COURSE TITLE: **CEF 506 - PYTHON/PERL PROGRAMMING DEVELOPMENT**

NAME	MATRICULATION NUMBER
FRU ISIDORE CHE	FE12A078
THEOPHILUS WABA NASALI	FE12A183
NKENG NEWTON	FE12A140
MOBA MELVIS RINGNYU	FE12A107
ENOMBE THIERRY EWANE	FE12A053
ALANGI DERRICK	FE12A113

Title: Creation of a multi-threaded web spider

Hardware Specification

This application was developed and run on a machine with the following hardware requirement. It could be run on a machine with better specification

RAM: 3.00GB
Processor: AMD Phenom™ II N620 Dual-Core Processor
Processor speed: 2.80GHz
Hard drive: 1TB
Mark: Hewlett Packard (HP) ProBook 6455b

Software Specification

The following software used:

Operating System: Ubuntu 14.04
Editor: Sublime text editor 3
Command-line: Ubuntu terminal
Libraries:

Web crawler libraries needed

- **Scrapy** - a library used for web crawling.
  - The following are imported from the Scrapy library:
    Item, Selector, Field and BaseSpider

FLOWCHART

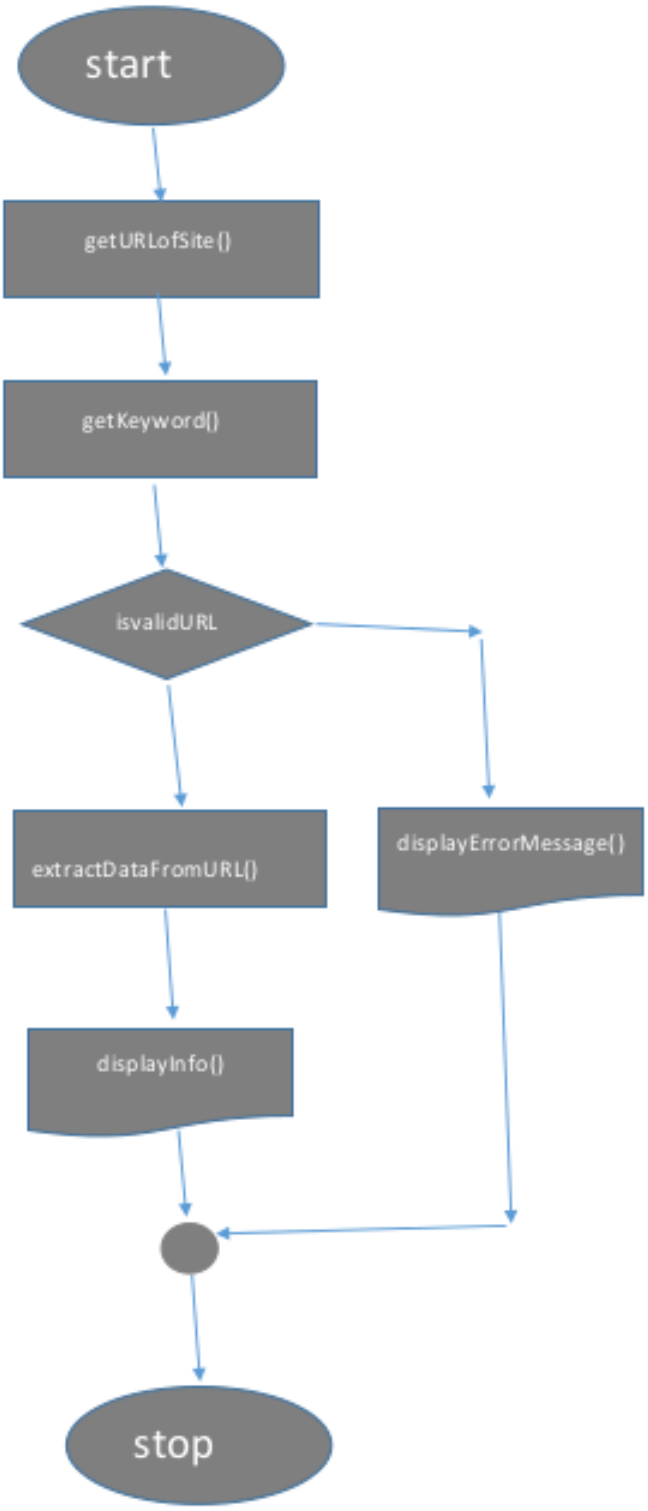

Figure 1: FlowChart of the spider webcrawler

CODE

```
from scrapy.item import Item, Field
from scrapy.selector import Selector
from scrapy.spider import BaseSpider


class Item(Item):
    # Items are defined in a declarative style. If you attempt to store a field
    # not defined here, an exception will be raised.
    title = Field()
    content = Field()
    links = Field()
    url = Field()


class SpiderMultithreaded(BaseSpider):
    """This spider crawls the website example.com."""
    # The name is the unique identifier for this spider.
    name = 'SpiderMultithreaded'
    # These URLs are the initial requests performed by the spider.
    start_urls = [
        'http://efarm.dev',
    ]

    # The default callback for the start urls is `parse`.
    # This method must return either items or requests.
    def parse(self, response):
        # Instance selector in order to query the html document.
        sel = Selector(response)
        # Instance our item. The item class have a dict-like interface.
        item = Item()
        # The method `extract()` always returns a list. So we extract the
        # first value with [0]. This is not needed when using the item loaders.
        # We can use a XPath rule to extract information from the html.
        item['title'] = sel.xpath('//h1/text()').extract()[0:]
        # Or we can use a CSS expression as well.
        item['content'] = sel.css('p::text').extract()[0:]
        item['links'] = sel.xpath('//a/text()').extract()[0:]
        item['url'] = response.url
```

Spaces:

Figure 2: Code snippet for spider webcrawler

OUTPUT

Figure 3: Snapshot of a section (content and links) of the web crawler output

Importance of Web crawling

Some of the importance of web crawling include:

- *Used to search for any important information on the web e.g jobs*
- *Improve the speed of a search engine*