实验四　数据保存

实验学时：4

实验要求：

掌握定义 item 的方法、使用 Feed exports 将数据保存到 csv、json 文件中的方法，掌握使用 pipeline 将数据保存到 MySQL 数据库中的方法。

。

实验内容：

1. 采集网页数据保存到文本文件。
   说明：访问古诗文网站（https://so.gushiwen.org/mingju/），会显示如图所示的页面，里面包含了很多名句，点击某一个名句（比如“山有木兮木有枝，心悦君兮君不知”），就会出现完整的古诗（如图所示）。

编写网络爬虫程序，爬取名句页面的内容，保存到一个文本文件中，然后，再爬取每个名句的完整古诗页面，把完整古诗保存到一个文本文件中。

【实验指导】

可以打开一个浏览器，访问要爬取的网页，然后在浏览器中查看网页源代码，找到诗句内容所在的位置，总结出它们共同的特征，就可以将它们全部提取出来了，具体实现参考代码如下：

```
#parse_poem.py
import requests
from bs4 import BeautifulSoup
import time

#函数 1：请求网页
def page_request(url,ua):
    response = requests.get(url,headers = ua)
    html = response.content.decode('utf-8')
    return html

#函数 2：解析网页
def page_parse(html):
    soup = BeautifulSoup(html,'lxml')
    title = soup('title')
    sentence = soup.select('div.left > div.sons > div.cont > a:nth-of-type(1)')
    poet = soup.select('div.left > div.sons > div.cont > a:nth-of-type(2)')
    sentence_list=[]
    href_list=[]
    for i in range(len(sentence)):
        temp = sentence[i].get_text()+ "---"+poet[i].get_text()
        sentence_list.append(temp)
        href = sentence[i].get('href')
```

```
            href_list.append("https://so.gushiwen.org"+href)
        return [href_list,sentence_list]

#函数 3：写入文本文件
def save_txt(info_list):
        import json
        with open(r'C:\\sentence.txt','a',encoding='utf-8') as txt_file:
            for element in info_list[1]:
                txt_file.write(json.dumps(element,ensure_ascii=False)+'\n\n')

#子网页处理函数：进入并解析子网页/请求子网页
def sub_page_request(info_list):
         subpage_urls = info_list[0]
         ua = {'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/46.0.2490.86 Safari/537.36'}
           sub_html = []
        for url in subpage_urls:
            html = page_request(url,ua)
            sub_html.append(html)
        return sub_html

#子网页处理函数：解析子网页，爬取诗句内容
def sub_page_parse(sub_html):
        poem_list=[]
        for html in sub_html:
            soup = BeautifulSoup(html,' lxml ')
            poem = soup.select('div.left > div.sons > div.cont > div.contson')
            poem = poem[0].get_text()
            poem_list.append(poem.strip())
        return poem_list

#子网页处理函数：保存诗句到 txt
def sub_page_save(poem_list):
        import json
        with open(r'C:\\poems.txt','a',encoding='utf-8') as txt_file:
            for element in poem_list:
                txt_file.write(json.dumps(element,ensure_ascii=False)+'\n\n')

if __name__ == '__main__':
        print("*****************开始爬取古诗文网站******************")
        ua = {'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/46.0.2490.86 Safari/537.36'}
        for i in range(1,4):
            url = 'https://so.gushiwen.org/mingju/default.aspx?p=%d&c=&t='%(i)
```

```
        time.sleep(1)
        html = page_request(url,ua)
        info_list = page_parse(html)
        save_txt(info_list)
        #处理子网页
        print("开始解析第%d"%(i)+"页")
        #开始解析名句子网页
        sub_html = sub_page_request(info_list)
        poem_list = sub_page_parse(sub_html)
        sub_page_save(poem_list)

    print("******************爬取完成**********************")
    print("共爬取%d"%(i*50)+"个古诗词名句，保存在如下路径：C:\\sentence.txt")
    print("共爬取%d"%(i*50)+"个古诗词，保存在如下路径：C:\\poem.txt")
```

2. 采集网页数据保存到 MySQL 数据库。

采集一个本地网页文件 web_demo.html，它记录了不同关键词的搜索次数排名，其内容如下：

```
<html>
<head><title>搜索指数</title></head>
<body>
<table>
<tr><td>排名</td><td>关键词</td><td>搜索指数</td></tr>
<tr><td>1</td><td>大数据</td><td>187767</td></tr>
<tr><td>2</td><td>云计算</td><td>178856</td></tr>
<tr><td>3</td><td>物联网</td><td>122376</td></tr>
</table>
</body>
</html>
```

在 Windows 系统中启动 MySQL 服务进程，打开 MySQL 命令行客户端，执行如下 SQL 语句创建数据库和表：

```
mysql > CREATE DATABASE webdb;
mysql > USE webdb;
mysql > CREATE TABLE search_index
mysql> create table search_index(
    -> id int,
    -> keyword char(20),
    -> number int);
```

编写网络爬虫程序，读取网页内容进行解析，并把解析后的数据保存到 MySQL 数据库中。

【实验指导】

参考代码如下：

```
# html_to_mysql.py
```

```
import requests
from bs4 import BeautifulSoup

# 读取本地 HTML 文件
def get_html():
    path = 'C:/web_demo.html'
    htmlfile= open(path,'r')
    html = htmlfile.read()
    return html

# 解析 HTML 文件
def parse_html(html):
    soup = BeautifulSoup(html,'html.parser')
    all_tr=soup.find_all('tr')[1:]
    all_tr_list = []
    info_list = []
    for i in range(len(all_tr)):
        all_tr_list.append(all_tr[i])
    for element in all_tr_list:
        all_td=element.find_all('td')
        all_td_list = []
        for j in range(len(all_td)):
            all_td_list.append(all_td[j].string)
        info_list.append(all_td_list)
    return info_list
 # 保存数据库
def save_mysql(info_list):
    import pymysql.cursors
    # 连接数据库
    connect = pymysql.Connect(
        host='localhost',
        port=3306,
        user='root',  # 数据库用户名
        passwd='123456',  # 密码
        db='webdb',
        charset='utf8'
    )

    # 获取游标
    cursor = connect.cursor()

    # 插入数据
    for item in info_list:
        id = int(item[0])
```

```
            keyword = item[1]
            number = int(item[2])
            sql = "INSERT INTO search_index(id,keyword,number) VALUES ('%d', '%s', %d)"
            data = (id,keyword,number)
            cursor.execute(sql % data)
            connect.commit()
        print('成功插入数据')

        # 关闭数据库连接
        connect.close()

if __name__ =='__main__':
        html = get_html()
        info_list = parse_html(html)
        save_mysql(info_list)
```

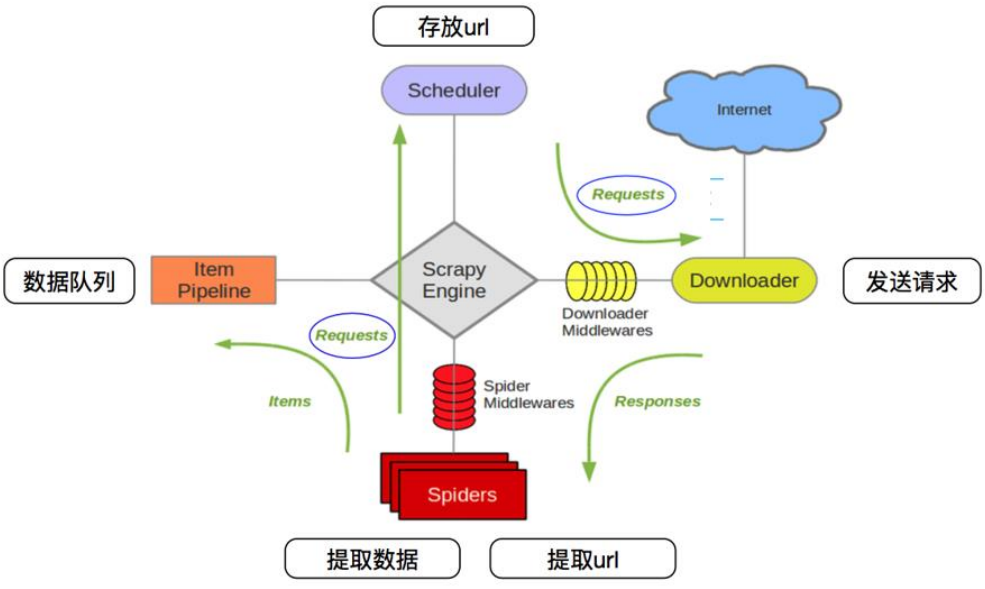
3. 采使用 Scrapy 框架爬取网页数据分别保存到文本文件和 MySQL 数据库中。

说明:

访问古诗文网站 (https://so.gushiwen.cn/mingjus/), 使用 Scrapy 框架编写爬虫程序, 爬取每个名句及其完整古诗内容，并把爬取到的数据分别保存到文本文件和 MySQL 数据库中。

【实验指导】

Scrapy 是一套基于 Twisted 的异步处理框架，是纯 Python 实现的爬虫框架，用户只需要定制开发几个模块就可以轻松地实现一个爬虫，用来抓取网页内容或者各种图片。Scrapy 运行于 Linux/Windows/MacOS 等多种环境，具有速度快、扩展性强、使用简便等特点。即便是新手，也能迅速学会使用 Scrapy 编写所需要的爬虫程序。Scrapy 可以在本地运行，也能部署到云端实现真正的生产级数据采集系统。Scrapy 用途广泛，可以用于数据挖掘、监测和自动化测试。其工作架构如下图:



主要的运行步骤如下：

①Scrapy 引擎从调度器中取出一个链接（URL）用于接下来的抓取；

②Scrapy 引擎把 URL 封装成一个请求并传给下载器；

③下载器把资源下载下来，并封装成应答包；

④爬虫解析应答包；

⑤如果解析出的是项目，则交给项目管道进行进一步的处理；

⑥如果解析出的是链接（URL），则把 URL 交给调度器等待抓取。

本实验主要完成以下步骤：

新建工程；安装 Scrapy

编写代码文件 items.py；

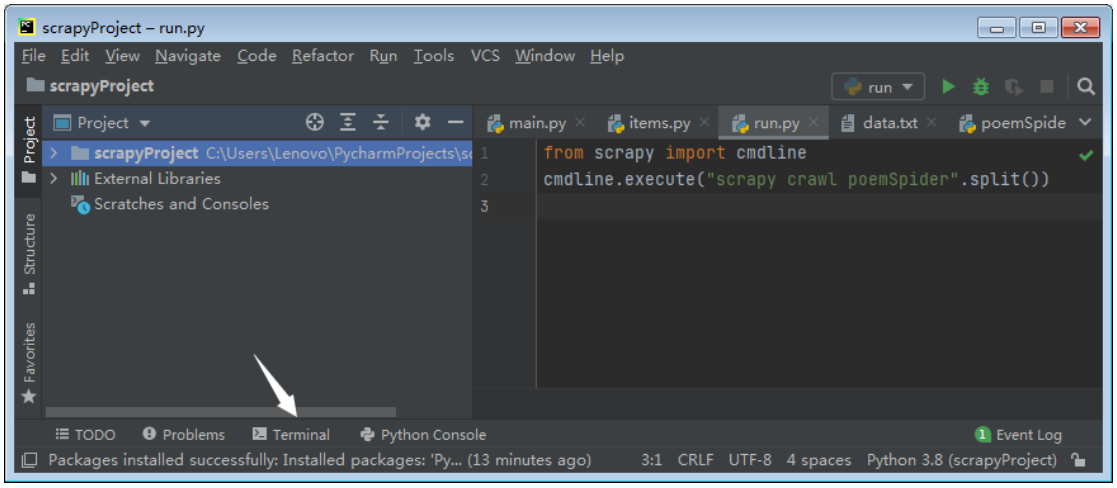
编写爬虫文件；

- 编写代码文件 pipelines.py；
- 编写代码文件 settings.py；
- 运行程序；

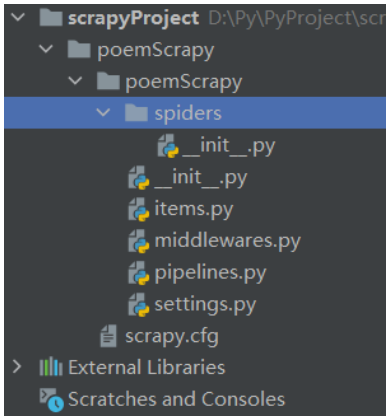
把数据保存到数据库中。

具体步骤说明如下：

一、在 PyCharm 中新建一个名称为“scrapyProject”的工程。在“scrapyProject”工程底部打开 Terminal 窗口（如图所示），在命令提示符后面输入命令“pip install scrapy”，下载 Scrapy 框架所需文件。



下载完成后，继续输入命令“scrapy startproject poemScrapy”，创建 Scrapy 爬虫框架相关目录和文件。创建完成以后的具体目录结构如图所示，这些目录和文件都是由 Scrapy 框架自动创建的，不需要手动创建。



在 Scrapy 爬虫程序目录结构中，各个目录和文件的作用如下：

- Spiders 目录：该目录下包含爬虫文件，需编码实现爬虫过程；
- __init__.py：为 Python 模块初始化目录，可以什么都不写，但是必须要有；
- items.py：模型文件，存放了需要爬取的字段；
- middlewares.py：中间件（爬虫中间件、下载中间件），本例中不用此文件；
- pipelines.py：管道文件，用于配置数据持久化，例如写入数据库；
- settings.py：爬虫配置文件；
- scrapy.cfg：项目基础设置文件，设置爬虫启用功能等。在本例中不用此文件。

二、编写代码文件 items.py

在 items.py 中定义字段用于保存数据，items.py 的具体参考代码内容如下：

```
import scrapy

class PoemscrapyItem(scrapy.Item):
    # 名句
    sentence = scrapy.Field()
    # 出处
    source = scrapy.Field()
    # 全文链接
    url = scrapy.Field()
    # 名句详细信息
    content = scrapy.Field()
```

三、编写爬虫文件

在 Terminal 窗口输入命令“cd poemScrapy”，进入对应的爬虫工程中，再输入命令“scrapy genspider poemSpider gushiwen.cn”，这时，在 spiders 目录下会出现一个新的 Python 文件 poemSpider.py，该文件就是我们要编写爬虫程序的位置。下面是 poemSpider.py 的参考代码：

```
import scrapy
from scrapy import Request
from ..items import PoemscrapyItem

class PoemspiderSpider(scrapy.Spider):
    name = 'poemSpider'    # 用于区别不同的爬虫
    allowed_domains = ['gushiwen.cn']  # 允许访问的域
    start_urls = ['http://so.gushiwen.cn/mingjus/']  # 爬取的地址
def parse(self, response):
        # 先获每句名句的 div
        for box in response.xpath('//*[@id="html"]/body/div[2]/div[1]/div[2]/div'):
            # 获取每句名句的链接
            url = 'https://so.gushiwen.cn' + box.xpath('.//@href').get()
            # 获取每句名句内容
            sentence = box.xpath('.//a[1]/text()') .get()
```

```
            # 获取每句名句出处
            source = box.xpath('.//a[2]/text()') .get()
            # 实例化容器
            item = PoemscrapyItem()
            # # 将收集到的信息封装起来
            item['url'] = url
            item['sentence'] = sentence
            item['source'] = source
            # 处理子页
            yield scrapy.Request(url=url, meta={'item': item}, callback=self.parse_detail)
        # 翻页
        next = response.xpath('//a[@class="amore"]/@href'). get()
        if next is not None:
            next_url = 'https://so.gushiwen.cn' + next
            # 处理下一页内容
            yield Request(next_url)
def parse_detail(self, response):
        # 获取名句的详细信息
        item = response.meta['item']
        content_list = response.xpath('//div[@class="contson"]//text()').getall()
        content = "".join(content_list).strip().replace('\n', '').replace('\u3000', '')
        item['content'] = content
        yield item
```

四、编写代码文件 **pipelines.py**

当我们成功获取需要的信息后，要对信息进行存储。在 Scrapy 爬虫框架中，当 item 被爬虫收集完后，将会被传递到 pipelines。现在要将爬取到的数据保存到文本文件中，可以使用的 pipelines.py 代码：

```
import json

class PoemscrapyPipeline:
    def __init__(self):
        # 打开文件
        self.file = open('data.txt', 'w', encoding='utf-8')

    def process_item(self, item, spider):
        # 读取 item 中的数据
        line = json.dumps(dict(item), ensure_ascii=False) + '\n'
        # 写入文件
        self.file.write(line)
        return item
```

五、编写代码文件 settings.py

```
BOT_NAME = 'poemScrapy'

SPIDER_MODULES = ['poemScrapy.spiders']
NEWSPIDER_MODULE = 'poemScrapy.spiders'

USER_AGENT = 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/90.0.4421.5 Safari/537.36'

# Obey robots.txt rules
ROBOTSTXT_OBEY = False

#  设置日志打印的等级
LOG_LEVEL = 'WARNING'

ITEM_PIPELINES = {
    'poemScrapy.pipelines.PoemscrapyPipeline': 1,
}
```

其中，更改 USER-AGENT 和 ROBOTSTXT_OBEY 是为了避免访问被拦截或出错；设置 LOG_LEVEL 是为了避免在爬取过程中显示过多的日志信息；设置 ITEM_PIPELINES 是因为本案例使用到 pipeline，需要先注册 pipeline，右侧的数字'1'为该 pipeline 的优先级，范围 1-1000，数值越小越优先执行。也可以根据实际需求，适当更改 settings.py 中的内容。

六、**运行程序**

有两种执行 Scrapy 爬虫的方法，第一种是在 Terminal 窗口中输入命令"scrapy crawl poemSpider"，然后回车运行，等待几秒钟后即可完成数据的爬取。第二种是在 poemScrapy 目录下新建 Python 文件 run.py（run.py 应与 scrapy.cfg 文件在同一层目录下），并输入下面代码：

```
from scrapy import cmdline
cmdline.execute("scrapy crawl poemSpider".split())
```

在 run.py 代码区域点击鼠标右键，在弹出的菜单里选择"Run"运行代码，就可以执行 Scrapy 爬虫程序。执行成功以后，就可以看到生成的数据文件 data.txt，其内容**类似**如下：

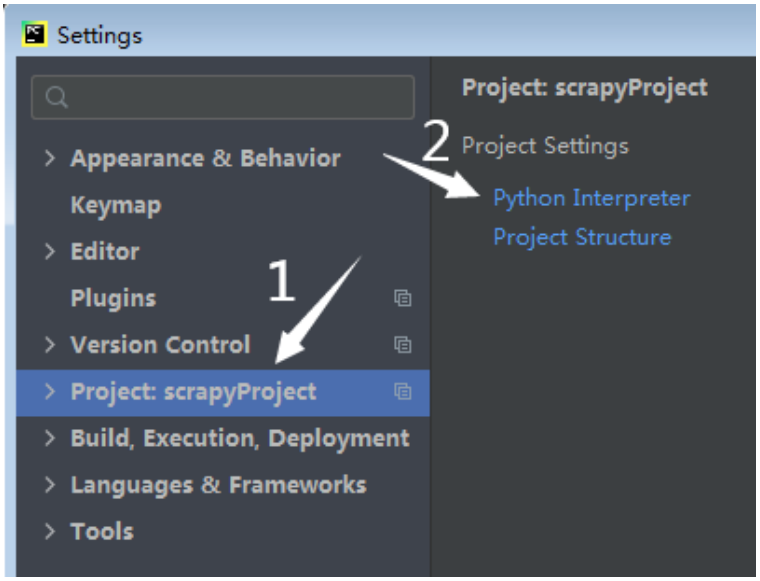
```
{"url": "https://so.gushiwen.cn/mingju/juv_2f9cf2c444f2.aspx", "sentence": "人道恶盈而好谦。", "source": "《易传·彖传上·谦》", "content":
```

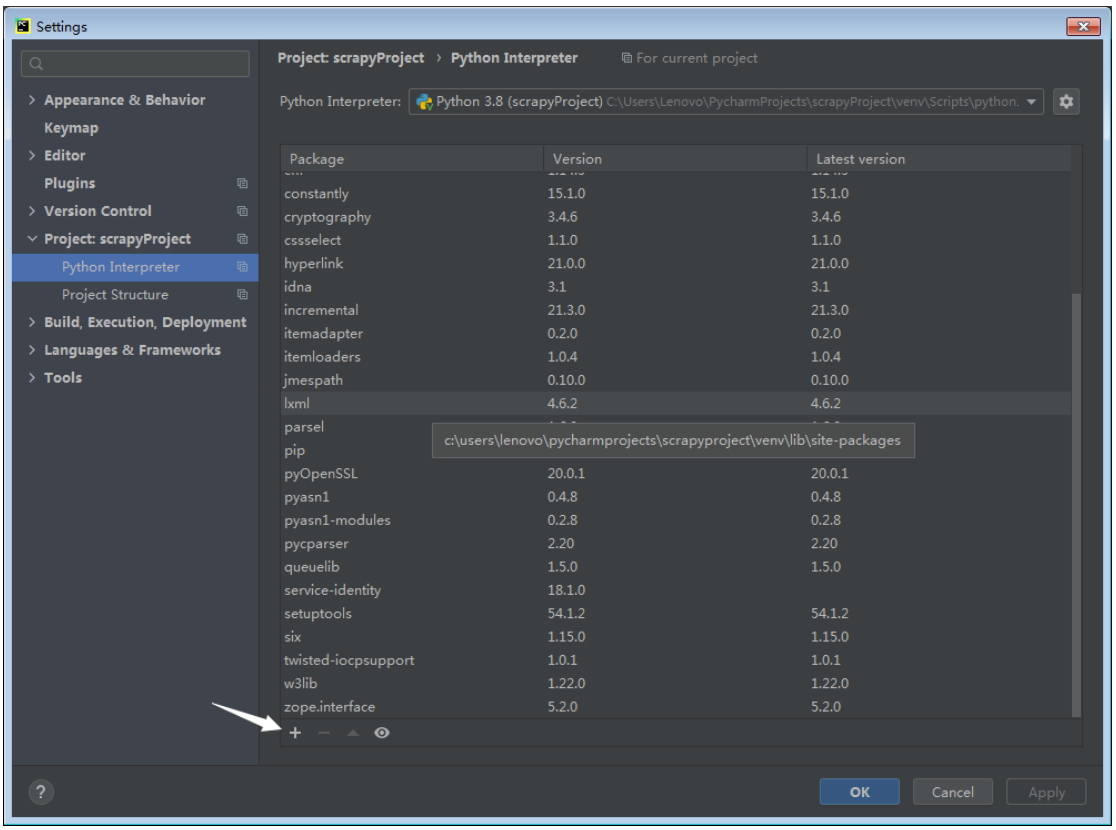
七、把数据保存到数据库中

为了把爬取到的数据保存到 MySQL 数据库中，需要首先安装 PyMySQL 模块。

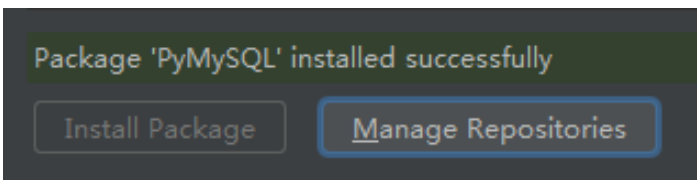
在 PyCharm 开发界面中点击"File->Settings…"，在打开的设置界面中（如图 3-10 所

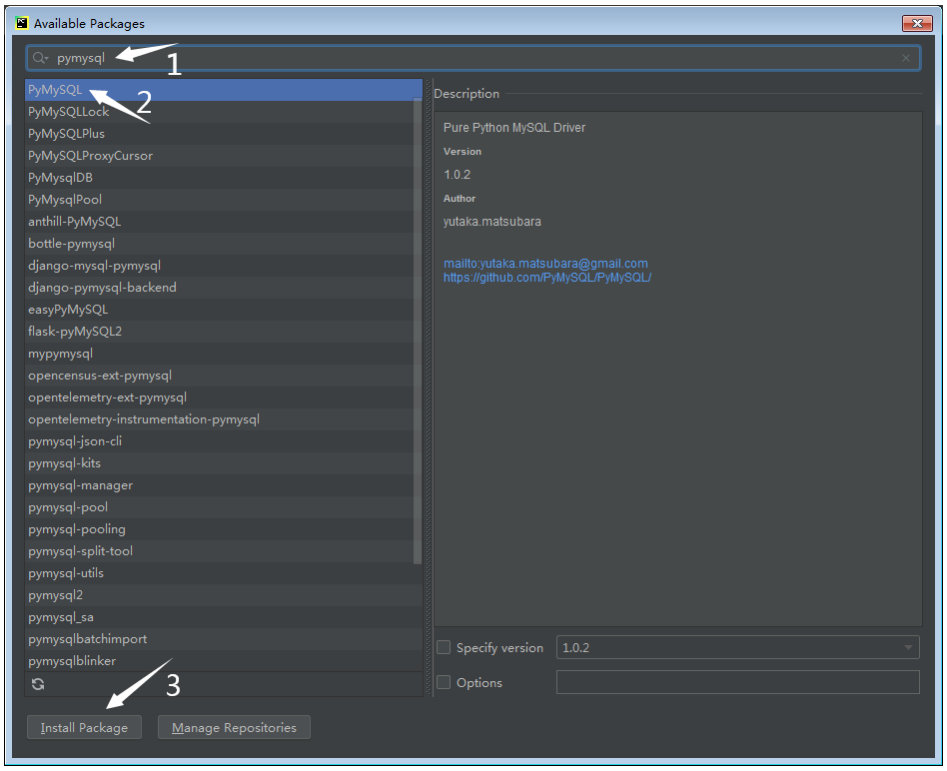
示），先点击“Project scrapyProject”，再点击“Python Interpreter”，会弹出如图所示的设置界面，点击界面底部的“+”。





在弹出的模块安装界面中（如下图二所示），先在搜索框中输入“pymysql”，然后，在搜索到的结果中点击“PyMySQL”条目，最后，点击界面底部的“Install Package”，开始安装模块，如果安装成功，会出现如下图所示的信息。





在 Windows 系统中启动 MySQL 服务进程，然后，打开 MySQL 命令行客户端，执行如下 SQL 语句创建一个名称为“poem”的数据库：

```
CREATE DATABASE poem;
```

然后，在 poem 数据库中创建一个名称为“beautifulsentence”的表，具体 SQL 语句如下：

```
DROP TABLE IF EXISTS `beautifulsentence`;
CREATE TABLE `beautifulsentence` (
  `source` varchar(255) NOT NULL,
  `sentence` varchar(255) NOT NULL,
  `content` text NOT NULL,
  `url` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

修改 pipelines.py，编写完成以后的 pipelines.py 代码如下：

```
from itemadapter import ItemAdapter
import json
import pymysql

class PoemscrapyPipeline:
    def __init__(self):
```

```
            # 连接 MySQL 数据库
            self.connect = pymysql.connect(
                host='localhost',
                port=3306,
                user='root',
                passwd='123456',  #设置成用户自己的数据库密码
                db='poem',
                charset='utf8'
            )
            self.cursor = self.connect.cursor()

        def process_item(self, item, spider):
            # 写入数据库
            self.cursor.execute('INSERT  INTO  beautifulsentence(source,sentence,content,url)
VALUES ("{}","{}","{}","{}")'.format(item['source'], item['sentence'], item['content'], item['url']))
            self.connect.commit()
            return item
def close_spider(self, spider):
        # 关闭数据库连接
            self.cursor.close()
            self.connect.close()
```

执行 Scrapy 爬虫程序，执行结束以后，如果执行成功，可以到 MySQL 数据库中使用如下命令查看数据：

```
USE poem;
SELECT * FROM beautifulsentence;
```