

第一讲 - 爬虫基础课程

学术志



CONTENTS

- 数据收集
- 爬虫原理
- Python基础
- Python基本操作
- 八爪鱼爬虫

数据获取

- 公开整理好的数据
- 各种数据库
- 统计局数据
- 搜索指数、清博指数等
- 学校图书馆：人民日报（本地镜像）
- 学校图书馆：新华社专供高教信息数据库
- 抽样调查
- 网络爬虫

爬虫软件

- EXCEL
 - 八爪鱼
 - 火车头采集器
-
- JAVA
 - Python
 - C/C++
 - R



虾米志
xueshuzhi001

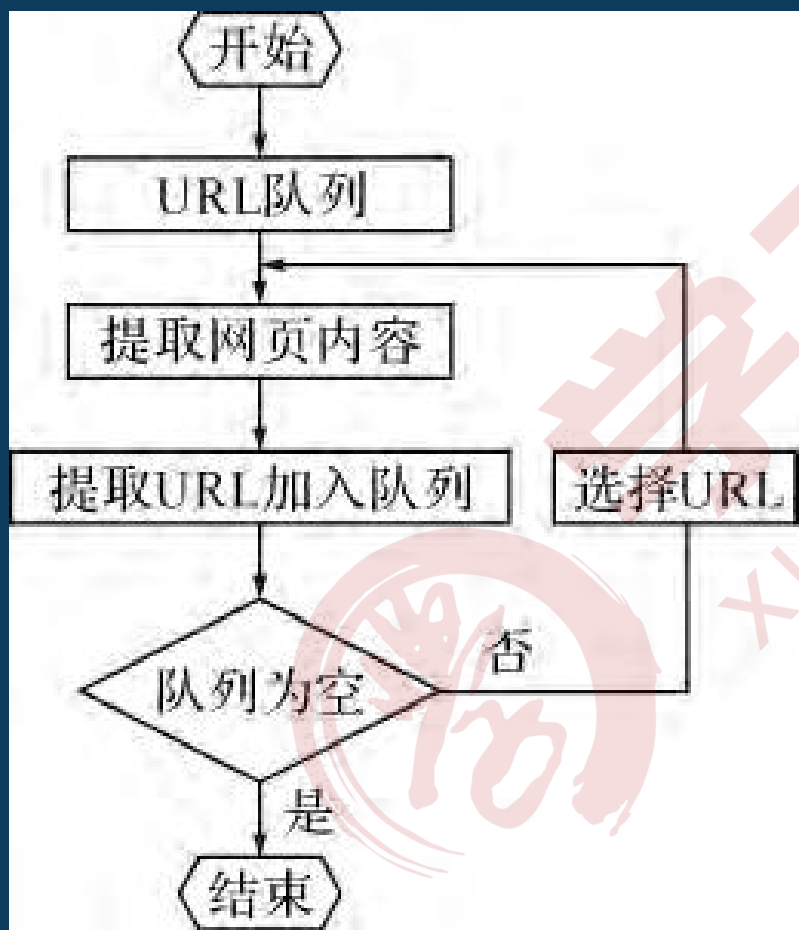
网络爬虫

网络爬虫（Crawler）又被称为网页蜘蛛，网络机器人，在FOAF（Friend-of-a-Friend）社区中，更经常的被称为网页追逐者，它是一种按照一定的规则，自动的抓取万维网信息的程序或者脚本。

很多站点，尤其是搜索引擎，都使用爬虫提供最新的数据，它主要是提供它访问过页面的一个副本，然后，搜索引擎就可以对得到的页面进行索引，以提供快速访问。

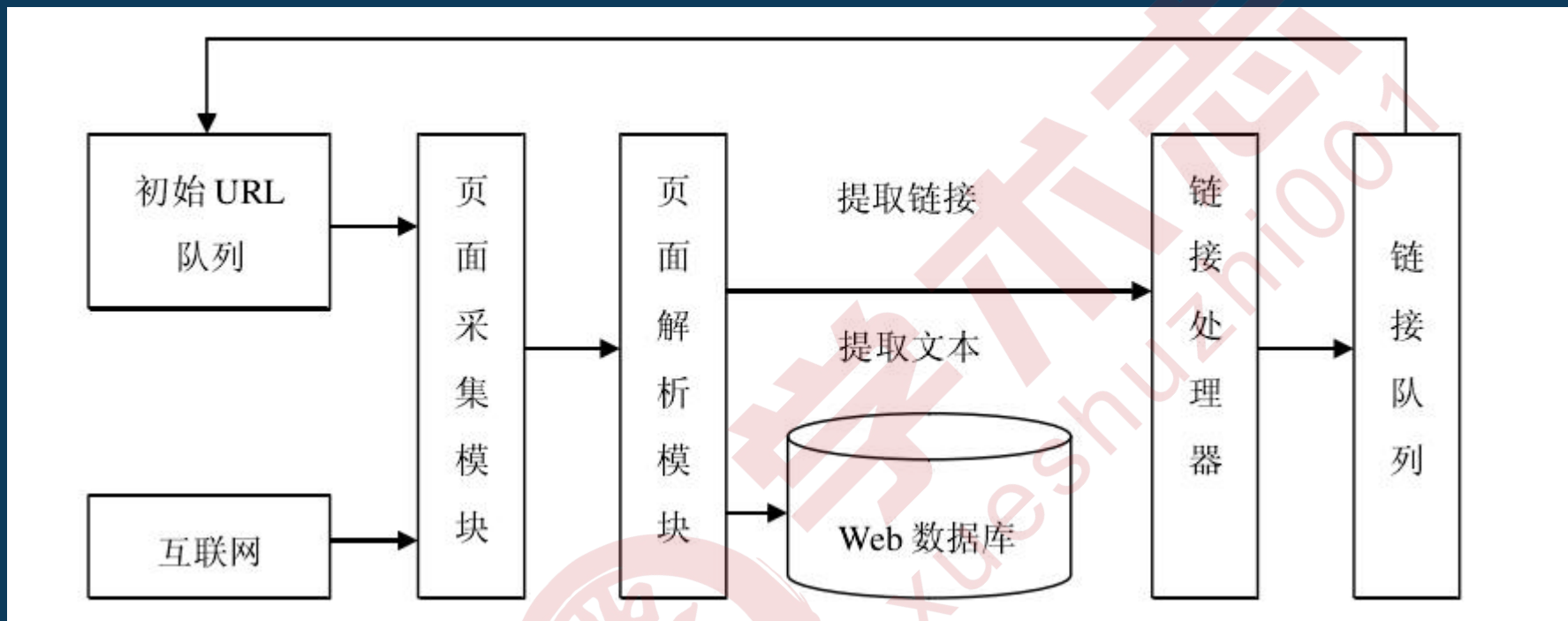
爬虫可以在web上用来自动执行一些任务，例如检查链接，确认html代码；也可以用来抓取网页上某种特定类型信息，例如抓取电子邮件地址（通常用于垃圾邮件）。

原理



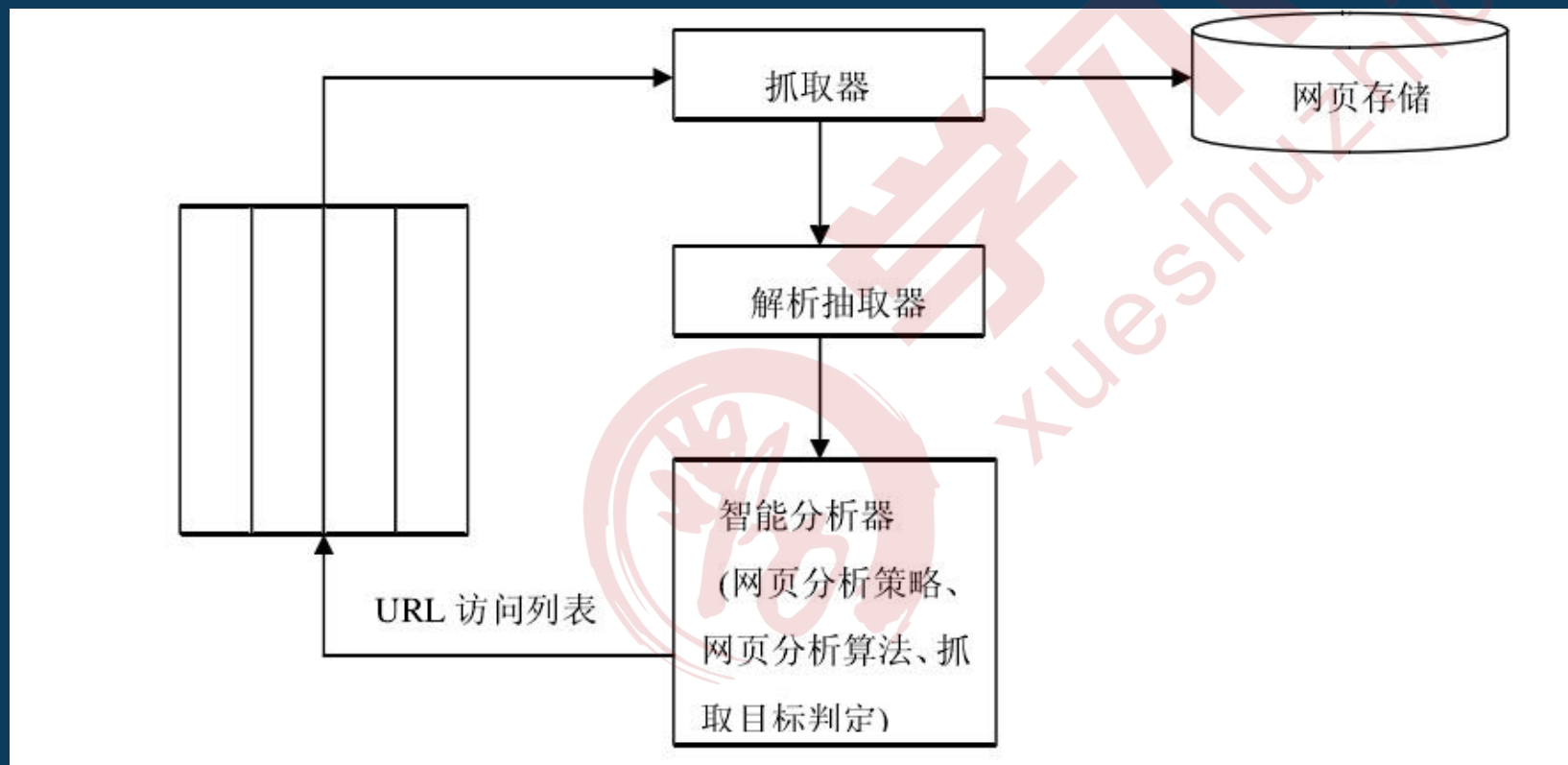
一个网络爬虫就是一种机器人，或者软件代理。大体上，它从一组要访问的URL链接开始，可以称这些URL为种子。爬虫访问这些链接，它辨认出这些页面的所有超链接，然后添加到这个URL列表，可以称作检索前沿。这些URL按照一定的策略反复访问。

通用网络爬虫



通用网络爬虫从种子链接开始，不断抓取URL网页，将这些URL全部放入到一个有序的待提取的URL队列里。**Web**信息提取器从这个队列里按顺序取出URL，通过**Web**上的协议，获取URL所指向的页面，然后从这些页面中分析提取出新的URL，并将它们放到等待提取的URL队列里。通用爬虫就是通过这样一种方式来不断遍历整个互联网，一直到等待提取的URL队列为空或者达到系统给定的停止条件为止。

聚焦爬虫



聚焦爬虫根据一定的网页分析算法，过滤与主题无关的链接，保留有用的链接并将其放入等待抓取的URL队列。然后，它将根据一定的搜索策略从队列中选择下一步要抓取的网页URL，并重复上述过程，直到达到系统的某一条件时停止。

聚焦爬虫需要解决三个主要问题：

- (1) 对抓取目标的描述或定义；
- (2) 对网页或数据的分析与过滤；
- (3) 对URL的搜索策略。

抓取目标的描述和定义是决定网页分析算法与URL搜索策略如何制订的基础。

网页分析算法和候选URL排序算法是决定搜索引擎所提供的服务形式和爬虫网页抓取行为的关键所在。、

网络爬虫的抓取策略

网页搜索策略

网页的抓取策略可以分为深度优先、广度优先和最佳优先三种。深度优先在很多情况下会导致爬虫的陷入(trapped)问题，目前常见的是广度优先和最佳优先方法。

爬行策略

网页爬虫的行为通常是四种策略组合的结果：

- (a)选择策略，决定所要下载的页面；
- (b)重新访问策略，决定什么时候检查页面的更新变化；
- (c)平衡礼貌策略，指出怎样避免站点超载；
- (d)并行策略，指出怎么协同达到分布式抓取的效果。

广度优先搜索策略

- 在抓取过程中，在完成当前层次的搜索后，才进行下一层次的搜索。
- 设计和实现相对简单。
- 在目前为覆盖尽可能多的网页，一般使用广度优先搜索方法。
- 有很多研究将广度优先搜索策略应用于聚焦爬虫中。基本思想是认为与初始URL在一定链接距离内的网页具有主题相关性的概率很大。
- 将广度优先搜索与网页过滤技术结合使用，先用广度优先策略抓取网页，再将其中无关的网页过滤掉。这些方法的缺点在于，随着抓取网页的增多，大量的无关网页将被下载并过滤，算法的效率将变低。

最佳优先搜索策略

- 按照一定的网页分析算法，预测候选URL与目标网页的相似度，或与主题的相关性，并选取评价最好的一个或几个URL进行抓取。
- 它只访问经过网页分析算法预测为“有用”的网页。
- 问题：在爬虫抓取路径上的很多相关网页可能被忽略，因为最佳优先策略是一种局部最优搜索算法。因此需要将最佳优先结合具体的应用进行改进，以跳出局部最优点。

选择策略

- 即使很大的搜索引擎也只能获取网络上可得到资源的一小部分。
- 由劳伦斯等人共同做的一项研究指出，没有一个搜索引擎抓取的内容达到网络的16%。网络爬虫通常仅仅下载网页内容的一部分，但是大家都还是强烈要求下载的部分包括最多的相关页面，而不仅仅是一个随机的简单的站点。
- 一个页面的重要程度与它自身的质量有关，与按照链接数、访问数得出的受欢迎程度有关，甚至与它本身的网址（后来出现的把搜索放在一个顶级域名或者一个固定页面上的垂直搜索）有关。设计一个好的搜索策略还有额外的困难，它必须在不完全信息下工作，因为整个页面的集合在抓取时是未知的。

重新访问策略

- 网络具有动态性很强的特性。抓取网络上的一小部分内容可能会花费很长的时间，通常用周或者月来衡量。当爬虫完成它的抓取的任务以后，很多操作是可能会发生的，这些操作包括新建、更新和删除。
- 从搜索引擎的角度来看，不检测这些事件是有成本的，成本就是我们仅仅拥有一份过时的资源。最常使用的成本函数，是新鲜度和过时性。

平衡礼貌策略

- 爬虫的使用对很多工作都是很有用的，但是对一般的社区，也需要付出代价。使用爬虫的代价包括：
 - 网络资源：在很长一段时间，爬虫使用相当的带宽高度并行地工作。
 - 服务器超载：尤其是对给定服务器的访问过高时。
 - 质量糟糕的爬虫：可能导致服务器或者路由器瘫痪，或者会尝试下载自己无法处理的页面。
 - 个人爬虫：如果过多的人使用，可能导致网络或者服务器阻塞。

并行策略

- 一个并行爬虫是并行运行多个进程的爬虫。
- 它的目标是最大化下载的速度，同时尽量减少并行的开销和下载重复的页面。
- 为了避免下载一个页面两次，爬虫系统需要策略来处理爬虫运行时新发现的URL，因为同一个URL地址，可能被不同的爬虫进程抓到。

爬虫架构

- 斯坦福大学设计了用于Google的爬虫
- 早期的Google爬虫系统由5个模块处理不同的任务。一个URL服务器从磁盘文件读URL列表并将其转发到Crawler上。
- 每个Crawler单独运行在一台机器上，采用单线程异步IO方式，一次维持300个连接并行爬行。
- Crawler将网页传输到存储服务器上压缩并保存。索引进程从HTML页面中抽取链接并存放在不同的文件中。
- 一个URL解析器读取这些链接文件并转化为绝对路径，由URL服务器读取。

后期Google的改进主要有：

- (1)采用自有的文件系统(GFS)和数据库系统(Big Table)来存取数据；
- (2)采用Map Reduce技术来分布式处理各种数据的运算。

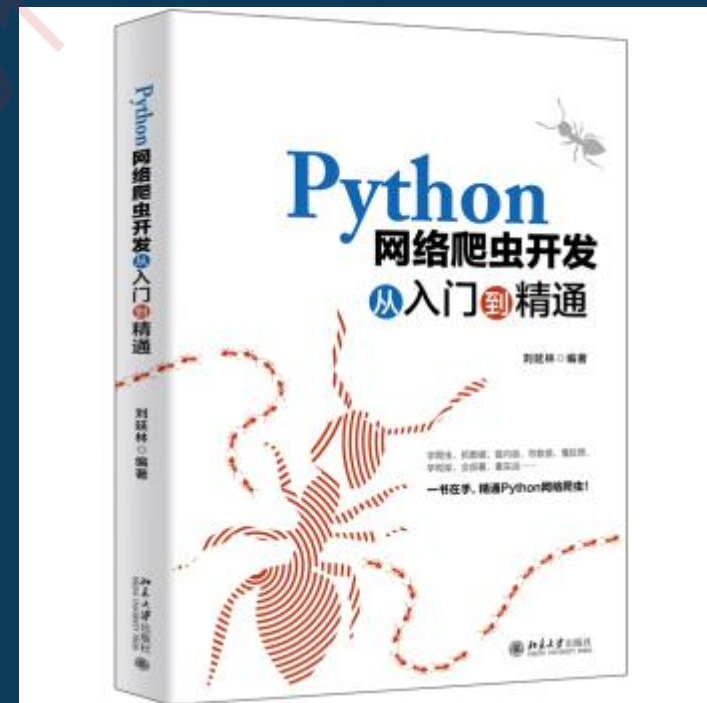
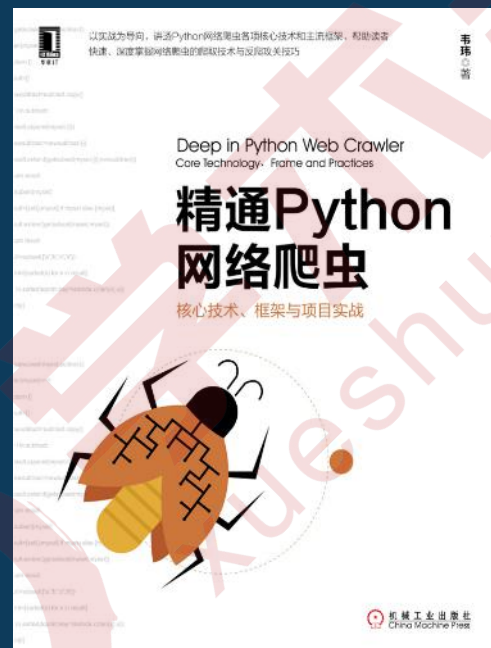
百度BaiduSpider

- 百度搜索引擎的一个自动程序。
- 它的作用是访问收集整理互联网上的网页、图片、视频等内容，然后分门别类建立索引数据库。
- 搜索引擎构建一个调度程序，来调度百度蜘蛛的工作，让百度蜘蛛去和服务器建立连接下载网页，计算的过程都是通过调度来计算的，百度蜘蛛只是负责下载网页，目前的搜索引擎普遍使用广布式多服务器多线程的百度蜘蛛来达到多线程的目的。

常用的网站（资源）

- github
- 微信公众号（搜python爬虫）
- 知乎
- CSDN
- W3school(<https://www.w3school.com.cn/python/index.asp>)
- 中国大学MOOC公开课

参考图书





Python简史

- Python语言是一种解释型、面向对象、动态数据类型的高级程序设计语言
- Python语言是数据分析师的首选数据分析语言，也是智能硬件的首选语言

数据分析



python™

文本分析

Web应用程序

网站开发

数据可视化

.....

情感分析

诞生

- Python与蟒蛇
- Guido van Rossum 于1989年在荷兰国家数学和计算机科学研究设计出来的



发展史



Python 2.0版本于2000年10月发布，2008年12月，Python 3.0发布，此版本与之前的Python 2.0不兼容

Python也因此分为了Python 3.x派系和Python 2.x派系两大阵营，2020年2.0+不再维护

编程排行

- Tiobe分析师指出，如果按照现在这个增速继续下去，很可能在3年后Python就会超越Java和C，成为最受欢迎的语言。 TIOBE INDEX

Sep 2020	Sep 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	15.95%	+0.74%
2	1	▼	Java	13.48%	-3.18%
3	3		Python	10.47%	+0.59%
4	4		C++	7.11%	+1.48%
5	5		C#	4.58%	+1.18%
6	6		Visual Basic	4.12%	+0.83%
7	7		JavaScript	2.54%	+0.41%
8	9	▲	PHP	2.49%	+0.62%
9	19	▲	R	2.37%	+1.33%
10	8	▼	SQL	1.76%	-0.19%
11	14	▲	Go	1.46%	+0.24%
12	16	▲	Swift	1.38%	+0.28%
13	20	▲	Perl	1.30%	+0.26%
14	12	▼	Assembly language	1.30%	-0.08%
15	15		Ruby	1.24%	+0.03%

Python语言特点



优雅、简单、明确



减少花哨、晦涩或以“炫技”为目的的代码



让数据分析师们摆脱了程序本身语法规则的泥潭，更快的进行数据分析

Python语言特点



强大的标准库



完善的基础代码库，覆盖了网络通信、文件处理、数据库接口、图形系统、XML处理等大量内容，被形象地称为“内置电池”
(batteries included)



Python使用者——“调包侠”

Python语言特点



良好的可扩展性



大量的第三方模块，覆盖了科学计算、Web开发、数据接口、图形系统等众多领域，开发的代码通过很好的封装，也可以作为第三方模块给别人使用。如Pandas、Numpy、Seaborn、Scikit-learn等等



免费、开源

Python语言缺点



运行速度慢



加密难



缩进规则



多线程灾难



机器学习的一把利器可读性强

便于上手，灵活性强

可与其他如Web应用程序进行整合



以统计推断为导向

数据分析之外的领域有所限制

包凌乱且一致性较差



- 网络爬虫
- 连接数据库
- 内容管理系统
- API构建



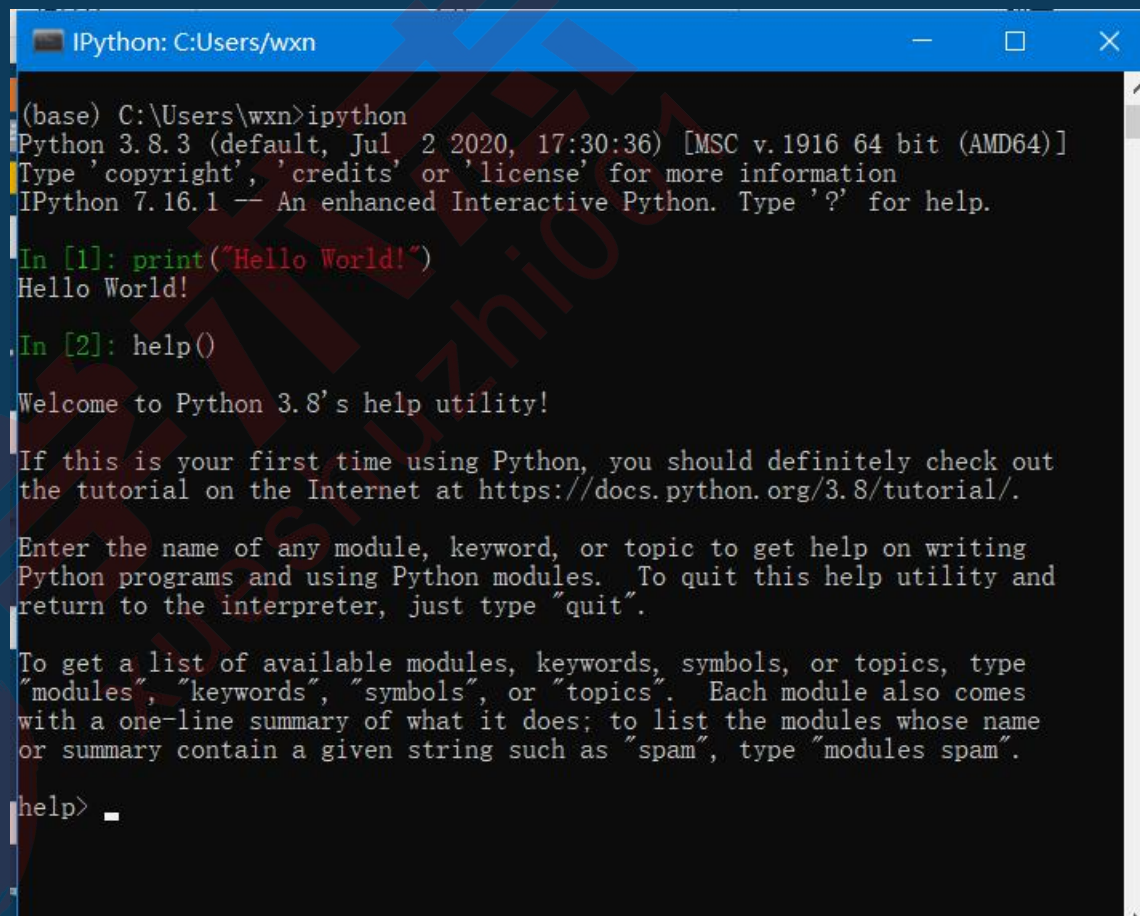
- × 统计分析
- × 交互式图标/面板

IDE

- 推荐使用Anaconda进行Python安装、环境配置及工具包管理
- Ipython
- PyCharm
- Jupyter Notebook
- Spyder

交互式计算和开发环境

- IPython鼓励一种“执行-探索”（execute-explore）的工作模式
- 输入代码之后，按下回车，便会立即得到代码运行结果
- 输入“help()”查看IPython的帮助文档



```
IPython: C:\Users\wxn
(base) C:\Users\wxn>ipython
Python 3.8.3 (default, Jul 2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.16.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: print("Hello World!")
Hello World!

In [2]: help()

Welcome to Python 3.8's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.8/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics". Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> .
```

Jupyter Notebook

Julia+Python+R = Jupyter

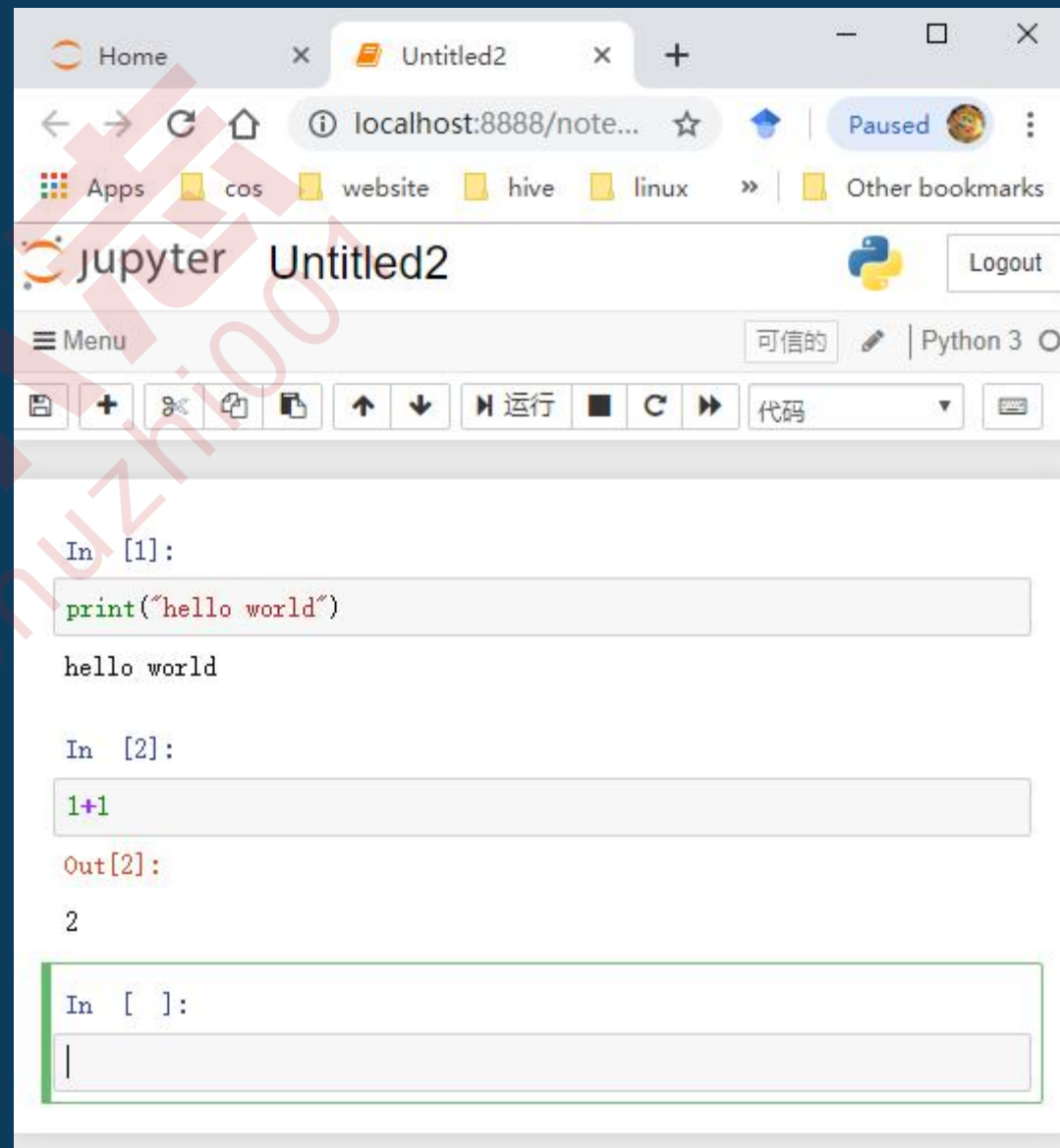
基于Web技术的交互式计算文档格式

支持Markdown和Latex语法

支持代码运行、文本输入、数学公式

编辑、内嵌式画图和其他如图片文件的

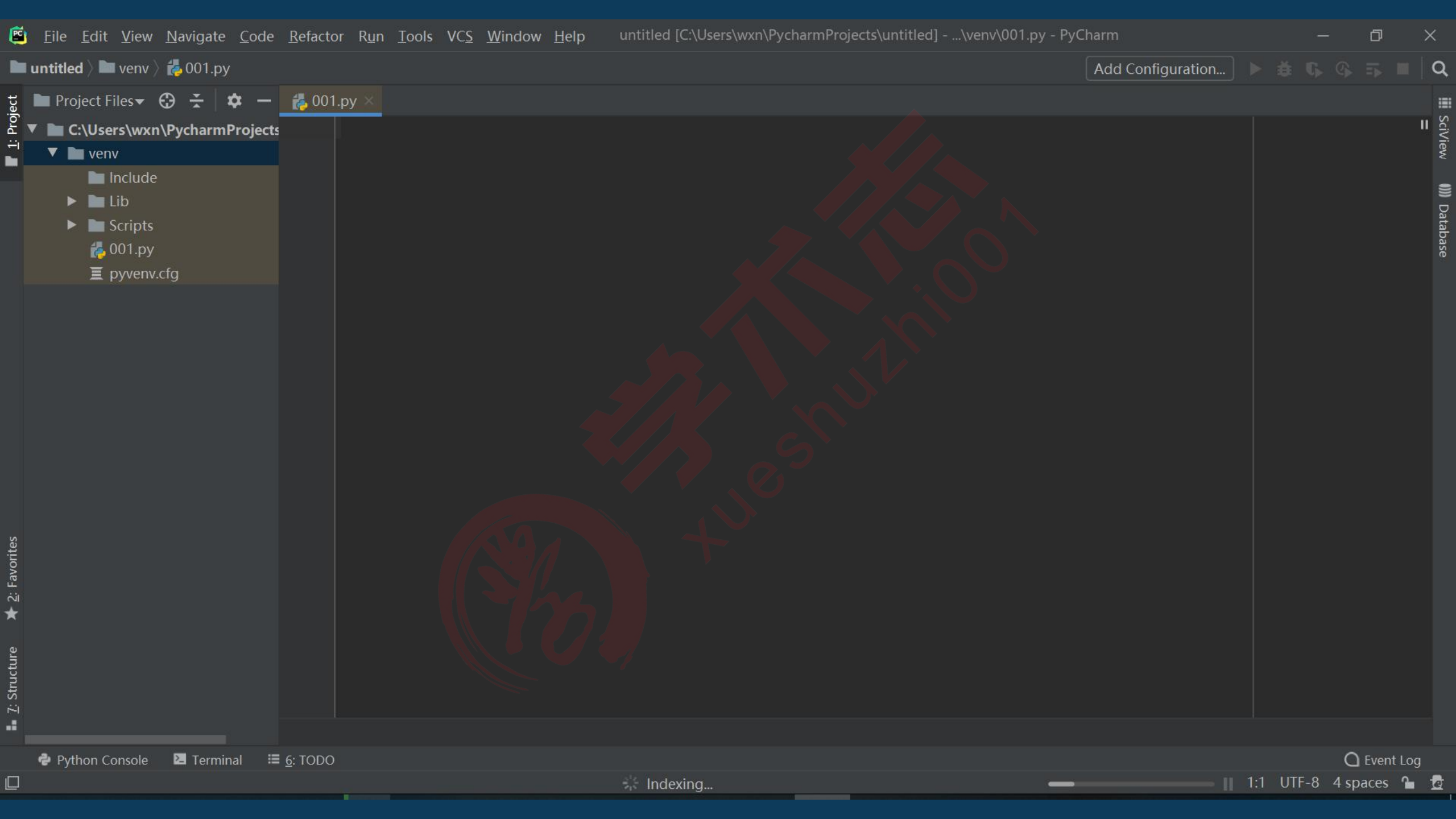
插入，是一个对代码友好的笔记本



- 启动命令：jupyter notebook
- 推荐使用Jupyter Notebook进行数据分析，并将自己数据分析的思考过程写在其中，方便之后整理思路以及向别人展示数据分析结果

集成开发环境IDE——PyCharm & Spyder

- PyCharm是一种Python IDE，带有一整套可以帮助用户在使用Python语言开发时提高其效率的工具，比如调试、语法高亮、Project管理、代码跳转、智能提示、自动完成、单元测试、版本控制。
- PyCharm 提供了一些高级功能，以用于支持Django框架下的专业Web开发。
- Spyder是Python(x,y)的作者为它开发的一个简单的集成开发环境。
- 和其他的Python开发环境相比，它最大的优点就是模仿MATLAB的“工作空间”的功能，可以很方便地观察和修改数组的值。



```
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4
5 This is a temporary script file.
6 """
7
8
```

Usage

Here you can get help of any object by pressing **Ctrl+H** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our [tutorial](#)

```
Python 3.8.3 (default, Jul 2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]:
```

Anaconda

- “leading open data science platform powered by Python”
- 自动配置Python环境，下载并安装Jupyter Notebook、qtconsole和集成开发环境Spyder
- 包管理器 conda

[Sign in to Anaconda Cloud](#)[Home](#)[Environments](#)[Learning](#)[Community](#)[Documentation](#)[Developer Blog](#)

Applications on

base (root)

Channels

Refresh



JupyterLab

1.0.2

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

[Launch](#)

Notebook

[6.0.0](#)

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

[Launch](#)

Python基础

- 数据类型
- 变量
- 注释
- print函数
- 数据类型
- 算术运算符
- 类型转换



势志
xueshuzhi001

解释型语言Python

- Python语言是一种**解释型**、面向对象、动态数据类型的高级程序设计语言



解释型语言的执行方式



编译型语言的执行方式

编译型	解释型	混合型
c	JavaScript	JAVA
c++	Python	C#
GO	Ruby	
Swift	PHP	
Object-C	Perl	
Pascal	Erlang	

数据类型

- 整数，正负整数,十六进制 (int)
- 浮点数，小数 (float)
- 字符串，用 “” 或 ‘’ 括起来的任意文本 (str)
- 字符串里有单引号或双引号，需要用转义字符 “\” 标识
- 布尔值，只有True和False两种值 (bool)
- 空值 (None),None != 0

整型

- 整型 (int)
- 整型的取值为整数，有正有负，如 2，-666，666 等。

```
In [10]: ▶ a = 123  
         type(a)
```

```
Out[10]: int
```

浮点型 (*float*)

- 浮点型的取值为小数，当计算有精度要求时被使用,由于小数点可以在相应的二进制的不同位置浮动，故而称为浮点数
- 如 3.14，-6.66 等，但是如果是非常大或者非常小的浮点数，就需要使用科学计数法表示，用 *e* 代替 10。

```
In [11]: ▶ num = 3.12e5  
         print(num)  
         type(num)
```

```
312000.0
```

```
Out[11]: float
```

字符串 (*str*)

- 字符串 (*str*)
- 字符串是以两个单引号或两个双引号包裹起来的文本

```
: ▶ name = "Jack"  
    type(name)
```

```
[12]: str
```

- 转义字符：字符串里常常存在一些如换行、制表符等有特殊含义的字符，这些字符称之为转义字符
- 比如 `\n` 表示换行，`\t` 表示制表符，Python还允许用 `r'''` 表示 `'''` 内部的字符串默认不转义

```
▶ print("I like Autumn")
print("I like Beijing's Autumn")
print("I like \n Beijing's Autumn")
print("I like \t Beijing's Autumn")
print(r"I like \n Beijing's Autumn")
```

```
I like Autumn
I like Beijing's Autumn
I like
  Beijing's Autumn
I like   Beijing's Autumn
I like \n Beijing's Autumn
```


布尔型 (*bool*)

- 布尔型 (*bool*)
- 布尔型只有 *True* 和 *False* 两种值。比较运算和条件表达式都会产生 *True* 或 *False*

```
Age = 18
nickname = "Bob"
print("Xiaoming's age is", Age)
print("After two years Xiaoming's age is", Age+2)
print("Xiaoming's age is more than 20 years old", Age > 20)
print("Xiaoming's age is less than 20 years old", Age < 20)
print("Xiaoming's nickname is Jack", nickname == "Jack")
```

```
Xiaoming's age is 18
After two years Xiaoming's age is 20
Xiaoming's age is more than 20 years old False
Xiaoming's age is less than 20 years old True
Xiaoming's nickname is Jack False
```

- 布尔型 (bool)
- 布尔值可以进行 *and*、*or* 和 *not* 运算, *and* 和 *or* 运算分别用 *&* 和 *|* 表示

and 运算

	True	False
True	True	False
False	False	False

or 运算

	True	False
True	True	True
False	True	False

- 布尔型 (bool)
- *not* 运算为非运算，即把 *True* 变成 *False*，*False* 变成 *True*。

```
print("True & False equals to", True & False)
print("True or False equals to", True | False)
print("Not True equals to", not True)
print("Not False equals to", not False)
```

```
True & False equals to False
True or False equals to True
Not True equals to False
Not False equals to True
```

空值 (*NoneType*)

- 空值是Python里一个特殊的值，用 *None* 表示，一般用 *None* 填充表格中的缺失值
- 使用 *type()* 函数来获取某值的类型

```
▶ print(type(2))  
print(type(3.1415))  
print(type("Xiaoming"))  
print(type(True))  
print(type(False))
```

```
<class 'int'>  
<class 'float'>  
<class 'str'>  
<class 'bool'>  
<class 'bool'>
```

日期数据

- 时间日期又是一种特有的格式 (`<class 'datetime.datetime'>`) , 这种格式不像我们常见数据格式容易操作, 在使用的時候有诸多不便。
- 例如我们想改变它的显示样式, 或者按照一定的年、月等特性进行分类。但是我们可以对`datetime`进行格式转换后操作。

```
from datetime import datetime
from datetime import timedelta

#生成一个现在时间日期类型
now = datetime.now()
delta = now - datetime(2020, 1, 1, 10, 10, 10)
print(now, type(now))
print(delta, type(delta))
```

```
2020-10-24 20:00:03.346918 <class 'datetime.datetime'>
297 days, 9:49:53.346908 <class 'datetime.timedelta'>
```

变量和常量

- 变量用来存储一些之后可能会变化的值

```
▶ age = 18#命名一个新变量age  
print(age)  
18
```

- 常量表示固定值，常量表示“不能变”的变量
- Python中是没有常量的关键字的，只是我们常常约定使用大写字母组合的变量名表示常量，也有不要对其进行赋值”的提醒作用

```
▶ pi = 3.14  
print(pi)
```

3.14

注释

- 如同我们在看书时做笔记一样
- Python语言会通过注释符号识别出注释的部分，将它们当做纯文本，并在执行代码时跳过这些纯文本
- 在Python语言中，使用 `#` 进行行注释

```
#我要命名一个新变量age  
age = 18#命名一个新变量age  
#新的变量age已经命名好了  
print(age)#打印变量值
```

18

注释

- 注释的目的是让人们能够轻松地读懂每一行代码，让人看了能知道这些代码的作用是什么
- 单行注释以“#” 开头
- 多行注释用3个单引号或3个双引号将注释括起来。

```
print("Hello World!")#我的第一行代码
```

```
Hello World!
```

```
age = 18#命名一个新变量age  
print(age)#打印变量age的结果  
"""
```

```
重新命名一个变量a, 并打印其类型  
"""
```

```
a = "123"  
type(a)
```

```
18
```

```
str
```


变量命名规则

- 变量名必须是大小写英文字母、数字或下划线 `_` 的组合，不能用数字开头，并且对大小写敏感
- 关键字不能用于命名变量（31个），如 `and`、`as`、`assert`、`break`、`class`、`continue`、`def`、`del` 等

```
import keyword
```

```
print(keyword.kwlist)
```

```
In [2]: import keyword
...:
...: print(keyword.kwlist)
['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']
```

变量赋值

- 通过赋值运算符 = 变量名和想要赋予变量的值连接起来，变量的赋值操作就完成了声明和定义的过程，在其他语言中需要制定类型；
- 同一变量可以反复赋值，而且可以是不同类型的变量，这也是Python语言称之为动态语言的原因

```
shot_id = 1
print(shot_id)
print(type(shot_id))
shot_id = '1'
print(shot_id);print(type(shot_id))
```

```
In [6]: shot_id = 1
...: print(shot_id);print(type(shot_id))
...: shot_id = '1'
...: print(shot_id);print(type(shot_id))
1
<class 'int'>
1
<class 'str'>
```

print函数

- 在Python 2.x版本中，同时兼容 `print` 和 `print()`
- 在Python 3.x版本中，`print` 函数为带括号的 `print()`
- 如果想要看变量的值，则直接在 `print` 后面加上变量名即可。
如果是想要输出提示信息，如一句话，那我们需要将提示信息用‘单引号包裹起来（这使得内容构成一个字符串）

```
In [8]: ▶ print("Hello World!") #我的第一行代码  
Hello World!
```

- 使用逗号,隔开变量与其他剩余内容, 则 `print` 在输出时会依次打印各个字符串或变量, 遇到逗号,时会输出一个空格
- `print` 函数不仅可以打印变量值, 也可以打印计算结果

```
In [9]: ▶ stu_id = 2
        print(stu_id) #打印stu_id值
        print(stu_id + 3) #shot_id + 3
        print("stu_id + 3 = ", stu_id + 3) #打印说明

2
5
stu_id + 3 = 5
```

算术运算符

- 二元数学运算符

运算	说明
$a + b$	a加b
$a - b$	a减b
$a * b$	a乘以b
a / b	a除以b
$a // b$	a除以b后向下圆整，丢弃小数部分
$a ** b$	a的b次方

- 整除：

```
In [21]: from __future__ import division
...: print("1/2 = ",1/2) #除法
...: print("1//2 = ",1//2)#向下取整
...: print("1/2.0 = ",1/2.0)
...: print("1//2.0 = ",1//2.0)
1/2 = 0.5
1//2 = 0
1/2.0 = 0.5
1//2.0 = 0.0
```

类型转换

- 函数 `int()`、`float()`、`str()` 和 `bool()` 分别用于将变量转换成整型、浮点型、字符串和布尔型变量

```
In [22]: print("shot_id is",type(shot_id))
...: print(type(float(shot_id)),float(shot_id))
...: print(type(str(shot_id)),str(shot_id))
...: print(type(bool(shot_id)),bool(shot_id))
shot_id is <class 'int'>
<class 'float'> 2.0
<class 'str'> 2
<class 'bool'> True
```


- 某些变量无法转换成数值型变量

```
In [24]: name = "Xiaoming"
...: print("The type is", type(name))
...: print(type(int(name)), int(name))
The type is <class 'str'>
Traceback (most recent call last):

  File "<ipython-input-24-326d2f089c89>", line 3, in <module>
    print(type(int(name)), int(name))
ValueError: invalid literal for int() with base 10: 'Xiaoming'
```

```
In [25]: print(type(int("2")), int("2"))
<class 'int'> 2
```


- 只有在变量值为 0 时, *bool* 转换的结果才为 *False*:

```
In [26]: name ="Xiaoming"
...: print("The type is",type(name))
...: print(type(bool(name)),bool(name))
The type is <class 'str'>
<class 'bool'> True
```

```
In [28]: print(type(int("2")),int("2"))
...: print(type(int("0")),int("0"))
<class 'int'> 2
<class 'int'> 0
```

- 除了使用 `type()` 外，我们还可以使用 `isinstance()` 来获得数据类型

```
In [30]: print(isinstance("2",str))  
        ...: print(isinstance(2,int))  
True  
True
```

```
In [32]: print(isinstance(2,str))  
False
```

Action

- 我的第一个爬虫程序 (python)
- 八爪鱼爬虫