img>标记

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```

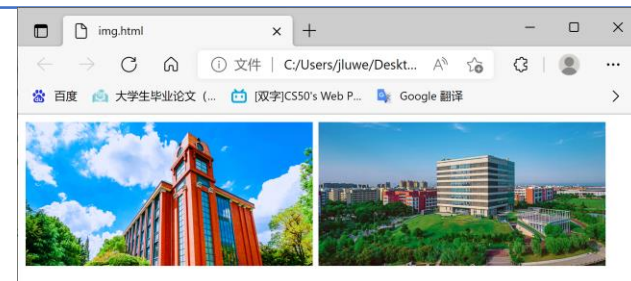
```

```

```

```
</body>
```

```
</html>
```



# <div> 和 <img>标记

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```

```

```
<div align="center">
```

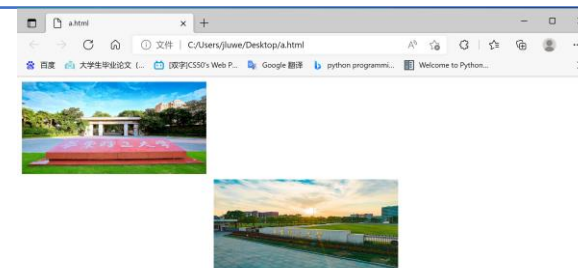
```

```

```
</div>
```

```
</body>
```

```
</html>
```



# <a> 标记

```
<!DOCTYPE html>
```

```
<html>
```

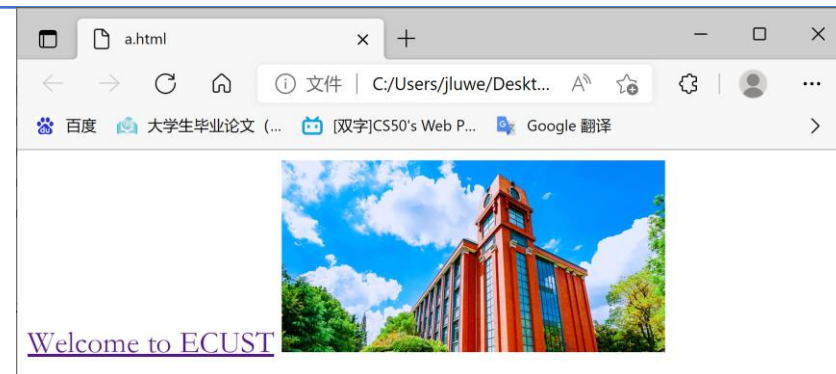
```
<body>
```

```
<a href="http://www.ecust.edu.cn">Welcome to ECUST</a>
```

```
<a href="http://xiaoban.ecust.edu.cn"> </a>
```

```
</body>
```

```
</html>
```



# 网络爬虫

**网络爬虫：** 是一种按照一定的规则， 自动地抓取万维网信息的程序或者脚本。

**爬虫常用库：** requests  
urllib  
beautifulsoup  
...

# 简单爬虫程序基本步骤

- ◆ 获取网页源码
- ◆ 根据源码中所在链接特点写出正则表达式
- ◆ 用正则对象匹配获取目标链接
- ◆ 用循环结构遍历目标链接并自动下载信息

# 爬取网页内容

**requests**第三方库：用**pip install requests**安装

```
import requests  
url="http://www.people.com.cn/"  
r=requests.get(url)  
print(r.text)
```



# Requests对象的属性

| 属性                          | 说明                    |
|-----------------------------|-----------------------|
| <b>r. text</b>              | 网页响应内容的 <b>字符串</b> 形式 |
| <b>r. encoding</b>          | <b>猜测</b> 网页响应内容编码方式  |
| <b>r. apparent_encoding</b> | 从网页内容 <b>分析</b> 出编码方式 |
| <b>r. content</b>           | 网页响应内容的 <b>二进制</b> 形式 |

# 修改编码方式

```
import requests  
url="http://www.people.com.cn/"  
r=requests.get(url)  
r.encoding=r.apparent_encoding  
data=r.text  
print(data)
```

# 爬取单张图片

```
import requests
url="http://www.people.com.cn/NMediaFile/2023/0329/MAIN202303
291337317271860541036.jpg"
r=requests.get(url)
data=r.content
fobj=open("result.jpg","wb")
fobj.write(data)
fobj.close()
```

# 爬取单个文件

```
import requests
url="http://file.finance.sina.com.cn/211.154.219.97:9494/MRGG/CNS
ESZ_STOCK/2022/2022-4/2022-04-30/8179894.PDF"
r=requests.get(url)
data=r.content
print(data)
fobj=open("格力年报.pdf","wb")
fobj.write(data)
fobj.close()
```

# 判断邮件地址是否合法

假定某E-mail地址由三部分构成：英文字母或数字（1～10个字符）、“@”、英文字母或数字（1～10个字符）、“.”，最后以com或org结束，其正则表达式为：

**'^[a-zA-Z0-9]{1,10}@[a-zA-Z0-9]{1,10}.(com|org)\$'**

输入E-mail地址测试串，忽略大小写，输出判断是否符合设定规则。

# 正则表达式

正则表达式 (**regular expression, re**):是由一些特定字符及其组合所组成的字符串表达式 (模板), 用来对目标字符串进行过滤操作。处理正则表达式需要引入**标准库是re**。

# 字符串匹配模式

## (1) 数字和字符

'**d**': 匹配一个数字

'**w**': 匹配一个字母、数字或下划线

例如: '\d\d\d'能匹配'021', 但不能匹配'AB1'

'\w\w\w'既能匹配'021', 又能匹配'AB1'

# 匹配数字与字符案例

```
import re  
content = 'Hello 123 world 456'  
result = re.findall('\d\d\d', content)  
print(result)
```

```
import re  
content = 'Hello 123 world 456'  
result = re.findall('\w\w\w', content)  
print(result)
```



# 字符串匹配模式

## (2) 任意单个字符

'.'：匹配任意单个字符

```
import re
content = 'Hello 123 world 456'
result = re.findall('.', content)
print(result)
```

# 字符串匹配模式

## (3) 多个字符

'\*': 表示任意多个字符(包括0个)

'+' : 表示至少1个字符

'?' : 表示0个或1个字符

'{n}': 表示n个字符

'{n, m}': 表示n至m个字符

# 匹配数字与字符案例

```
import re
content = 'Hello 123 world 456'
result = re.findall('.*', content)
print(result)
```

```
import re
content = 'Hello 123 world 456'
result = re.findall('.{5}', content)
print(result)
```

# 字符串匹配模式

## (4) 字符范围

'**[ ]**': 表示字符范围，1组方括号只能表示1个字符

例如: '**[0-9a-zA-Z\\_]**'匹配1个数字、字母或下划线

'**|**': 表示或关系

例如: '**[P|p]**ython'可匹配'Python'或'python'

# 字符串匹配模式

## (5) 开头和结尾

'**^**': 表示行开头, '^d'表示必须以数字开头

'**\$**': 表示行结束, 'd\$'表示必须以数字结束

例如: '^py\$'只能匹配整行完整的'py'

# 字符串匹配模式

## (6) 特殊字符

换行符'\n'、回车符'\r'、空白符'\s'、制表位符'\t'等，  
要用'\''转义或加r前缀统一转义

# re库的内置函数

- ◆ `match()`: 用于从起始位置匹配
- ◆ `search()`: 搜索整个字符串的所有匹配
- ◆ `findall()`: 以列表形式返回全部能匹配的字符串
- ◆ `compile()`: 创建一个正则表达式对象

# 判断邮件地址是否合法

```
import re

p=re.compile('^[a-zA-Z0-9]{1,10}@[a-zA-Z0-9]{1,10}.(com|org)$',re.I)

while True:

    s=input("请输入测试E-mail地址(输入 '0' 退出程序):\n")

    if s=='0':

        break
```



# 判断邮件地址是否合法

```
m=p. match(s)

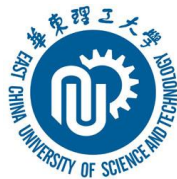
print(m)

if m:

    print('%s符合规则' %s)

else:

    print('%s不符合规则' %s)
```



谢 谢