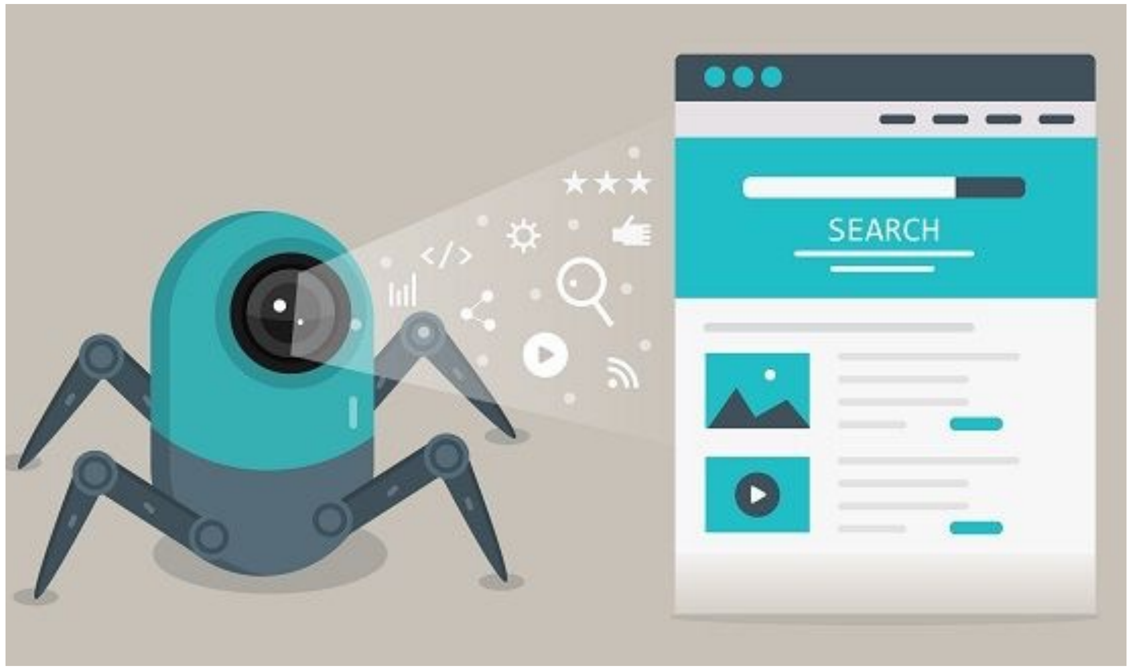
Web Crawler Project

Using Scrapy Framework in Python

Shubham Kumar Singh 15SCSE101020
Deep Anand 15SCSE101021

What is web crawler?

- Internet bot that systematically browses the World Wide Web.
- Sometimes called spider or spiderbot.
- Web search engines use web crawling.
- Web crawler download all the visited pages for later processing by search engines.
- Also used to gather specific type of information from web pages.



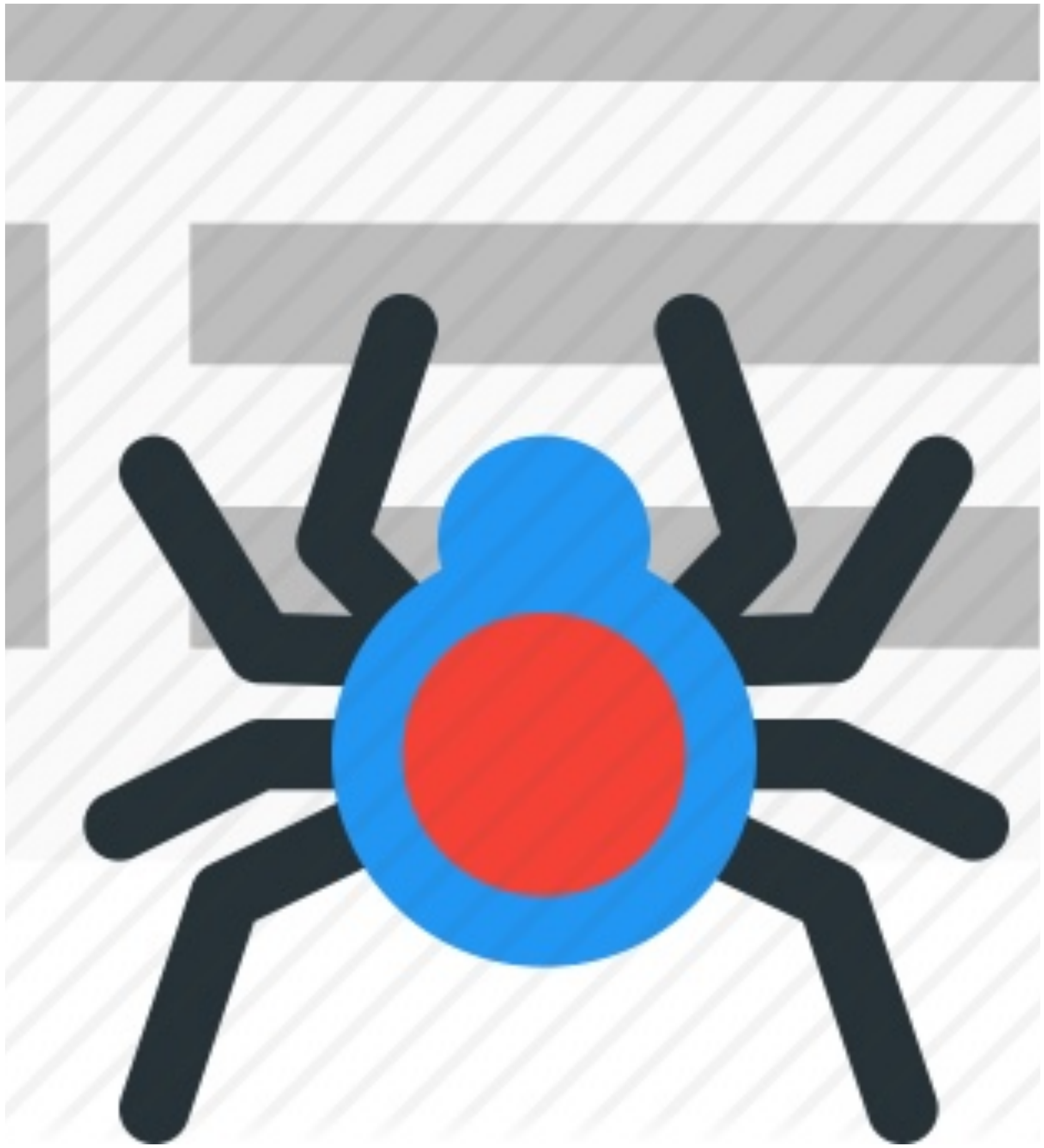
Motivation

- Widely used
- Great future scope
- Can be a unique product with extra features added
- Can gather crucial information

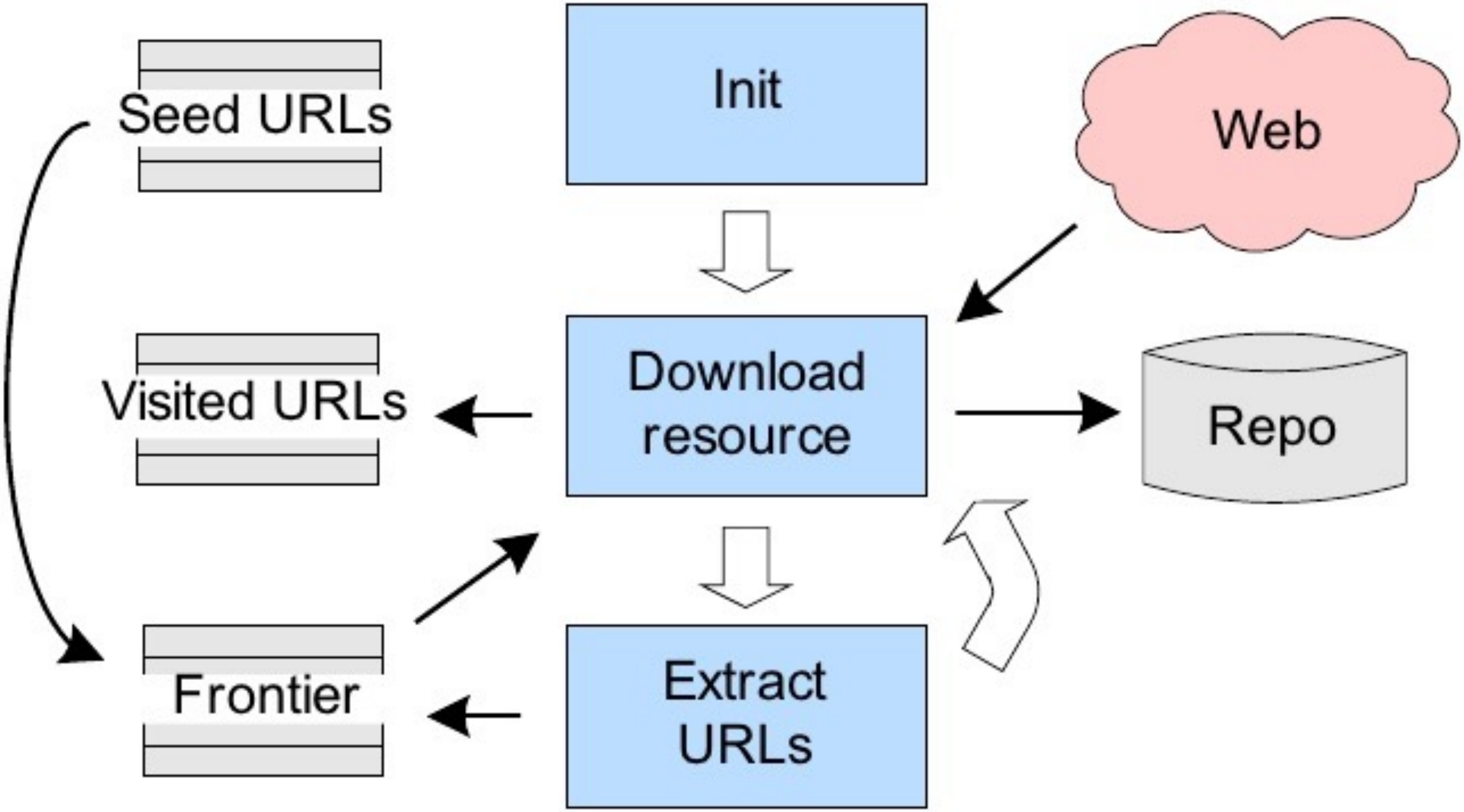
A key motivation for designing Web crawlers has been to retrieve Web pages and add their representations to a local repository.

Basic crawler operation

- Begin with known “seed” pages
- Fetch and parse them
- Extract URLs they point to
- Place the extracted URLs on a Queue
- Fetch each URL on a Queue and repeat



Traditional Web Crawler



Uses for crawling :

- Complete web search engine

  Search Engine = **Crawler** + indexer searcher + GUI

  - Find Stuff
  - Gather stuff
  - Check stuff

What is Scrapy?

- Scrapy is an open source and collaborative framework for extracting the data you need from websites. In a fast, simple yet extensible way.
- Scrapy is free and open source web-crawling framework written in Python.
- It is currently maintained by Scrapinghub Ltd.
- Scrapy was born at London-based web-aggregation and e-commerce company Mydeco, where it was developed and maintained by employees of Mydeco and Insophia (a web-consulting company based in Montevideo, Uruguay).

Steps used while working on Scrapy

- Download Anaconda from *www.anaconda.com*
- Install Scrapy using command: *$sudo -H pip install scrapy*
- In Anaconda created a new Environment called *ScrapyEnvironment*
- In Terminal used command: *$scrapy activate ScrapyEnvironment*
- In Terminal used command: *$scrapy startproject MyScraper*

- This command creates a folder to work with. In that folder navigate to the “spider” folder, that’s where we will be working.

- Open Anaconda app > Open Spyder.

- Navigate to File Explorer and open MyScraper > Spider.

- Create a new file with name FirstSpider.py.

- Write code in file.

Basic Spider in Python

```python
import scrapy
class QuotesSpider(scrapy.Spider):
    name = "quotes"
    def start_requests(self):
        urls = [
            'quotes.toscrape.com/page/1/',
            'quotes.toscrape.com/page/2/',
        ]
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse)

    def parse(self, response):
        page = response.url.split("/")[-2]
        filename = 'quotes-%s.html' % page
        with open(filename, 'wb') as f:
            f.write(response.body)
        self.log('Saved file %s' % filename)
```

Executing the Code(MacOs)

- After writing code open the terminal.
- To exit the zsh we use *$exec bash –login*
- Type the following command: *$source activate ScrapyEnvironment*
- Then navigate to the folder where we have our file which is inside the spider folder, using cd Desktop/…..

- Use command: *$scrapy crawl quotes*

- Remember that in above command quotes is used because we set the “name” variable in our FirstSpider.py file to “quotes”, see the above code.

- This will generate two files named “quotes-1.html” and “quotes-2.html”.

- We have successfully downloaded the website data and now can work on that data.

Scrapy Shell

We can use Scrapy Shell (provides interactive testing) in terminal which could come handy in many ways. For example if we want to run a quick command or view a webpage.

```
$scrapy shell
fetch("https://www.xyz.com")
view(response)
print(response.text)
```

The challenges of “Web Crawling”

There are three important characteristics of the web that makes crawling very difficult :

- Its large volume
- Its fast rate of change
- Dynamic pages generation