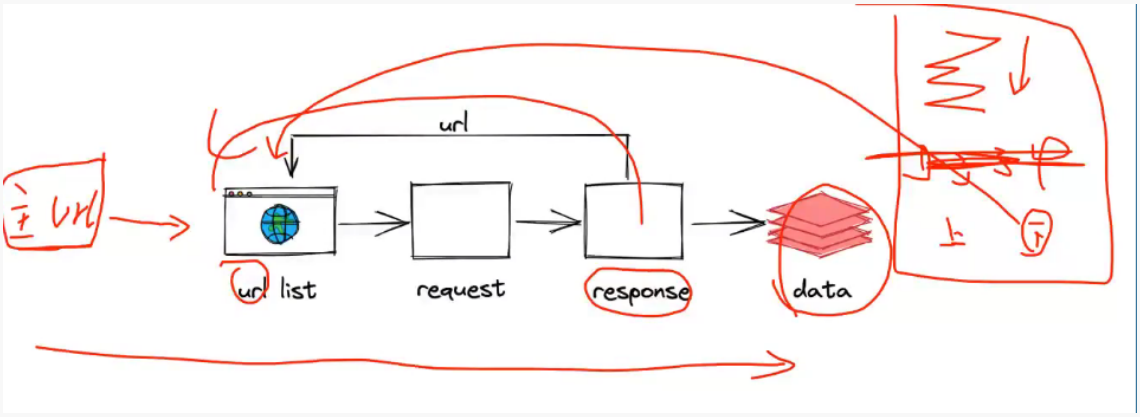
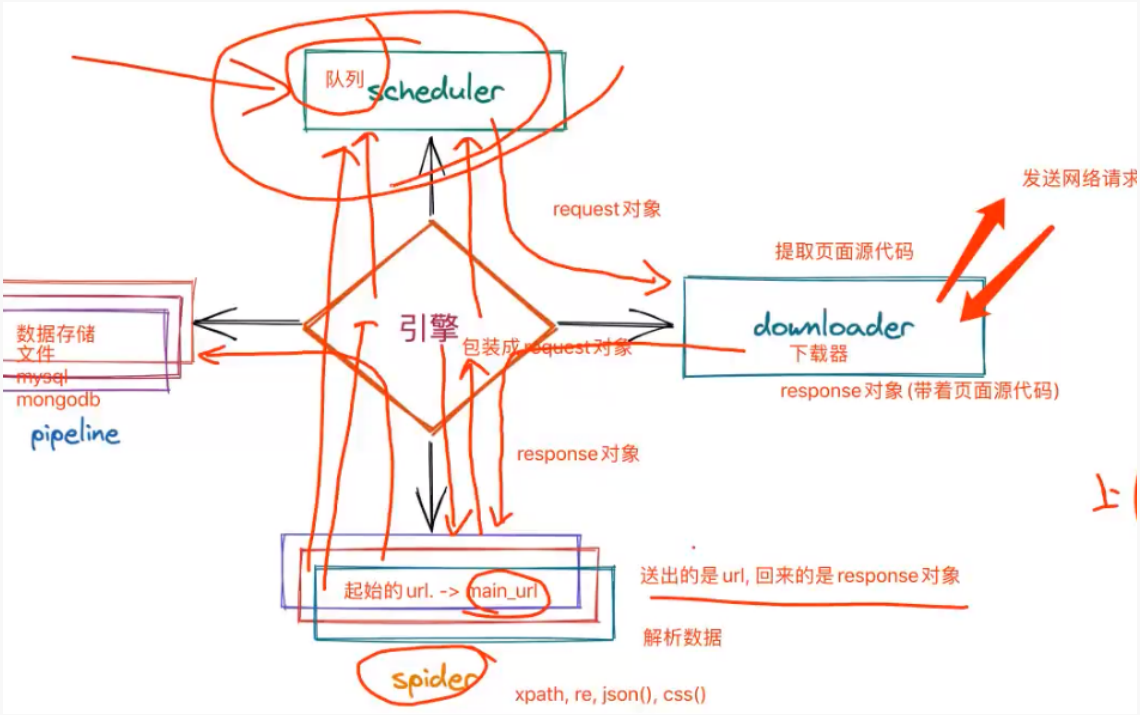
🍔 原理





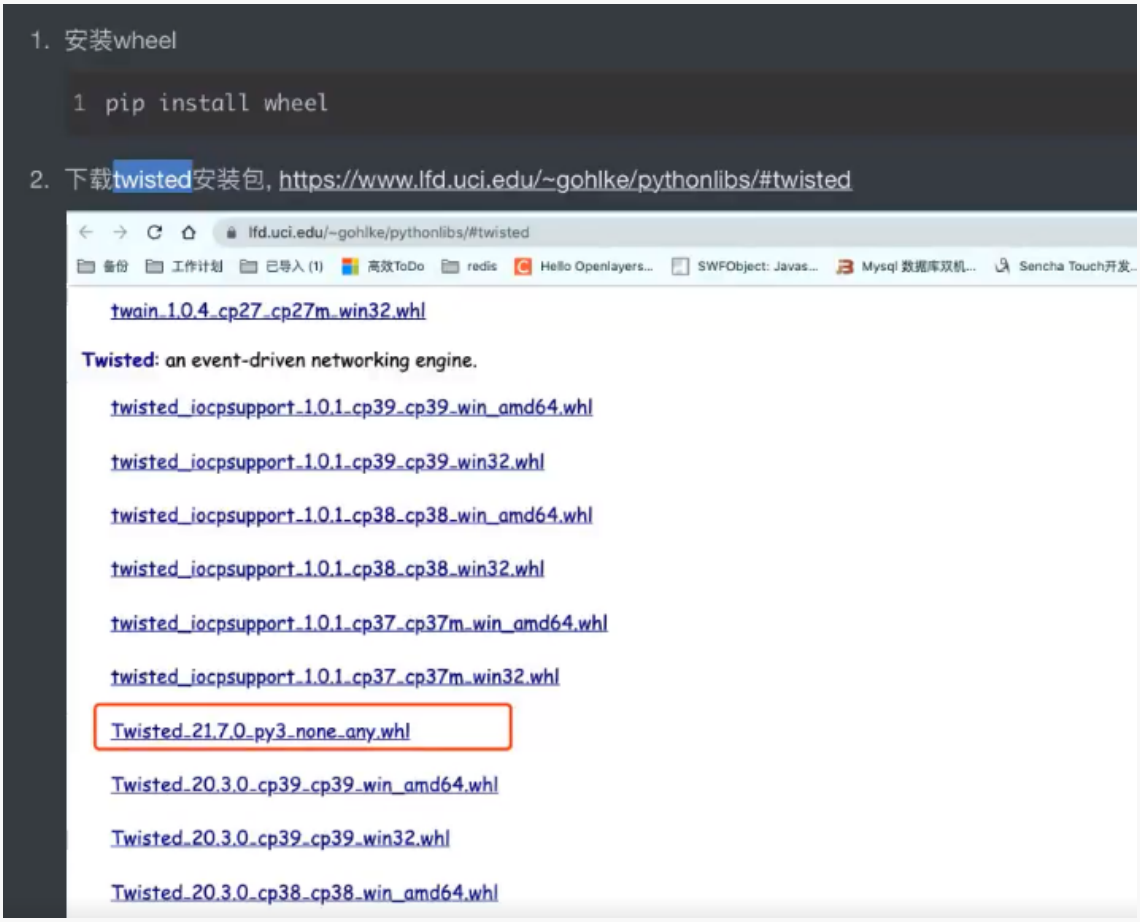
1. 爬虫中起始的url构造成request对象,并传递给调度器
2. 引擎 从 调度器 中获取到request对象.,然后交给 下载器
3. 由 下载器 来获取到页面源代码,并封装成response对象. 并回馈给 引擎
4. 引擎 将获取到的response对象传递给 spider,由 spider 对数据进行解析(parse).并回馈给引擎
5. 引擎 将数据传递给pipeline进行数据持久化保存或进一步的数据处理.
6. 在此期间如果spider中提取到的并不是数据,而是子页面ur.可以进一步提交给调度器,进而重复 步骤2 的过程

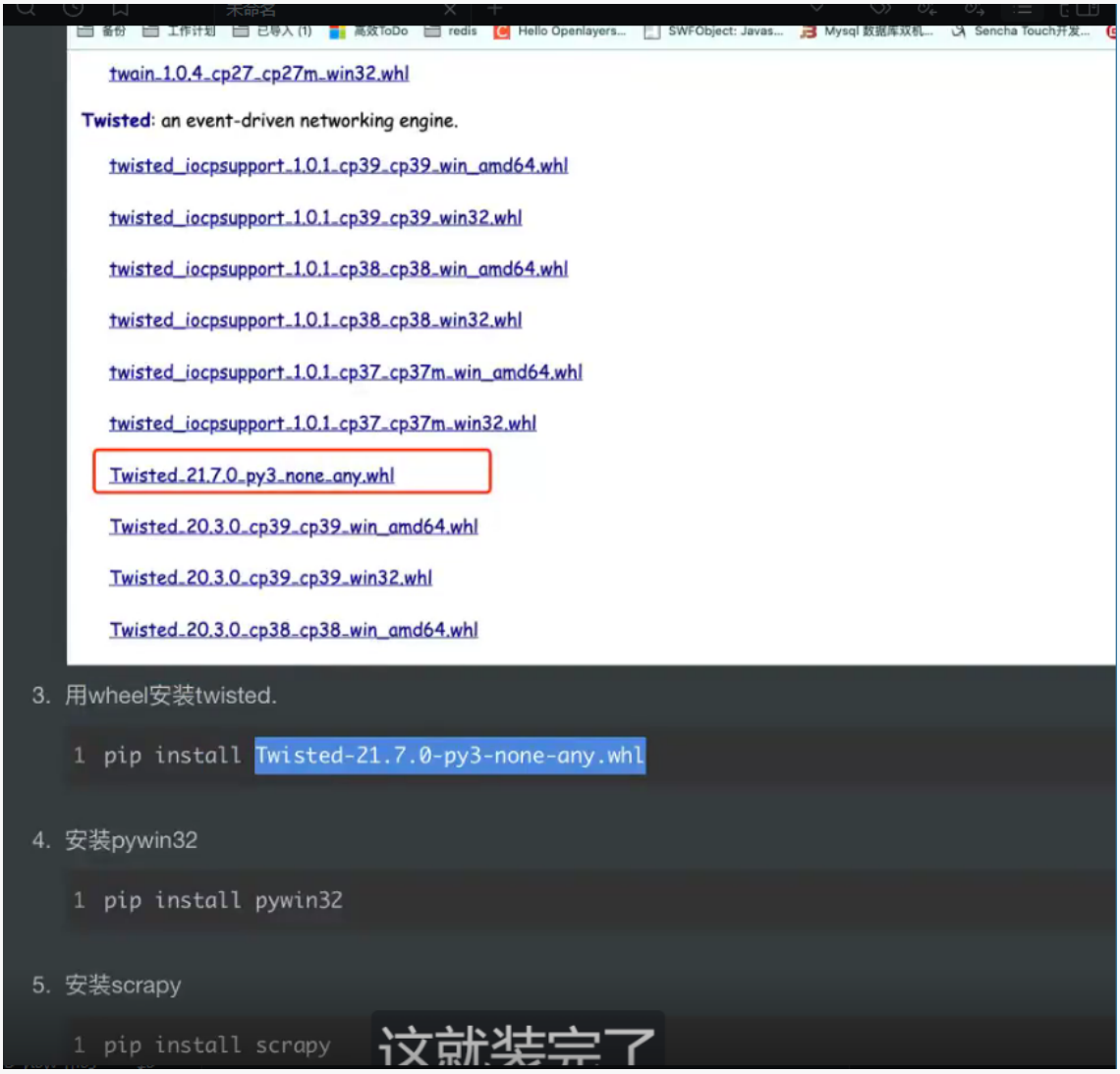
安装

```
pip install scrapy
```

如果安装成功,直接去创建项目即可.如果报错可能需要安装VC++14.0库才可以, 根据报错信息进行一点点的调整,多试几次pip.直至success.
如果上述过程还是无法正常安装scrapy,可以考虑用下面的方案来安装

如果还是不行的话






```
scrapy version
```

🍔 启动

右键我们的文件夹，找到open in terminal

```
scrapy startproject game
```

这个spider就是我们图上的底下的那个东西

items，它是为了我们 script 去封装比较大的这种数据的时候，我们会用到这个，如果你封你的数据很小，就像咱们今天要抓这个 4399 的这个往那而言，因为我们的目标很小，就有游戏名、游戏类型以及这个发布时间，就三个字段，对吧？就三个我们要存储的东西，所以用不上items，

middlewares，中间件

pipelines，数据存储

settings，是配置信息

```
cd game
scrapy genspider xiao 4399.com
```

这个就不是4399.com


```
scrapy crawl xiao
```

这个也是会打印日志的，如果我们不想打印出日志的话

LOG_LEVEL = "WARNING"

> 就是说只有WARNING及其以上的才可以被进行打印

日志的级别：DEBUG，INFO，WARNING，ERROR，CRITICAL

🍔 基础使用入门

```
import scrapy


class XiaoSpider(scrapy.Spider):
    name = "xiao"   #爬虫的名字
    allowed_domains = ["4399.com"]# 允许的域名
    start_urls = ["https://www.4399.com/"]# 起始页面的url，并不是我们允许的域名，是子级，#是我们要抓取的url

    def parse(self, response):# 默认是用来解析的
        # pass
        print(response)
        print(response.text)
        # 提取数据(不用进行导包)
        # response.json()
        # response.xpath()
        # response.css()
        txt = response.xpath("//ul[@class='tm_list']/li/a/text()").extract()# 这个xpath我们是在调试工具中的# 这里返回的是一个列表
        print(txt)

        # 分块提取数据
        li_list = response.xpath("//ul[@class='tm_list']/li")
        for li in li_list:
            # name = li.xpath("./a/text()").extract()[0]# 这样写不稳妥，可能会进行报错
            name = li.xpath("./a/text()").extract_first()#如果没有，返回none
            print(name)
```

```
            dic = {
                "name":name
            }
            # 需要要使用yield传递给管道
            yield dic   # 如果返回的是数据，那么可以直接返给了管道pipeline
            # 如果是return的话，那么循环一次就直接返回了
        # 如果是在外面的话，返回一个列表的话，那么列表太大，不方便
```

```
# Define your item pipelines here
#
# Don't forget to add your pipeline to the ITEM_PIPELINES setting
# See: https://docs.scrapy.org/en/latest/topics/item-pipeline.html


# useful for handling different item types with a single interface
from itemadapter import ItemAdapter

# 记住，管道默认是不生效的，我们需要在设置中把这个给注释取消，就可以进行生效了，
# ITEM_PIPELINES = {
#    key 就是管道的路径
#    value  就是管道的优先级
#    "game.pipelines.GamePipeline": 300,数越小，优先级越高
# }
class GamePipeline: # 随便定义了一个类
    def process_item(self, item, spider): # 这个item就是我们传递过来的数据,spider是爬虫
        print(item)
        print(spider.name)
        return item # return到下一个管道

# 下面这个是我们自己定义的
class NewPipeline:
    def process_item(self, item, spider): # 这个item就是我们传递过来的数据,spider是爬虫
        item['love'] = "我爱周杰伦"
        print(item)
        print(spider.name)
        return item # return到下一个管道
```

```
BOT_NAME = "game"

SPIDER_MODULES = ["game.spiders"]
NEWSPIDER_MODULE = "game.spiders"
# 大约在第15行
LOG_LEVEL = "WARNING"



ITEM_PIPELINES = {
```

```
    "game.pipelines.GamePipeline": 300,
    "game.pipelines.NewPipeline":299,
}
```