值得学习的编程方法

- 采用原型设计的思路，先利用lines和blocks两个生成器完成核心部分的编写，直接用函数添加标签替换高亮标记，达到核心要求。然后再次变成使整个程序变得功能强大。
- 将程序分为处理程序、规则、过滤器、解析器这四个部分，使每一个部分都模块化，提高了可扩展性和灵活性，然后再让他们彼此联系。
  处理程序：先提供一个由start、end、sub三个方法为主的基类，这个基类提供了能让文本转化成任何标记语言的通用方法（即start、end、sub）。无论文本是什么格式的，无论要转化成什么标记语言，都需要调用的三个方法。要想具体化到某种标记语言就可以在此基础上继承。从这个基类派生出来的子类，就可以实现某种具体的标记语言的标签和元素。比如这部分就通过子类实现了HTML语言的标签和元素形式，当然也可以再建一个子类实现别的标记语言的具体标签。
  规则：规则就是指要对某个块添加什么样的标签，是标题还是段落等。这部分就是要去定义每个规则对满足什么特点的块使用。规则也是先提供了一个基类，包括了每种规则都需要使用的方法condition和action。然后在基类的基础上派生了了各种针对某种标记语言的具体标签规则（本程序只有html也可以有其他）的子类。最后在主程序（解析器）中添加并逐个判断每个块是否满足规则即可。
  过滤器：通过正则表达式来判断每个块的内容是否符合元素标记的语法，如果符合会把纯文本的某些标记换成HTML的元素标记方法。
  解析器：就相当于主程序，也是会先建立一个基类来编写提取块，以及对块使用过滤器并判断规则的方法。然后它派生的子类就会支出在某种纯文本下要使用什么样的过滤器和规则才能转化成某种标记语言。
  由于这种模块化的编程，如果要扩展更多HTML标签，只需要在原有基础上添加新的标签或添加新的规则和过滤器即可。如果想要扩展另一种标记语言，比如XML只需要在处理程序中再生成一个有XML具体标签和元素的子类，然后就能添加现有规则下的XML标签了。如果纯文本为markdown格式，只需要添加新规则或改变规则判断条件以及添加新的解析器子类并在类中添加这些新规则和新过滤器即可。至于这几种模块的配合架构无需改变。
- callback方法的好处：这是这个程序的一大亮点。不直接调用方法，采用callback方法，将两个字符串拼接成一个方法名的方式去生成方法名。这可以极大地提高变成的灵活性和可拓展性。这样给callback两个有意义的参数（第一个参数指要执行的操作第二个参数指要对什么标签执行）这样就能自动的调用相应方法，不需要人为找方法，只需要让每个方法都按照相应的规则命名即可。这样使用起来很灵活（实现了直接通过每个规则所对应的标签名的调用HTML标签类中相应标签的三种操作方法）还容易拓展新的东西。这样无论有多少个方法都不需要在外部调用方法的时候去查找具体的方法名是什么，只需根据名字的含义调用callback这一个函数，最重要的是当不确定会调用哪个方法的时候这样更有效。
- 使用生成器来处理可迭代对象。首先建立了生成行的生成器lines，然后在blocks中对它进行遍历，每迭代一行就会采取一定的操作，每当生成一个块，blocks就出现一个迭代值。在主函数中就可以对blocks再遍历得到每一个值，这比全部生成块再去索引方便多了。

编程细节

- 确定每个块执行哪种规则：通过action返回的True确保，标题、段落这种只被添加一种规则，这样不仅保证不会出现某段文字既是标题又是题目（这是不可能的），同事保证只执行一次能输出块内容的方法，防止同一个块输出多次。action返回值为False也就是像列表项这种既要添加列表标

签又要添加列表项标签，这种通过规则列表中添加规则的顺序，使不输出块内容的规则在前（列表），最后一个是输出内容的标签（列表项），也是保证值输出一次块内容并在判断完第一个规则会不退出循环转而去判断与之有关的紧挨着下一个的规则。

- condition用于判断当前模块需不需要执行这种规则，action负责执行规则也就是将标签输出到块的前后，并返回一个判断是否还继续其他规则的布尔量。
- start和end的作用是为块添加标签，而sub是通过正则表达式来实现来替换文本中的高亮部分的标志。
- 本程序使用re.sub方法并使用函数作为其参数实现新的正则表达式，并用编组来提取内容。
- 本程序通过列表储存多种规则和多种过滤器，列表的每个元素直接是对应类的实例化对象并不是标准数据类型，因此列表存在这些对象就存在，这也是为什么能用inside这个listrule类的实例变量来充当多个块的标志尾的原因，页应该用这种对象中的属性来替代全局变量来编程。
- 字符串的strip方法不仅仅会删除两头的空格，还会删除末尾的\n，但不会影响内部。
- 注意过滤器调用re.sub()函数的实现过程

```
    def addFilters(self, pattern, name):
        def filter(block, handler):
            return re.sub(pattern, handler.sub(name), block)
        self.filters.append(filter)
```

本程序中addFIlter负责忘filter列表中添加元素，同时每次添加外界传入别替换的正则表达式。根据代码，实际上addTilter添加的是其中定义的一个函数名，也就是往filters列表中添加的是函数对象。`for filter in self.filters:block = filter(block, self.handler)`这样通过列表中的元素为函数名使用函数，并执行`re.sub(pattern, handler.sub(name), block)`pattern是外界传过来的正则表达式，而handler.sub(name)的返回值是一个满足re.sub第二个参数要求的函数名，handler.sub方法内部通过一系列机制通过通过callback函数返回针对不同元素所设计的sub替换的方法的名称。这也就看出来了callback的优点，在name这个传递的参数可变的时候可以准确调用handler类中具体的某种sub替换方法。
而添加规则的addRule方法就简单了，它将不同的需要检查的Rule类的对象作为rules列表的元素，而rule方法一个是实现了判断另一个作用就是与处理程序结合来输出标签和块。rule类也是通过type实例变量实现在未知的情况下通过callback准确调用handler类中具体规则的执行方法。
* print会自动打印\n，也是利用这种特点使标签和块分行。

改进

- 不再使用标准输入流获得文件，在程序内部让用户自己选择某个路径下的文件。
  由`python test.py <input.txt> output.html`这种通过命令行对标准输入输出流重定向，改为内部通过GUI让用户胡选择地址
- 加入GUI
  1.使用图形文件位置选择让用户自主选择要转换文本，以及生成的新html文件的位置
  2.转换完成以后自动调用浏览器查看转化成的html文件
- 增加封装程度，我设计了一个比较完整的包，用户只需要使用一个start()函数就能完成此过程。我这里面也只有这一个函数，我自己加的东西也都是封装在一个类里里，并且用户无需关心。
- 将所有没有调用self的实例方法，全部换为静态方法。
- 设计更人性化。通过异常的检测考虑了用户可能出错的地方，并出错后让用户能够返回。让程序更符合人的使用习惯。

- 采用PyLint的建议进行了部分更改

参考

https://blog.csdn.net/louishu_hu/article/details/53737965
星期三, 24. 十月 2018 10:10下午