

১) লগিং সম্পর্কে আরো জানতে হলে এখানে যেতে হবে :
<https://docs.python.org/3/howto/logging.html>। যারা প্রোগ্রামিংয়ে অপেক্ষাকৃত নতুন, তারা এটি পড়ে শুরুতে পুরোপুরি বুঝতে পারবে না, তবুও একবার পড়ে নিলে ভালো হয়। পাইথনে বেশ কয়েক মাস প্রোগ্রামিং করার পরে আরেকবার এটি পড়লে আগের চেয়ে অনেক বেশি বোৰা যাবে।

সম্পূর্ণ ওয়েব ক্রলার কোড

এখন আমি মূল ক্রলারের একটি আউটলাইন লিখব। এখানে বেশিরভাগ ফাংশনই এখনো ইমপ্লিমেন্ট করা হয়নি। ধীরে ধীরে আমরা ফাংশনগুলো তৈরি করব। বইটি এতদূর ঠিকভাবে পড়া হয়ে থাকলে কোড বুঝতে কোনো সমস্যা হবে না, কারণ কোডে নতুন কিছু নেই। এজন্য কোড নতুন করে ব্যাখ্যাও করব না। আমি কেবল দেখানোর চেষ্টা করব যে, কীভাবে একটু বড় প্রোগ্রাম আমরা তৈরি করতে পারি।

```
import logging
import sys
import requests
import re
import csv

def get_category_list(content):
    """get_category_list takes content of home page
    and returns a list of categories and their urls"""
    pass

def get_book_list(content):
    """get_book_list takes content of a book list page
    and returns a list of books (name and url)"""
    pass

def get_product_details(content):
    """get_product_details takes content of a product page,
    parses the page and returns details about a product"""
    pass

def get_page_content(url):
```

```
"""get_page_content takes a url and returns the content
of the page"""
pass

def get_next_page(content):
    """Checks the content of a book list page and returns
    link of the next page, returns None, if there is no
    more next page"""
    pass

def scrape_book_info(book_info, category_name):
    """ gets the content of a book details page,
    and parses different components and stores the info """
    pass

def crawl_category(category_name, category_url):
    """crawls a particular category of books"""
    while True:
        content = get_page_content(category_url)
        book_list = get_book_list(content)

        for book in book_list:
            scrape_book_info(book, category_name)

        if get_next_page(content) is None:
            break

def crawl_website():
    """crawl_website() is the main function that
    coordinates the whole crawling task"""
    url = "http://books.toscrape.com/index.html"
    host_name = "books.toscrape.com"

    content = get_page_content(url)
    category_list = get_category_list(content)

    for category in category_list:
        category_name, category_url = category
        crawl_category(category_name, category_url)
```

```

if __name__ == "__main__":
    logging.basicConfig(format='%(asctime)s %(message)s',
datefmt='%m/%d/%Y %I:%M:%S %p',
filename="bookstore_crawler.log", level=logging.DEBUG)

    with open("book_list.csv", "w") as csvf:
        csv_writer = csv.DictWriter(csvf,
fieldnames=["Name", "Category", "UPC", "URL", "ImageURL",
"Price", "Availability", "Description"])
        csv_writer.writeheader()

crawl_website()

```

ওপরের কোড পড়া শুরু করতে হবে এখান থেকে : if __name__ == "__main__":। আশা করি, প্রোগ্রামটি কীভাবে ডিজাইন করতে চাইছি, সেটি পাঠকদের বুবাতে কোনো সমস্যা হবে না। আর বিভিন্ন ফাংশন যেহেতু আমি ইমপ্লিমেন্ট করতে চাইনি, তাই pass স্টেটমেন্ট ব্যবহার করেছি। এটি লিখলে পাইথন বুবো নেয় যে, এখানে কিছু করা হচ্ছে না।

এখন আমরা একটু একটু করে পুরো কোড লিখব। আমি এখন কেবল যেসব জায়গায় পরিবর্তন হবে, সেসব জায়গার কোড দেখাব। বই পড়ার সঙে সঙে নিজের কম্পিউটারে প্রোগ্রাম লিখে ও রান করে দেখতে হবে।

```

def get_category_list(content):
    """get_category_list takes content of home page
    and returns a list of categories and their urls"""
    return category_pat.findall(content)

def get_page_content(url):
    """get_page_content takes a url and returns the content
    of the page"""
    try:
        response = requests.get(url)
    except requests.exceptions.RequestException as e:
        logging.error(e)

    if response.ok:
        return response.text

    logging.error("Can not get content from URL:" + url)

```

```

return None

def crawl_website():
    """crawl_website() is the main function that
    coordinates the whole crawling task"""
    url = "http://books.toscrape.com/index.html"
    host_name = "books.toscrape.com"

    content = get_page_content(url)
    if content is None:
        logging.critical("Failed to get content from " +
url)
        sys.exit(1)

    category_list = get_category_list(content)

    for category in category_list:
        category_url, category_name = category
        print(category_url)
        sys.exit(1)
        crawl_category(category_name, category_url)

if __name__ == "__main__":
    # Compile different regular expression patterns
    category_pat = re.compile(r'<li>\s*<a
href="(catalogue/category/books/.*?)">\s*(\w+[\s\w]+\w)\s*?
<', re.M | re.DOTALL)

    logging.basicConfig(format='%(asctime)s %(message)s',
datefmt='%m/%d/%Y %I:%M:%S %p',
filename="bookstore_crawler.log", level=logging.DEBUG)

    with open("book_list.csv", "w") as csvf:
        csv_writer = csv.DictWriter(csvf,
fieldnames=["Name", "Category", "UPC", "URL", "ImageURL",
"Price", "Availability", "Description"])
        csv_writer.writeheader()

    crawl_website()

```

ওপরের মতো করে আমরা কোডে পরিবর্তন করব। তার পরে প্রোগ্রামটি রান করলে আমরা নিশ্চিত হতে পারব যে, প্রোগ্রামটি ঠিকঠাক কাজ করছে। আমাদের `get_category_list()` ও `get_page_content()` ফাংশন দুটো তৈরি হয়ে গেল। আমরা প্রোগ্রাম রান করলে দেখতে পাব যে, ক্যাটাগরির লিংক দেখতে এরকম—

`catalogue/category/books/travel_2/index.html`

তাহলে পুরো লিংক (বা ইউআরএল) পেতে হলে আমাদেরকে "http://", `host_name`, "/" ও ক্যাটাগরি লিংক একসঙ্গে যুক্ত করতে হবে।

```
def crawl_website():
    """crawl_website() is the main function that
    coordinates the whole crawling task"""
    url = "http://books.toscrape.com/index.html"
    host_name = "books.toscrape.com"

    content = get_page_content(url)
    if content == "":
        logging.critical("Got empty content from " + url)
        sys.exit(1)

    category_list = get_category_list(content)

    for category in category_list:
        category_url, category_name = category
        category_url = "http://" + host_name + "/" +
category_url
        print(category_url)
        sys.exit(1)
        crawl_category(category_name, category_url)
```

এখন আমরা ওপরের মতো করে আমাদের কোড পরিবর্তন করব এবং রান করব। তাহলে নিচের মতো আউটপুট দেখব :

`http://books.toscrape.com/catalogue/category/books/travel_2
/index.html`

আমরা আউটপুট দেখে নিশ্চিত হয়ে নিলাম যে, একটি ক্যাটাগরির ইউআরএল ঠিকঠাক তৈরি করতে পারছি। এখন আমরা দেখব, একটি ক্যাটাগরির জন্য যদি একাধিক পেজ থাকে, সেগুলো আমরা ঠিকঠাক পাছি কি না। এজন্য আমরা `get_next_page()` ফাংশনটি তৈরি করব।

```
def get_next_page(url, content):
    """Checks the content of a book list page and returns
    link of the next page,
    returns None, if there is no more next page"""
    result = next_page_pat.findall(content)
    if len(result) == 0:
        return None
    i = url.rfind("/")
    return url[0:i+1] + result[0]
```

ফাংশনটি কাজ করছে কি না, সেটি বোঝার জন্য আমরা নিচের মতো করে `crawl_category()` ফাংশনটি একটু পরিবর্তন করে নেব। কিছু অংশ আমি তিনটি ডবল কোটেশন চিহ্ন দিয়ে শুরু ও শেষ করে দিয়েছি, যেন পাইথন সেই লাইনগুলোকে কমেন্ট মনে করে এবং না চালায়, কারণ আমরা সেগুলো এখন চালাতে চাই না।

```
def crawl_category(category_name, category_url):
    """crawls a particular category of books"""
    while True:
        print(category_url)
        content = get_page_content(category_url)
        """
        book_list = get_book_list(content)

        for book in book_list:
            scrape_book_info(book, category_name)
        """
        next_page = get_next_page(category_url, content)
        if next_page is None:
            break

        category_url = next_page
```

এখন আরেকটু কোড পরিবর্তন করতে হবে, যেন আমরা সরাসরি `crawl_category()` ফাংশনটি কল করি।

```

if __name__ == "__main__":
    # Compile different regular expression patterns
    category_pat = re.compile(r'<li>\s*<a href="(catalogue/category/books/.*)">\s*(\w+[\s\w]+\w)\s*?<', re.M | re.DOTALL)
    next_page_pat = re.compile(r'<li class="next"><a href="(.*)">next</a></li>')

    logging.basicConfig(format='%(asctime)s %(message)s',
                        datefmt='%m/%d/%Y %I:%M:%S %p',
                        filename="bookstore_crawler.log", level=logging.DEBUG)

    with open("book_list.csv", "w") as csvf:
        csv_writer = csv.DictWriter(csvf,
                                    fieldnames=["Name", "Category", "UPC", "URL", "ImageURL",
                                    "Price", "Availability", "Description"])
        csv_writer.writeheader()

        #crawl_website()
        crawl_category("mystery",
                        "http://books.toscrape.com/catalogue/category/books/mystery_3/index.html")

```

এখন প্রোগ্রাম রান করলে আমরা দেখতে পাব যে get_next_page() ফাংশন ঠিকভাবে কাজ করছে।

```

$ python book_crawler_final.py
http://books.toscrape.com/catalogue/category/books/mystery_3/index.html
http://books.toscrape.com/catalogue/category/books/mystery_3/page-2.html

```

পরীক্ষা করা হয়ে গেলে পরীক্ষার উদ্দেশ্যে আমরা যতটুকু কোড পরিবর্তন করেছিলাম, সেগুলো আবার ঠিকঠাক করে আগের অবস্থায় নিতে হবে। এখন আমরা get_book_list() ফাংশনটি লিখে ফেলব।

```

def get_book_list(content):
    """get_book_list takes content of a book list page
    and returns a list of books (name and url)"""
    content = content.replace("\n", " ")

```

```
return book_list_pat.findall(content)
```

আর যেখানে রেগুলার এক্সপ্রেশন প্যাটার্নগুলো লিখছি, সেখানে নিচের প্যাটার্নটি যোগ করব।

```
book_list_pat = re.compile(r'<h3><a href="(.*?)"  
title="(.*?)">' )
```

আমরা যদি প্রোগ্রাম রান করি, তাহলে আউটপুট দেখব একটি টাপলের লিস্ট। প্রতিটি টাপল দেখতে এরকম : ('.../.../.../1000-places-to-see-before-you-die_1/index.html', '1,000 Places to See Before You Die')। কিন্তু একটি বইয়ের পুরো লিংক (ইউআরএল) হচ্ছে এরকম : http://books.toscrape.com/catalogue/1000-places-to-see-before-you-die_1/index.html। তাহলে আমরা যে ইউআরএল পাচ্ছি, সেখান থেকে .../.../.../ বাদ দিয়ে দিতে হবে। তারপর যে অংশটুকু থাকবে, তাকে আমরা যুক্ত করব <http://books.toscrape.com/catalogue/>-এর শেষে।

এবারে আমরা `scrape_book_info()` ও `get_product_details()` ফাংশন দুটো তৈরি করে ফেলব। আর সেই সঙ্গে বই-সম্পর্কিত বিভিন্ন তথ্য পাওয়ার জন্য কিন্তু রেগুলার এক্সপ্রেশনও আমাদের লিখতে হবে। এজন্য আমি টার্মিনালে রেগুলার এক্সপ্রেশনগুলো আগে তৈরি করে নেব এবং সেই সঙ্গে পরীক্ষাও করে নেব যে, ওগুলো ঠিকঠাক কাজ করছে কি না।

```
>>> import requests  
>>> import re  
>>> url = "http://books.toscrape.com/catalogue/1000-places-  
to-see-before-you-die_1/index.html"  
>>> resp = requests.get(url)  
>>> content = resp.text  
>>> content = content.replace("\n", " ")  
>>> img_pat = re.compile(r'<div class="item active">\s*')  
>>>  
>>> img_pat.findall(content)  
['.../.../media/cache/9e/10/9e106f81f65b293e488718a4f54a6a3f.  
jpg']  
>>>  
>>> desc_pat = re.compile(r'<div id="product_description"  
class="sub-header">.*?<p>(.*?)</p>')  
>>> desc_pat.findall(content)
```

['Around the World, continent by continent, here is the best the world has to offer: 1,000 places guaranteed to give travelers the shivers. Sacred ruins, grand hotels, wildlife preserves, hilltop villages, snack shacks, castles, festivals, reefs, restaurants, cathedrals, ...more']

```

>>>
>>> upc_pat = re.compile(r'<th>UPC</th>\s*<td>(.*?)</td>')
>>> upc_pat.findall(content)
['228ba5e7577e1d49']

>>>
>>> price_pat = re.compile(r'<th>Price (incl.
tax)</th>\s*<td>(.*?)</td>')
>>> price_pat.findall(content)
[]

>>> price_pat = re.compile(r'<th>Price \ (incl.
tax)</th>\s*<td>(.*?)</td>')
>>> price_pat.findall(content)
['£26.08']

>>>
>>> price_pat = re.compile(r'<th>Price \ (incl.
tax)</th>\s*<td>\D+(\d+?)</td>')
>>> price_pat.findall(content)
[]

>>> price_pat = re.compile(r'<th>Price \ (incl.
tax)</th>\s*<td>\D+([\d.]+?)</td>')
>>> price_pat.findall(content)
['26.08']

>>>
>>> avail_pat =
re.compile(r'<th>Availability</th>\s*<td>(.*?)</td>')
>>> avail_pat.findall(content)
['In stock (1 available)']

```

ওপরের অংশটুকু নিজে নিজে চেষ্টা করতে হবে। এমনকি দেখে দেখে টাইপ করলেও হবে, কিন্তু শুধু পড়ে গেলে চলবে না। এখানে কয়েকটি বিষয় লক্ষ করতে হবে। প্রথমত, আমি desc_pat.findall(content)-এর আউটপুটের কিছু অংশ বাদ দিয়ে দিয়েছি, ইচ্ছা করেই। আউটপুট আরো বড় আসবে (পুরো বর্ণনাটুকু চলে আসবে)। তারপরে upc_pat ও price_pat-এ **\s*** না দিলেও চলত, কিন্তু তাও আমি দিয়ে রাখলাম, যদি কোথাও </th>

অধ্যায় ৮ - ওয়েব ক্রলিং (Web Crawling)

ও <td>-এর মাঝে স্পেস চলে আসে, সেই আশঙ্কায়। তৃতীয় বিষয় হচ্ছে, price_pat তৈরি করার সময় আমাকে বেশ কয়েকবার চেষ্টা করতে হয়েছে। (incl. tax)-এ যেহেতু প্যারেন্থেসিস রয়েছে, তাই আমরা \(\text{incl. tax}\) লিখে দিয়েছি। \(\text{d}\) দিয়ে বোঝাচ্ছি যে, এখানে লেফট প্যারেন্থেসিস ক্যারেষ্টার— ' (' থাকবে, \) দিয়ে বোঝাচ্ছি যে, এখানে রাইট প্যারেন্থেসিস ক্যারেষ্টার— ')' থাকবে। যেহেতু রেগুলার এক্সপ্রেশনে প্যারেন্থেসিস চিহ্নের আলাদা অর্থ রয়েছে, তাই এমনটি করতে হয়েছে। আর দামের অংশ থেকে কেবল 26.08 নিতে চাই। তার আগে কিছু ক্যারেষ্টার থাকবে, যেগুলো ডিজিট নয়। তাই লিখেছি \D+, অর্থাৎ এক বা একাধিক ক্যারেষ্টার যেগুলো ডিজিট নয়। তারপরে এক বা একাধিক ক্যারেষ্টার থাকবে যেগুলো ডিজিট কিংবা ডট। তাই লিখেছি [\d.]+। আরেকটি বিষয় হচ্ছে, আমরা ইমেজের লিংক পাচ্ছি এরকম :

```
'.../..../media/cache/9e/10/9e106f81f65b293e488718a4f54a6a3f.jpg'
```

কিন্তু পুরো ইমেজের লিংক (ব্রাউজারে ইমেজের ওপর রাইট ক্লিক করলে Copy Image Address অপশন আসবে, সেটিতে ক্লিক করলে ইমেজের পুরো লিংকটি পাওয়া যাবে) হচ্ছে :

```
http://books.toscrape.com/media/cache/9e/10/9e106f81f65b293e488718a4f54a6a3f.jpg
```

তাহলে ইমেজের পুরো লিংকটি কীভাবে তৈরি করব, তা নিশ্চয়ই বলে দিতে হবে না?

এখন তাহলে get_product_details() ফাংশনটি তৈরি করে ফেলা যাক।

```
def get_product_details(content):
    """get_product_details takes content of a product page,
    parses the page and returns details about a product"""
    image_base = "http://books.toscrape.com/"
    result = img_pat.findall(content)
    if len(result) == 0:
        logging.warn("Image url not found!")
        image_url = ""
    else:
        img_url = result[0]
        img_url = img_url.replace("../../", "")
        image_url = image_base + img_url
    result = desc_pat.findall(content)
```

```

if len(result) == 0:
    logging.warn("Description not found!")
    description = ""
else:
    description = result[0]

result = upc_pat.findall(content)
if len(result) == 0:
    logging.warn("UPC not found!")
    upc = ""
else:
    upc = result[0]

result = price_pat.findall(content)
if len(result) == 0:
    logging.warn("Price not found!")
    price = ""
else:
    price = result[0]

result = avail_pat.findall(content)
if len(result) == 0:
    logging.warn("Availability not found!")
    availability = ""
else:
    availability = result[0]

return upc, price, image_url, availability, description

```

আর `scrape_book_info()` ফাংশনটি দাঁড়াবে এমন :

```

def scrape_book_info(book_info, category_name):
    """ gets the content of a book details page,
    and parses different components and stores the info """
    book_url, book_name = book_info
    book_url = book_url.replace("../..../", "")
    book_url = "http://books.toscrape.com/catalogue/" +
    book_url

    content = get_page_content(book_url)

```

```
content = content.replace("\n", " ")

upc, price, image_url, availability, description =
get_product_details(content)
print(upc, price, image_url)
```

এখন আমরা চাইলে পুরো প্রোগ্রামটি রান করতে পারি। সব ঠিকঠাক থাকলে নিচের মতো আউটপুট আসা শুরু করবে।

```
$ python book_crawler_final.py
a22124811bfa8350 45.17
http://books.toscrape.com/media/cache/6d/41/6d418a73cc7d4ec
fd75ca11d854041db.jpg
ce60436f52c5ee68 49.43
http://books.toscrape.com/media/cache/fe/8a/fe8af6ceec77189
86380c0fde9b3b34f.jpg
f9705c362f070608 48.87
http://books.toscrape.com/media/cache/c7/1a/c71a85dbf8c2dbc
75cb271026618477c.jpg
1809259a5a5f1d8d 36.94
http://books.toscrape.com/media/cache/ca/30/ca30b1afe1e76ce
7ba1db8176d398e53.jpg
a94350ee74deaa07 37.33
http://books.toscrape.com/media/cache/45/21/4521c581ba727f5
c835e34860cbf53e5.jpg
cc1936a9f4e93477 44.34
http://books.toscrape.com/media/cache/6c/e3/6ce3003931701c7
a3fd5354917538ea9.jpg
```

আর কোনো সমস্যা হলেও অসুবিধা নেই। কিছুক্ষণের মধ্যেই আমি পুরো কোড লিখে দেব। এখন আমার বাকি কাজ হচ্ছে সিএসভি ফাইলে সব তথ্য লেখা। এজন্য প্রোগ্রামের শুরুতেই আমরা লিখেছি :

```
with open("book_list.csv", "w") as csvf:
    csv_writer = csv.DictWriter(csvf, fieldnames=["Name",
    "Category", "UPC", "URL", "ImageURL",
    "Price", "Availability", "Description"])
    csv_writer.writeheader()
```

অধ্যায় ৮ - ওয়েব ক্রলিং (Web Crawling)

এখানে আমি বলে দিচ্ছি যে, সিএসভি ফাইলের নাম হচ্ছে book_list.csv। আর সেই সঙ্গে csv মডিউলের DictWriter ক্লাস ব্যবহার করছি। এটি ব্যবহার করার সুবিধা হচ্ছে, writerow() ফাংশনে আমি লিস্টের পরিবর্তে ডিকশনারি ব্যবহার করতে পারব। লিস্টের বদলে আমি কেন ডিকশনারি ব্যবহার করতে চাই? কারণ একটি বইয়ের যেহেতু নানান রকম তথ্য আছে, লিস্ট ব্যবহার করলে সবগুলোর একটি ক্রম নির্ধারণ করে দিয়ে সেটি মেনে চলতে হয়। আর ডিকশনারির ক্ষেত্রে আমাদের কাজটি অনেক সহজ হয়ে যায়, কোনটি আগে লিখলাম, কোনটি পরে, সেটি নিয়ে মাথা ঘামাতে হয় না। গত দু-তিনটি বাকে আমি কী লিখলাম, সেটি অনেকেই বুঝতে পারবে না, এবং তাতে কোনো অসুবিধা নেই। ওয়েব ক্রলার নিয়ে আমাদের অভিজ্ঞতা একটু বাড়লেই আমরা বুঝতে পারব।

এখানে আরো একটা ব্যাপার বলে রাখা ভালো, উইন্ডোজ অপারেটিং সিস্টেমে এই কোডটি রান করলে রানটাইমে UnicodeEncodeError এক্সেপশন দেখাতে পারে। সে ক্ষেত্রে ফাইলটি ওপেন করার সময় আমাদের কনটেন্টের এনকোডিং কী হবে সেটি বলে দিতে হবে। কনটেন্ট এনকোডিং "utf-8" দিয়ে কাজ চালানো যায়, তবে আরো ভালো বুদ্ধি হচ্ছে রেসপন্সের এনকোডিং ব্যবহার করা।

```
>>> url = "http://books.toscrape.com/index.html"
>>> response = requests.get(url)
>>> response.encoding
'ISO-8859-1'
```

সূতরাং সিএসভি ফাইল ওপেন করার সময় আমরা কনটেন্ট এনকোডিং লিখে দেব 'ISO-8859-1'। খেয়াল করতে হবে যে, বইয়ের নাম টার্মিনালে প্রিন্ট করার সময় যদি ঠিকমতো না-ও দেখায় সিএসভি ফাইলে তা ঠিকমতোই সেত হবে।

```
with open("book_list.csv", "w", encoding="ISO-8859-1") as csvf:
    csv_writer = csv.DictWriter(csvf, fieldnames=["Name",
    "Category", "UPC", "URL", "ImageURL",
    "Price", "Availability", "Description"])
    csv_writer.writeheader()
```

scrape_book_info() ফাংশনটি এখন দেখতে এরকম হবে :

```
def scrape_book_info(book_info, category_name):
    """ gets the content of a book details page,
    and parses different components and stores the info """
```

```

book_url, book_name = book_info
book_dict = {"Name": book_name, "Category":
category_name}

book_url = book_url.replace("../..../", "")
book_url = "http://books.toscrape.com/catalogue/" +
book_url
book_dict["URL"] = book_url

print("Scraping book", book_name)
logging.info("Scraping : " + book_url)

content = get_page_content(book_url)
content = content.replace("\n", " ")

upc, price, image_url, availability, desc =
get_product_details(content)
book_dict["UPC"] = upc
book_dict["Price"] = price
book_dict["ImageURL"] = image_url
book_dict["Availability"] = availability
book_dict["Description"] = desc

csv_writer.writerow(book_dict)

```

এখনে book_dict হচ্ছে একটি ডিকশনারি এবং আমি যখন যে তথ্য পাচ্ছি, তখনই তা ডিকশনারিতে যোগ করছি। আমার পুরো প্রোগ্রাম তৈরি হয়ে গেল। আমি নিচে কোড দিচ্ছি। এ ছাড়া গিটহাবেও আমি প্রোগ্রামটি দিয়ে রেখেছি (এই লিংকে : <https://github.com/DimikOrg/Python-Book-Website-Crawler>)। তবে নিচের কোড দেখার আগে নিজে নিজে প্রোগ্রামটি রান করার চেষ্টা করলে খুব ভালো হয়। প্রোগ্রাম রান করতে গিয়ে বিভিন্ন রকম ঝামেলা হবে, বিভিন্ন এরর আসবে, সেগুলো নিজে নিজে সমাধান করার চেষ্টা করাটাই কিন্তু প্রোগ্রামিং শেখার মূল বিষয়। এই নিজের করার কাজটাই কিন্তু একজন প্রোগ্রামারকে অন্যদের চেয়ে অনেক এগিয়ে নিয়ে যায়। আশা করি, বেশিরভাগ পাঠকই নিজে প্রোগ্রামটি রান করার জন্য কয়েক ঘণ্টা শ্রম দেবে।

```

import logging
import sys
import requests
import re

```

```

import csv
from html import unescape

def get_category_list(content):
    """get_category_list takes content of home page
    and returns a list of categories and their urls"""
    return category_pat.findall(content)

def get_book_list(content):
    """get_book_list takes content of a book list page
    and returns a list of books (name and url)"""
    content = content.replace("\n", " ")
    return book_list_pat.findall(content)

def get_product_details(content):
    """get_product_details takes content of a product page,
    parses the page and returns details about a product"""
    image_base = "http://books.toscrape.com/"
    result = img_pat.findall(content)
    if len(result) == 0:
        logging.warn("Image url not found!")
        image_url = ""
    else:
        img_url = result[0]
        img_url = img_url.replace("../..", "")
        image_url = image_base + img_url

    result = desc_pat.findall(content)
    if len(result) == 0:
        logging.warn("Description not found!")
        description = ""
    else:
        description = unescape(result[0])

    result = upc_pat.findall(content)
    if len(result) == 0:
        logging.warn("UPC not found!")
        upc = ""
    else:
        upc = result[0]

```

```
result = price_pat.findall(content)
if len(result) == 0:
    logging.warn("Price not found!")
    price = ""
else:
    price = result[0]

result = avail_pat.findall(content)
if len(result) == 0:
    logging.warn("Availability not found!")
    availability = ""
else:
    availability = result[0]

return upc, price, image_url, availability, description

def get_page_content(url):
    """get_page_content takes a url and returns the content
    of the page"""
    try:
        response = requests.get(url)
    except requests.exceptions.RequestException as e:
        logging.error(e)

    if response.ok:
        return response.text

    logging.error("Can not get content from URL:" + url)
    return ""

def get_next_page(url, content):
    """Checks the content of a book list page and returns
    link of the next page,
    returns None, if there is no more next page"""
    result = next_page_pat.findall(content)
    if len(result) == 0:
        return None
    i = url.rfind("/")
    return url[0:i+1] + result[0]
```

```
def scrape_book_info(book_info, category_name):
    """ gets the content of a book details page,
    and parses different components and stores the info """
    book_url, book_name = book_info
    book_name = unescape(book_name)
    book_dict = {"Name": book_name, "Category":
category_name}

    book_url = book_url.replace("../..../", "")
    book_url = "http://books.toscrape.com/catalogue/" +
book_url
    book_dict["URL"] = book_url

    print("Scraping book", book_name)
    logging.info("Scraping : " + book_url)

    content = get_page_content(book_url)
    content = content.replace("\n", " ")

    upc, price, image_url, availability, desc =
get_product_details(content)
    book_dict["UPC"] = upc
    book_dict["Price"] = price
    book_dict["ImageURL"] = image_url
    book_dict["Availability"] = availability
    book_dict["Description"] = desc

    csv_writer.writerow(book_dict)

def crawl_category(category_name, category_url):
    """crawls a particular category of books"""
    while True:
        content = get_page_content(category_url)

        book_list = get_book_list(content)

        for book in book_list:
            scrape_book_info(book, category_name)
```

```

next_page = get_next_page(category_url, content)
if next_page is None:
    break

category_url = next_page

def crawl_website():
    """crawl_website() is the main function that
    coordinates the whole crawling task"""
    url = "http://books.toscrape.com/index.html"
    host_name = "books.toscrape.com"

    content = get_page_content(url)
    if content == "" :
        logging.critical("Got empty content from " + url)
        sys.exit(1)

    category_list = get_category_list(content)

    for category in category_list:
        category_url, category_name = category
        category_url = "http://" + host_name + "/" + \
            category_url
        crawl_category(category_name, category_url)

if __name__ == "__main__":
    # Compile different regular expression patterns
    category_pat = re.compile(r'<li>\s*<a href="(catalogue/category/books/.*?)">\s*(\w+[\s\w]+\w)\s*?<', re.M | re.DOTALL)
    next_page_pat = re.compile(r'<li class="next"><a href="(.*)">next</a></li>')
    book_list_pat = re.compile(r'<h3><a href="(.*)" title="(.*)">' )
    img_pat = re.compile(r'<div class="item active">\s*.?\<p>(.?)</p>' )

```

```

upc_pat = re.compile(r'<th>UPC</th>\s*<td>(.*?)</td>')
price_pat = re.compile(r'<th>Price \\\(incl.
tax\)</th>\s*<td>\D+([\d.]*)</td>')
avail_pat =
re.compile(r'<th>Availability</th>\s*<td>(.*?)</td>')

logging.basicConfig(format='%(asctime)s %(message)s',
datefmt='%m/%d/%Y %I:%M:%S %p',
filename="bookstore_crawler.log", level=logging.DEBUG)

with open("book_list.csv", "w", encoding="ISO-8859-1") as csvf:
    csv_writer = csv.DictWriter(csvf,
fieldnames=["Name", "Category", "UPC", "URL", "ImageURL",
"Price", "Availability", "Description"])
    csv_writer.writeheader()

    crawl_website()
    print("Crawling Done!")

```

ওপরের কোডে আমি বাড়তি দুটো কাজ করেছি— book_name ও desc-এর মানের ওপরে এইচটিএমএল আনএসকেপিং করা। যদি বইয়ের নাম বা বিবরণে কোনো এইচটিএমএল কোড থাকে যেমন > তাহলে, তাকে > চিহ্ন দিয়ে প্রতিস্থাপন করাটাই হচ্ছে আনএসকেপিং। পাইথনের html মডিউলে unescape নামে একটি ফাংশন রয়েছে এই কাজ করার জন্য।

```

>>> from html import unescape
>>> print(unescape('It's Only the Himalayas'))
It's Only the Himalayas

```

প্রোগ্রামটি এখন ঠিকভাবে রান করবে এবং কিছু সময়ের মধ্যেই সব বইয়ের তথ্য book_list.csv ফাইলে সেভ করে ফেলবে। প্রোগ্রাম রান শেষ হলে আমরা ওই ফাইলটি খুলতে পারি, এবং সেখানে দেখব যে, মোট এক হাজারটি বইয়ের তথ্য রয়েছে।

অধ্যায় ৯ - প্রোগ্রামিংয়ের আনন্দযাত্রা

প্রোগ্রামিং শেখা মানে কেবল দু-একটি বই কিংবা ব্লগ আর্টিকেল পড়া নয়। প্রোগ্রামিং হচ্ছে একটি শিল্প, যার জন্য নিজেকে প্রস্তুত করতে হবে, সময় নিয়ে অনুশীলন করতে হবে। আবার এটি ঠিক অন্য বই পড়ার মতো নয় যে, সব পড়ে ফেললাম, শেখা হয়ে গেল। আমি তাই বইটি লেখার সময় সব তথ্য হাজির করতে চাইনি, বরং আমি যদি কাউকে প্রোগ্রামিং শেখাতাম, তাহলে যেভাবে শেখানোর চেষ্টা করতাম, বইটিতে সেভাবেই সবকিছু তুলে ধরেছি। বইটি যারা দু-তিনবার পড়বে এবং প্রতিবারই প্রোগ্রামগুলো নিজে লিখে দেখবে, তাদের কাছে প্রোগ্রামিং অনেক সহজ ও সুন্দর রূপে ধরা দেবে।

পাইথন দিয়ে প্রোগ্রামিং শেখা বইতে আমরা পাইথন দিয়ে প্রোগ্রামিংয়ের মূল ধারণাগুলোর সঙ্গে পরিচিত হয়েছিলাম। আর এই বইতে পাইথনের বেশ কিছু নতুন জিনিস শিখলাম। সেই সঙ্গে আমরা একটু বড় প্রোগ্রাম তৈরি করাও শিখে ফেললাম। যেহেতু আমরা ওয়েব ক্রলার তৈরি করা শিখেছি, আমাদের এখন চেষ্টা করতে হবে, আরো কয়েকটি ওয়েবসাইট ক্রল করার জন্য। আর সেটি করতে গিয়ে আমাদের দক্ষতা বৃদ্ধি পাবে। ওয়েব ক্রলিংয়ের জন্য নানান রকম ফ্রেমওয়ার্ক রয়েছে (যেমন, scrapy) কিন্তু আমরা সেসব এখন শিখব না। আগে নিজেদের দক্ষতা বৃদ্ধি করার চেষ্টা করব, কারণ দীর্ঘ মেয়াদে এটি সুফল বয়ে আনবে। কোনো বন্ধু হয়তো বলবে, দেখো, scrapy দিয়ে কী সহজে ক্রলিং করে ফেলছি, কিংবা রেণুলার এক্সপ্রেশনের বদলে BeautifulSoup ব্যবহার করে কাজ চলে যাচ্ছে! এসব কথায় প্রভাবিত হওয়া চলবে না। আমরা যখন নিজেদের দক্ষতা আন্তর্জাতিক পর্যায়ে নিয়ে যেতে পারব, তখন প্রয়োজনমতো বিভিন্ন টুলস (Scrapy, BeautifulSoup ইত্যাদি) ব্যবহার করব।

এরপর আমরা কী শিখব?

কারো যদি ডেক্সটপ-ভিত্তিক সফটওয়্যার তৈরির প্রয়োজন হয়, তাহলে গুই (GUI → Graphical User Interface) তৈরি করা শিখতে পারে। তবে আজকাল কিন্তু এ ধরনের কাজের প্রয়োজন খুব কমই হয়, তাই আমি গুই শেখার পেছনে সময় দেওয়াটাকে উৎসাহিত করব না।

ওয়েব প্রোগ্রামিং আমাদের সবারই কম-বেশি শিখতে হবে। তবে শুরুতেই ফ্রেমওয়ার্ক ব্যবহার করা ঠিক নয়। বরং ওয়েব কীভাবে কাজ করে, এটি জানা জরুরি। ফ্রেমওয়ার্ক ব্যবহার না করে একটি সাধারণ ওয়েব অ্যাপ্লিকেশন তৈরি করা শিখতে হবে। তারপর আমরা একটি সহজ ওয়েব

অধ্যায় ৯ - প্রোগ্রামিংয়ের আনন্দযাত্রা

ফ্রেমওয়ার্ক ব্যবহার করা শিখতে পারি, যেমন Flask। ওয়েবে যথেষ্ট অভিজ্ঞতা অর্জন করার পরে Django ফ্রেমওয়ার্কটি শিখতে হবে, কারণ এটি পাইথনের সবচেয়ে জনপ্রিয় ও বহুল ব্যবহৃত ওয়েব ফ্রেমওয়ার্ক।

এই বইটি পড়ার পরে আরো ভালোভাবে আমরা পাইথন শেখার চেষ্টা করতে পারি। সেজন্য আমি পাইথনের অফিশিয়াল টিউটোরিয়াল পড়ার পরামর্শ দেব। সেটি পাওয়া যাবে এখানে : <https://docs.python.org/3/tutorial/index.html>।

যারা ডেটা অ্যানালাইসিস, মেশিন লার্নিং ইত্যাদি কাজে পাইথন ব্যবহার করবে, তারা তাদের প্রয়োজনমতো লাইব্রেরিগুলোর ব্যবহার শিখে নেবে।

আবার গেম প্রোগ্রামিংয়ে উৎসাহীদের আমি বলব পাইগেম (<https://www.pygame.org>) শেখার জন্য। ওয়েবসাইটে গিয়ে টিউটোরিয়াল ঘাঁটলেই শেখা যাবে।

আমরা দেখতে পাচ্ছি, প্রোগ্রামিংয়ের জগৎ অনেক বিশাল। অনেক কিছু শেখার আছে এখানে। তাই অস্ত্রিল হয়ে সবকিছু শেখার চেষ্টা করাটা সঠিক হবে বলে আমার মনে হয় না। বরং কারো বেসিক যদি শক্ত থাকে, তাহলে সে যখন যা প্রয়োজন হয়, তা শিখে নিতে পারবে। আশা করি, পাঠক প্রোগ্রামিংয়ের মাঝে আনন্দ খুঁজে পাবে। প্রোগ্রামিংয়ের এই আনন্দযাত্রায় সবার জন্য শুভকামনা রইল।