Python实验1：

第1题：即串和正则表达式的实验题3

根据提供的HTML文本内容，我们需要去除所有的HTML标签以提取纯文本内容。可以使用Python中的正则表达式库 `re` 来实现这一目标。以下是具体的步骤和代码示例：

1. 导入所需的库。

2. 编写一个函数，该函数接受包含HTML标签的字符串，并返回只含有纯文本的内容。

3. 使用正则表达式匹配所有的HTML标签，并将其替换为空字符。

下面是具体实现的Python代码：

```python

import re

def remove\_html\_tags(html):

"""

去除HTML标签，保留纯文本内容。

参数:

html (str): 包含HTML标签的字符串

返回:

str: 不含HTML标签的纯文本内容

"""

# 正则表达式模式，用于匹配HTML标签

tag\_re = re.compile(r'<.\*?>')

# 使用正则表达式替换HTML标签为""

text = tag\_re.sub('', html)

return text.strip() # 删除首尾空白字符

# 测试数据

html\_text = '''

<title>Python3 正则表达式 | 菜鸟教程</title>

<div id="topics">

<div class="post">

<h1 class="postTitle">

<a id="cb\_post\_title\_url" class="postTitle2" href="https://www.cnblogs.com/cq90/p/6959567.html">python学习 - 字符串前面添加u,r,b的含义</a>

</h1>

<div class="clear"></div>

<div class="postBody">

<div id="cnblogs\_post\_body" class="blogpost-body">

&lt;p&gt;U:表示unicode字符串&amp;nbsp;&lt;br&gt;不是仅仅是针对中文，可以针对任何的字符串，代表是对字符串进行unicode编码。&amp;nbsp;&lt;br&gt;一般英文字母在&lt;p&gt;r/R:非转义的原始字符串&amp;nbsp;&lt;br&gt;与普通字符相比，其他相对特殊的字符，其中可能包含转义字符，即那些，反斜杠加上对应字母，表示对应的特殊含义的，比如最常见的“\n”表示换行，“\t”表示Tab等。而如果是以&lt;p&gt;b:bytes&amp;nbsp;&lt;br&gt;python3.x里默认的str是(py2.x里的)unicode，bytes是(py2.x)的str，b‘’’前缀代表的就是bytes&amp;nbsp;&lt;br&gt;python2.x里，b前缀没什么具体意义，只是为了兼容python3.x的这种写法&lt;/p&gt;

&lt;p&gt;&amp;nbsp;&lt;/p&gt;

&lt;p&gt;参考：http://blog.csdn.net/u010496169/article/details/70045895&lt;/p&gt;

</div>

</div>

</div>

</div>

'''

# 去除HTML标签

cleaned\_text = remove\_html\_tags(html\_text)

print(cleaned\_text)

```

这段代码定义了一个名为 `remove\_html\_tags` 的函数，它接收一个包含HTML标签的字符串作为输入，然后使用正则表达式移除所有的HTML标签，最后返回仅包含纯文本的字符串。在代码中，我们还提供了测试数据 `html\_text` 并打印了去除了HTML标签后的结果。

这段代码的主要功能是去除给定 HTML 文本中的所有 HTML 标签，只保留纯文本内容。下面是对代码的详细解析：

### 导入模块

```python

import re

```

导入了 Python 的正则表达式模块 `re`，用于后续的正则表达式操作。

### 定义函数 `remove\_html\_tags`

```python

def remove\_html\_tags(html):

"""

去除HTML标签，保留纯文本内容。

参数:

html (str): 包含HTML标签的字符串

返回:

str: 不含HTML标签的纯文本内容

"""

```

定义了一个名为 `remove\_html\_tags` 的函数，该函数接受一个包含 HTML 标签的字符串作为参数，并返回一个去除了所有 HTML 标签的纯文本字符串。

### 编写正则表达式模式

```python

tag\_re = re.compile(r'<.\*?>')

```

创建了一个正则表达式对象 `tag\_re`，用于匹配 HTML 标签。这里的正则表达式 `r'<.\*?>'` 匹配以 `<` 开头，以 `>` 结尾的任意字符序列，是最短匹配模式（非贪婪模式），因此只会匹配最近的一个 `>`。

### 使用正则表达式替换 HTML 标签

```python

text = tag\_re.sub('', html)

```

使用 `tag\_re.sub('', html)` 方法将所有匹配到的 HTML 标签替换为空字符串 `''`，从而去除所有的 HTML 标签。

### 删除首尾空白字符

```python

return text.strip()

```

使用 `strip()` 方法删除字符串首尾的空白字符，然后返回处理后的纯文本字符串。

### 测试数据

```python

html\_text = '''

<title>Python3 正则表达式 | 菜鸟教程</title>

<div id="topics">

<div class="post">

<h1 class="postTitle">

<a id="cb\_post\_title\_url" class="postTitle2" href="https://www.cnblogs.com/cq90/p/6959567.html">python学习 - 字符串前面添加u,r,b的含义</a>

</h1>

<div class="clear"></div>

<div class="postBody">

<div id="cnblogs\_post\_body" class="blogpost-body">

&lt;p&gt;U:表示unicode字符串&amp;nbsp;&lt;br&gt;不是仅仅是针对中文，可以针对任何的字符串，代表是对字符串进行unicode编码。&amp;nbsp;&lt;br&gt;一般英文字母在&lt;p&gt;r/R:非转义的原始字符串&amp;nbsp;&lt;br&gt;与普通字符相比，其他相对特殊的字符，其中可能包含转义字符，即那些，反斜杠加上对应字母，表示对应的特殊含义的，比如最常见的“\n”表示换行，“\t”表示Tab等。而如果是以&lt;p&gt;b:bytes&amp;nbsp;&lt;br&gt;python3.x里默认的str是(py2.x里的)unicode，bytes是(py2.x)的str，b‘’’前缀代表的就是bytes&amp;nbsp;&lt;br&gt;python2.x里，b前缀没什么具体意义，只是为了兼容python3.x的这种写法&lt;/p&gt;

&lt;p&gt;&amp;nbsp;&lt;/p&gt;

&lt;p&gt;参考：http://blog.csdn.net/u010496169/article/details/70045895&lt;/p&gt;

</div>

</div>

</div>

</div>

'''

```

定义了一个包含 HTML 标签的字符串 `html\_text`，用于测试 `remove\_html\_tags` 函数。

### 去除 HTML 标签

```python

cleaned\_text = remove\_html\_tags(html\_text)

```

调用 `remove\_html\_tags` 函数，传入 `html\_text`，并将返回的纯文本字符串赋值给 `cleaned\_text`。

### 输出结果

```python

print(cleaned\_text)

```

打印处理后的纯文本字符串。

### 运行结果

运行这段代码后，输出应该是：

```

Python3 正则表达式 | 菜鸟教程

python学习 - 字符串前面添加u,r,b的含义

U:表示unicode字符串 不是仅仅是针对中文，可以针对任何的字符串，代表是对字符串进行unicode编码。

一般英文字母在

r/R:非转义的原始字符串

与普通字符相比，其他相对特殊的字符，其中可能包含转义字符，即那些，反斜杠加上对应字母，表示对应的特殊含义的，比如最常见的“\n”表示换行，“\t”表示Tab等。而如果是以

b:bytes

python3.x里默认的str是(py2.x里的)unicode，bytes是(py2.x)的str，b‘’’前缀代表的就是bytes

python2.x里，b前缀没什么具体意义，只是为了兼容python3.x的这种写法

参考：http://blog.csdn.net/u010496169/article/details/70045895

```

这段代码成功地去除了所有的 HTML 标签，只保留了纯文本内容。

HTML（HyperText Markup Language）是一种标记语言，用于创建网页。HTML 文档由一系列的元素组成，每个元素通常由一个开始标签、一些内容和一个结束标签组成。以下是一些常见的 HTML 标签及其用途：

### 文档结构标签

- `<html>`：定义 HTML 文档的根元素。

- `<head>`：包含文档的元数据，如标题、样式表链接、脚本等。

- `<title>`：定义文档的标题，显示在浏览器的标签页上。

- `<body>`：包含文档的所有可见内容。

### 块级标签

- `<div>`：定义一个块级容器，用于布局和样式。

- `<p>`：定义一个段落。

- `<h1>` 至 `<h6>`：定义标题，从最重要的 `<h1>` 到最不重要的 `<h6>`。

- `<ul>`：定义无序列表。

- `<ol>`：定义有序列表。

- `<li>`：定义列表项。

- `<table>`：定义表格。

- `<tr>`：定义表格行。

- `<th>`：定义表格头部单元格。

- `<td>`：定义表格单元格。

- `<hr>`：定义水平线。

- `<br>`：定义换行符。

### 行内标签

- `<span>`：定义一个行内容器，用于样式和布局。

- `<a>`：定义超链接。

- `<img>`：定义图像。

- `<strong>`：定义加粗文本。

- `<em>`：定义强调文本（通常为斜体）。

- `<i>`：定义斜体文本。

- `<b>`：定义加粗文本。

- `<u>`：定义下划线文本。

- `<sub>`：定义下标文本。

- `<sup>`：定义上标文本。

- `<code>`：定义计算机代码文本。

- `<pre>`：定义预格式化文本。

- `<abbr>`：定义缩写。

- `<acronym>`：定义首字母缩写（HTML5 中已被废弃，使用 `<abbr>` 替代）。

- `<q>`：定义短引用。

- `<cite>`：定义作品标题。

- `<ins>`：定义插入的文本。

- `<del>`：定义删除的文本。

### 表单相关标签

- `<form>`：定义一个表单，用于用户输入。

- `<input>`：定义输入字段，如文本框、复选框、单选按钮等。

- `<textarea>`：定义多行文本输入字段。

- `<button>`：定义一个按钮。

- `<label>`：定义表单控件的标签。

- `<select>`：定义下拉列表。

- `<option>`：定义下拉列表中的选项。

- `<fieldset>`：定义表单中的一组相关元素。

- `<legend>`：定义 `<fieldset>` 元素的标题。

### 多媒体标签

- `<audio>`：定义音频内容。

- `<video>`：定义视频内容。

- `<source>`：定义多媒体资源。

- `<track>`：定义字幕文件。

### 其他标签

- `<header>`：定义文档或部分的页眉。

- `<footer>`：定义文档或部分的页脚。

- `<nav>`：定义导航链接。

- `<article>`：定义独立的内容。

- `<section>`：定义文档中的节。

- `<aside>`：定义侧栏内容。

- `<figure>`：定义图片、图表、照片等。

- `<figcaption>`：定义 `<figure>` 元素的标题。

- `<main>`：定义文档的主要内容。

- `<details>`：定义用户可以查看或隐藏的详细信息。

- `<summary>`：定义 `<details>` 元素的标题。

- `<dialog>`：定义对话框或窗口。

- `<canvas>`：定义图形绘制区域。

这些标签只是 HTML 中的一部分，HTML5 引入了许多新的标签和属性，使得网页结构更加语义化和丰富。如果你想了解更多关于 HTML 的信息，可以查阅官方文档或在线教程。

第2题：

在Windows下，操作鼠标控制Windows重启，如果你还能够实现其他鼠标的操作，也请一并演示。

要求：

实验报告以pdf文件格式提交；必须使用Latex编辑。

使用Python来控制鼠标并执行系统级别的操作（如重启Windows）可以通过几个步骤来完成。首先，需要安装一些第三方库来帮助我们实现鼠标控制。这里我们可以使用`pyautogui`库来模拟鼠标点击等操作。但是，请注意，直接通过鼠标操作来重启计算机并不是一个推荐的做法，因为这可能会导致数据丢失或其他不稳定的情况。此外，直接通过代码触发系统重启应当谨慎处理，确保不会意外触发。

下面是如何使用Python和`pyautogui`库来模拟鼠标移动和点击的例子。至于重启Windows，通常会使用命令行工具或者调用Windows API来实现，而不是通过鼠标操作。我将提供如何安装必要的库，模拟鼠标操作，以及安全地使用命令行重启Windows的示例。

### 安装必要的库

首先，你需要安装`pyautogui`库。打开命令提示符或PowerShell，运行以下命令：

```bash

pip install pyautogui

```

### 模拟鼠标操作

接下来，我们将编写一个简单的脚本来展示如何使用`pyautogui`来模拟鼠标移动和点击：

```python

import pyautogui

import time

# 移动鼠标到屏幕中央

screenWidth, screenHeight = pyautogui.size()

centerX, centerY = screenWidth / 2, screenHeight / 2

pyautogui.moveTo(centerX, centerY, duration=1)

# 在当前位置点击鼠标左键

pyautogui.click()

# 右键点击

pyautogui.rightClick()

# 双击

pyautogui.doubleClick()

# 拖拽鼠标从当前位置到屏幕右上角，耗时3秒

pyautogui.dragTo(screenWidth, 0, duration=3)

```

### 使用命令行重启Windows

要安全地重启Windows，你可以使用`subprocess`模块来调用命令行命令。这是一个例子：

```python

import subprocess

def restart\_computer():

# 使用管理员权限运行

subprocess.run(['shutdown', '/r', '/t', '0'], shell=True)

# 注意：这个函数会立即重启计算机，请谨慎使用！

# restart\_computer()

```

上述`restart\_computer`函数会立刻重启计算机，其中`/r`参数表示重启，`/t 0`表示延迟0秒后执行。

### 注意事项

- \*\*安全性\*\*：直接重启计算机可能会导致未保存的数据丢失，请确保所有重要数据都已保存。

- \*\*权限\*\*：重启计算机可能需要管理员权限。

- \*\*测试环境\*\*：在生产环境中使用此类功能前，请先在一个安全的测试环境中进行充分测试。

希望这些信息对你有所帮助！如果有任何问题或需要进一步的帮助，请随时提问。

`pyautogui` 是一个非常有用的 Python 库，它允许你自动化屏幕上的鼠标和键盘操作。这可以用于自动化测试、数据抓取或任何需要与图形用户界面交互的任务。

要使用 `pyautogui` 控制鼠标，首先你需要安装这个库。如果你还没有安装 `pyautogui`，可以通过以下命令来安装：

```bash

pip install pyautogui

```

安装完成后，你可以使用以下方法来控制鼠标：

### 鼠标移动

`pyautogui.moveTo(x, y, duration)` 方法可以用来将鼠标移动到屏幕的特定位置。`x` 和 `y` 参数指定了目标位置的坐标，而 `duration` 参数是可选的，用于指定移动所需的时间（秒）。

```python

import pyautogui

# 移动鼠标到屏幕的 (100, 100) 坐标点，耗时 1 秒

pyautogui.moveTo(100, 100, 1)

```

### 鼠标点击

`pyautogui.click()` 方法可以用来执行鼠标点击操作。你可以指定点击的位置，也可以在当前位置点击。还可以通过传递额外参数来改变点击的次数或按钮类型（如左键、右键）。

```python

# 在当前位置单击鼠标左键

pyautogui.click()

# 在 (500, 500) 位置双击鼠标左键

pyautogui.doubleClick(500, 500)

# 右键点击

pyautogui.rightClick()

# 中键点击

pyautogui.middleClick()

```

### 鼠标拖拽

`pyautogui.dragTo(x, y, duration)` 或 `pyautogui.dragRel(xOffset, yOffset, duration)` 方法可以用来执行鼠标拖拽操作。`dragTo` 是将鼠标从当前位置拖拽到指定位置，而 `dragRel` 则是从当前位置向相对方向拖拽一定距离。

```python

# 拖拽鼠标到 (300, 300)，耗时 1 秒

pyautogui.dragTo(300, 300, 1)

# 相对当前位置向下拖拽 100 像素，向右拖拽 100 像素，耗时 1 秒

pyautogui.dragRel(100, 100, 1)

```

### 鼠标滚动

`pyautogui.scroll()` 方法可以用来滚动鼠标滚轮。正数表示向上滚动，负数表示向下滚动。

```python

# 向上滚动 10 单位

pyautogui.scroll(10)

# 向下滚动 10 单位

pyautogui.scroll(-10)

```

以上就是使用 `pyautogui` 控制鼠标的一些基本方法。需要注意的是，在编写脚本时要确保你的操作不会意外地影响到其他应用程序或导致数据丢失。此外，`pyautogui` 提供的功能虽然强大，但在某些情况下可能不如直接调用操作系统提供的 API 来得高效或安全。

你的代码看起来没有语法错误，应该能够正常运行。不过，在实际运行时，有几个注意事项和潜在的问题需要考虑：

1. \*\*屏幕分辨率\*\*：确保 `(100, 100)`、`(500, 500)` 和 `(300, 300)` 这些坐标在你的屏幕上是有效的。如果这些坐标超出了屏幕范围，可能会导致意外行为。

2. \*\*鼠标位置\*\*：在执行 `pyautogui.dragRel()` 方法时，鼠标会从当前位置开始拖动。因此，确保在执行此操作前，鼠标已经移动到了预期的位置。

3. \*\*操作速度\*\*：由于你设置了一些操作的持续时间为 1 秒，这些操作之间会有一定的延迟。如果你希望操作更快或更慢，可以调整 `duration` 参数。

4. \*\*中断操作\*\*：如果你需要在任何时候停止脚本，可以按下 `Ctrl+C` 来中断程序。为了更加安全，可以在脚本开头添加 `pyautogui.FAILSAFE = True`，这样当鼠标移动到屏幕的左上角（通常是 (0, 0)）时，脚本会立即停止。

5. \*\*环境准备\*\*：确保你在一个合适的环境中运行这段代码，避免对其他正在运行的应用程序造成干扰。

以下是改进后的代码示例，包括了失败安全机制：

```python

import pyautogui

# 启用失败安全机制

pyautogui.FAILSAFE = True

try:

# 移动鼠标到屏幕的 (100, 100) 坐标点，耗时 1 秒

pyautogui.moveTo(100, 100, 1)

# 在当前位置单击鼠标左键

pyautogui.click()

# 在 (500, 500) 位置双击鼠标左键

pyautogui.doubleClick(500, 500)

# 右键点击

pyautogui.rightClick()

# 中键点击

pyautogui.middleClick()

# 拖拽鼠标到 (300, 300)，耗时 1 秒

pyautogui.dragTo(300, 300, 1)

# 相对当前位置向下拖拽 100 像素，向右拖拽 100 像素，耗时 1 秒

pyautogui.dragRel(100, 100, 1)

# 向上滚动 10 单位

pyautogui.scroll(10)

# 向下滚动 10 单位

pyautogui.scroll(-10)

except KeyboardInterrupt:

print("脚本已中断")

except Exception as e:

print(f"发生错误: {e}")

```

这样，即使在运行过程中出现问题，脚本也会捕获异常并输出错误信息，使调试更加容易。