4. 你的第一个python程序

5分钟搞定一个python爬虫

1. 寻找目标站点
2. Chrome浏览器抓包
    - Chrome抓包工具的使用
    - 更多开发者功能
3. postman生成代码
    - 如何用postman生成python代码
4. pycharm运行效果
    - pycharm如何编辑和运行python文件

程序知识点讲解：

1. 变量：变量的作用其实就是给数据取一个名字而已。
2. 函数：将程序的运行逻辑进行封装，好提高代码的利用率
3. 模块：一个或多个py源文件组成一个模块
4. 参数：给函数参数命名，实际上也是变量。

Python程序规范

Python脚本扩展名

1. 扩展名
2. 文件命名规范

Python关键字

关键字是一门语言在内部定义好的名称，这些名称具有特殊的意义。在编写程序的时候需要用到，每个关键字代表的含义都不一样。但是我们在给变量、函数、参数以及文件命名的时候一定要注意避免使用这些内置的名称。最新版本的python3目前一共有35个关键字，分别是如下这些：

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await',
'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except',
'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is',
'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try',
'while', 'with', 'yield']
```

如何获取最新的关键词信息？python的关键字保存在`keyword`模块下的`kwlist`属性下。

Python注释

所有的程序语言都会有注释，注释的作用就是解析文件、变量、函数、参数以类等所代表的意思或其它一些补充信息。

1. 单行注释

   > 单行注释是用井号（#）开头，井号后面的所有字符都是注释的内容

2. 多行注释

   > 多行注释是用三引号进行注释，可以是三个双引号`"""注释内容"""`也可以是三个单引号`'''多行注释'''`，一般用三个双引号的比较多。

Python命名规范(标识符)

标识符是什么鬼？

> 标识符其实就是我们在程序里面的符号命名，比如变量的名称、函数名称、参数名称、类名称等等

python的标识符是由数字(`0-9`)、字母(`a-zA-Z`)和下划线(`_`)组成的，区分大小写，且不能以数字开头。

一些特殊的标识符命名含义

某些标识符类 (除了关键字) 具有特殊的含义。这些标识符类的命名模式是以下划线字符打头和结尾:

- `_*`

不会被 `from module import *` 导入。特殊标识符 `_` 在交互式解释器中被用来存放最近一次求值结果；它保存在 `builtins` 模块中。当不处于交互模式时，`_` 无特殊含义也没有预定义。

- `__*__`

  系统定义的名称，在非正式场合下被叫做 "dunder" 名称。 这些名称是由解释器及其实现（包括标准库）定义的。 现有系统定义名称相关的讨论请参见 **特殊方法名称** 等章节。 未来的 Python 版本中还将定义更多此类名称。 任何情况下 *任何* 不遵循文档所显式指明的 `__*__` 名称使用方式都可能导致无警告的错误。

- `__*`

  类的私有名称。这种名称在类定义中使用时，会以一种混合形式重写以避免在基类及派生类的 "私有" 属性之间出现名称冲突。

Python文件编码

如果一条注释位于 Python 脚本的第一或第二行，并且匹配正则表达式 `coding[=:]\s*([-\w.]+)`，这条注释会被作为编码声明来处理。

推荐的编码声明格式如下：

```
# -*- coding: <encoding-name> -*-
```

如果没有编码声明，则默认编码为 UTF-8。此外，如果文件的首字节为 UTF-8 字节顺序标志 (`b'\xef\xbb\xbf'`)，文件编码也声明为 UTF-8 (这是 Microsoft 的 **notepad** 等软件支持的形式)。

Python缩进（重点）

python程序是靠缩进的方式来确定代码的层级的，而其他的语言则很多是通过花括号来进行区分。因此缩进对于python程序来讲是非常重要的。不同层级的缩进代表着你的语句和代码块是否在同样的逻辑中运行。

由于不同的操作系统和不同的编辑器的制表符 `tab` 键所代表的缩进长度不一样，因此这里建议不要用制表符来进行缩进，而是统一使用4个空格 `space` 进行缩进或者是让编辑器将制表符转化为4个空格。

下面这种方式的缩进虽然是正确的，但是不推荐。因为不统一缩进的长度。

```
def perm(l):
        # Compute the list of all permutations of l
    if len(l) <= 1:
                  return [l]
    r = []
    for i in range(len(l)):
             s = l[:i] + l[i+1:]
             p = perm(s)
             for x in p:
              r.append(l[i:i+1] + x)
    return r
```

下面演示各种缩进的错误:

```
 def perm(l):                       # error: first line indented
for i in range(len(l)):             # error: not indented
    s = l[:i] + l[i+1:]
        p = perm(l[:i] + l[i+1:])   # error: unexpected indent
        for x in p:
                r.append(l[i:i+1] + x)
            return r                # error: inconsistent dedent
```

(实际上，前三个错误会被解析器发现；只有最后一个错误是由词法分析器发现的 --- `return r` 的缩进无法匹配弹出栈的缩进层级。)

注意：**在一个源文件中如果混合使用制表符和空格符缩进，并使得确定缩进层次需要依赖于制表符对应的空格数量设置，则被视为不合规则；此情况将会引发 `TabError` 。**

Python行拼接

为了让你的python程序更加的美观，Python推出一套PEP的规范。如果一行代码或数据过长了，那么对于代码的阅读体验是非常不友好的。因此需要把长的代码或数据变成多行。这就涉及到行拼接的问题。

显式的行拼接

两个或更多个物理行可使用反斜杠字符 (`\`) 拼接为一个逻辑行，规则如下: 当一个物理行以一个不在字符串或注释内的反斜杠结尾时，它将与下一行拼接构成一个单独的逻辑行，反斜杠及其后的换行符会被删除。例如:

```
if 1900 < year < 2100 and 1 <= month <= 12 \
   and 1 <= day <= 31 and 0 <= hour < 24 \
   and 0 <= minute < 60 and 0 <= second < 60:   # Looks like a valid
date
        return 1
```

以反斜杠结束的行不能带有注释。反斜杠不能用来拼接注释。反斜杠不能用来拼接形符，字符串除外 (即原文字符串以外的形符不能用反斜杠分隔到两个物理行)。不允许有原文字符串以外的反斜杠存在于物理行的其他位置。

隐式的行拼接

圆括号、方括号或花括号以内的表达式允许分成多个物理行，无需使用反斜杠。例如:

```
month_names = ['Januari', 'Februari', 'Maart',      # These are the
               'April',   'Mei',      'Juni',       # Dutch names
               'Juli',    'Augustus', 'September',  # for the months
               'Oktober', 'November', 'December']   # of the year
```

隐式的行拼接可以带有注释。后续行的缩进不影响程序结构。后续行也允许为空白行。隐式拼接的行之间不会有 NEWLINE 形符。隐式拼接的行也可以出现于三引号字符串中 (见下)；此情况下这些行不允许带有注释。