Tedu.cn
达内教育

Python开发进阶

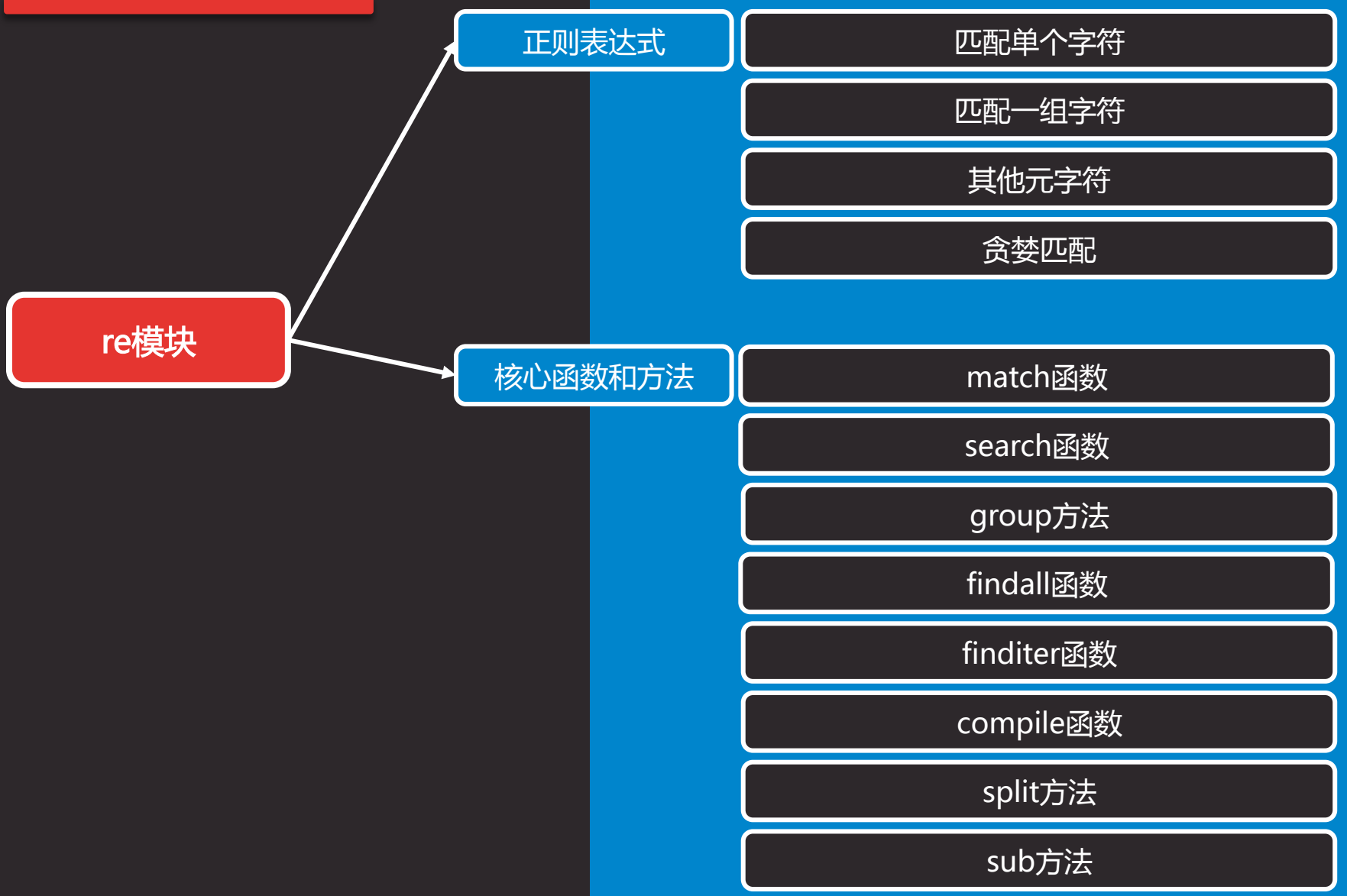
NSD PYTHON2

DAY04

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	re模块
	10:30 ~ 11:20	
	11:30 ~ 12:00	
下午	14:00 ~ 14:50	socket模块
	15:00 ~ 15:50	
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑

re模块


re模块
正则表达式
匹配单个字符
匹配一组字符
其他元字符
贪婪匹配
核心函数和方法
match函数
search函数
group方法
findall函数
finditer函数
compile函数
split方法
sub方法

Tedu.cn
达内教育

正则表达式

匹配单个字符

知识讲解

记号	说 明
.	匹配任意字符（换行符除外）
[...x-y...]	匹配字符组里的任意字符
[^...x-y...]	匹配不在字符组里的任意字符
\d	匹配任意数字，与[0-9]同义
\w	匹配任意数字字母字符，与[0-9a-zA-Z_]同义
\s	匹配空白字符，与[\r\v\f\t\n]同义

匹配一组字符

知识讲解

记号	说 明
literal	匹配字符串的值
re1|re2	匹配正则表达式re1或re2
*	匹配前面出现的正则表达式零次或多次
+	匹配前面出现的正则表达式一次或多次
?	匹配前面出现的正则表达式零次或一次
{M, N}	匹配前面出现的正则表达式至少M次最多N次

其他元字符

知识讲解

记号	说 明
^	匹配字符串的开始
$	匹配字符串的结尾
\b	匹配单词的边界
()	对正则表达式分组
\nn	匹配已保存的子组

贪婪匹配

- *、+和?都是贪婪匹配操作符，在其后加上?可以取消其贪婪匹配行为
- 正则表达式匹配对象通过groups函数获取子组

```
>>> data = 'My phone number is: 150888899999'
>>> m = re.search('.+(\d+)', data)
>>> print m.groups()
('9',)
>>>
>>> m = re.search('.+?(\d+)', data)
>>> m.groups()
('150888899999',)
```

知识讲解

Tedu.cn
达内教育

核心函数和方法

match函数

- 尝试用正则表达式模式从字符串的开头匹配，如果匹配成功，则返回一个匹配对象；否则返回None

```
>>> import re
>>> m = re.match('foo', 'food')          #成功匹配
>>> print(m)
<_sre.SRE_Match object; span=(0, 3), match='foo'>
>>>
>>> m = re.match(‘foo’, ‘seafood’)      #未能匹配
>>> print(m)
None
```

知识讲解

search函数

- 在字符串中查找正则表达式模式的第一次出现，如果匹配成功，则返回一个匹配对象；否则返回None

```
>>> import re
>>> m = re.search('foo', 'food')
>>> print(m)
<_sre.SRE_Match object; span=(0, 3), match='foo'>
>>>
>>> m = re.search(‘foo’, ‘seafood’)     #可以匹配在字符中间的模式
>>> print(m)
<_sre.SRE_Match object; span=(3, 6), match='foo'>
```

知识讲解

group方法

- 使用match或search匹配成功后，返回的匹配对象可以通过group方法获得匹配内容

```
>>> import re
>>> m = re.match('foo', 'food')
>>> print(m.group())
foo

>>> m = re.search('foo', 'seafood')
>>> m.group()
'foo'
```

知识讲解

findall函数

- 在字符串中查找正则表达式模式的所有（非重复）出现；返回一个匹配对象的列表

```
>>> import re
>>> m = re.search('foo', 'seafood is food')
>>> print(m.group())    #search只匹配模式的第一次出现
foo
>>>
>>> m = re.findall('foo', 'seafood is food')  #获得全部的匹配项
>>> print(m)
['foo', 'foo']
```

知识讲解

finditer函数

- 和findall()函数有相同的功能，但返回的不是列表而是迭代器；对于每个匹配，该迭代器返回一个匹配对象

```
>>> import re
>>> m = re.finditer('foo', 'seafood is food')
>>> for item in m:
...     print(item.group())
...
foo
foo
```

知识讲解

compile函数

- 对正则表达式模式进行编译，返回一个正则表达式对象
- 不是必须要用这种方式，但是在大量匹配的情况下，可以提升效率

```
>>> import re
>>> patt = re.compile('foo')
>>> m = patt.match('food')
>>> print(m.group())
foo
```

知识讲解

split方法

- 根据正则表达式中的分隔符把字符分割为一个列表，并返回成功匹配的列表
- 字符串也有类似的方法，但是正则表达式更加灵活

```
>>> import re      #使用 . 和 - 作为字符串的分隔符
>>> mylist = re.split('\.|-', 'hello-world.data')
>>> print(mylist)
['hello', 'world', 'data']
```

知识讲解

sub方法

- 把字符串中所有匹配正则表达式的地方替换成新的字符串

```
>>> import re
>>> m = re.sub('X', 'Mr. Smith', 'attn: X\nDear X')
>>> print(m)
attn: Mr. Smith
Dear Mr. Smith
```

知识讲解

案例1：分析apache访问日志

- 编写一个apche日志分析脚本
  1. 统计每个客户端访问apache服务器的次数
  2. 将统计信息通过字典的方式显示出来
  3. 分别统计客户端是Firefox和MSIE的访问次数
  4. 分别使用函数式编程和面向对象编程的方式实现

课堂练习

socket模块


socket模块
C/S架构
什么是C/S架构
套接字
面向连接与无连接
socket函数与方法
创建TCP服务器
创建TCP客户端
创建UDP服务器
创建UDP客户端

C/S架构

什么是C/S架构

- 服务器是一个软件或硬件，用于提供客户需要的“服务”
- 硬件上，客户端常见的就是平时所使用的PC机，服务器常见的有联想、DELL等厂商生产的各种系列服务器
- 软件上，服务器提供的服务主要是程序的运行，数据的发送与接收、合并、升级或其它的程序或数据的操作

知识讲解

套接字

- 套接字是一种具有“通讯端点”概念的计算机网络数据结构
- 套接字起源于20世纪70年代加利福尼亚大学伯克利分校版本的Unix
- 一种套接字是Unix套接字，其“家族名”为AF_UNIX
- 另一种套接字是基于网络的，“家族名”为AF_INET
- 如果把套接字比做电话的插口，那么主机与端口就像区号与电话号码的一对组合

知识讲解

面向连接与无连接

- 无论你使用哪一种地址家族，套接字的类型只有两种。一种是面向连接的套接字，另一种是无连接的套接字
- 面向连接的主要协议就是传输控制协议TCP，套接字类型为SOCK_STREAM
- 无连接的主要协议是用户数据报协议UDP，套接字类型为SOCK_DGRAM
- python中使用socket模块中的socket函数实现套接字的创建

知识讲解

Tedu.cn
达内教育

socket函数与方法

创建TCP服务器

- 创建TCP服务器的主要步骤如下：
  1. 创建服务器套接字：s = socket.socket()
  2. 绑定地址到套接字：s.bind()
  3. 启动监听：s.listen()
  4. 接受客户连接：s.accept()
  5. 与客户端通信：recv()/send()
  6. 关闭套接字：s.close()

知识讲解

案例2：创建TCP时间戳服务器

- 编写一个TCP服务器
  1. 服务器监听在0.0.0.0的21567端口上
  2. 收到客户端数据后，将其加上时间戳后回送给客户端
  3. 如果客户端发过来的字符全是空白字符，则终止与客户端的连接

课堂练习

创建TCP客户端

- 创建TCP客户端的步骤主要如下：
  1. 创建客户端套接字：cs = socket.socket()
  2. 尝试连接服务器：cs.connect()
  3. 与服务器通信：cs.send()/cs.recv()
  4. 关闭客户端套接字：cs.close()

知识讲解

案例3：创建TCP时间戳客户端

- 编写一个TCP客户端
  1. 连接服务器的21567
  2. 接收用户从键盘上的输入
  3. 发送接收到的字符串给服务器
  4. 如果用户按ctrl + c则退出程序

课堂练习

创建UDP服务器

- 创建UDP服务器的主要步骤如下：
  1. 创建服务器套接字：s = socket.socket()
  2. 绑定服务器套接字：s.bind()
  3. 接收、发送数据：s.recvfrom()/ss.sendto()
  4. 关闭套接字：s.close()

知识讲解

案例4：创建UDP时间戳服务器

- 编写一个UDP服务器
  1. 服务器监听在0.0.0.0的21567端口上
  2. 收到客户端数据后，将其加上时间戳后回送给客户端

课堂练习

创建UDP客户端

- 创建UDP客户端的步骤主要如下：
  1. 创建客户端套接字：cs = socket.socket()
  2. 与服务器通信：cs.sendto()/cs.recvfrom()
  3. 关闭客户端套接字：cs.close()

知识讲解

案例5：创建UDP时间戳客户端

- 编写一个UDP客户端
  1. 连接服务器的21567
  2. 接收用户从键盘上的输入
  3. 发送接收到的字符串给服务器
  4. 如果用户按ctrl + c则退出程序

课堂练习

Tedu.cn
达内教育

总结和答疑