# ICTPRG434 Task 5: Image Downloader

### *Background*

This task focuses on web scraping, an essential skill for extracting information from websites. You will create a Python program to download all images from a specified webpage and save them to a folder. This project will introduce you to working with web content in Python, using either regular expressions or Beautiful Soup, a popular web scraping library.

Please ensure you have read the following in **Automate the Boring Stuff with Python**, or are confident with the concepts:

- **Chapter 7: Pattern Matching with Regular Expressions**
- **Chapter 9: Reading and Writing Files**
- **Chapter 12: Web Scraping**

We will be using either regular expressions or Beautiful Soup for this exercise. See the following resources for more information:

- **Beautiful Soup documentation**
- **Beautiful Soup: Build a Web Scraper with Python (realpython.com)**
- **Python Regular Expression HOWTO**
- **Regular Expressions 101 (regex tester)**

### *Objectives*

- Understand the basics of web scraping and how to apply it to extract images from websites.
- Learn to use Python libraries such as Beautiful Soup or regular expressions for parsing HTML content.
- Develop skills in handling HTTP requests for downloading web content.
- Practice managing file operations in Python, including creating directories and saving files.
- Enhance understanding of error handling and user interface design within Python applications.

### *Core Functionalities*

- **Extract image URLs:** Parse a webpage to find and extract all image URLs.
- **Download images:** Download each image from its URL and save it to a specified directory.
- **Directory creation:** Automatically create the directory for storing images if it doesn't exist.
- **Error handling:** Manage invalid inputs, failed downloads, and parsing errors.
- **User interface:** Provide a simple interface for inputting webpage URL and feedback on the download process.


### *Functions to Implement*

1. `download_images(url: str, folder_name: str)`: This function is responsible for fetching all images from the specified webpage URL, creating a directory to store them, and invoking another function to download each image individually.
2. `download_image(img_url: str, folder_name: str)`: This function handles the downloading of a single image from its URL and saving it into the designated folder, managing HTTP requests and file writing for each image.

**Initial Setup**:

- Start by setting up a new Python script and defining the main function.
- Install and import necessary libraries (`requests` for fetching webpage content, Beautiful Soup or `re` for parsing, and `os` for file operations).

**Implementing Web Scraping**:

- Prompt the user to enter the URL of the webpage they wish to scrape.
- Fetch the webpage content using the `requests` library.
- Parse the webpage to find image URLs. You can use either regular expressions to search for image URL patterns or Beautiful Soup to parse the HTML and extract image tags.
- Handle relative and absolute URLs appropriately. i.e., if a link doesn't start with a `http://` or `https://`, you must join the base URL with the image URL.

**Downloading and Saving Images**:

- For each image URL found, download the image using `requests`.
- Save the images in a specified directory on your local machine. Ensure to handle naming conflicts and potential download errors.

**User Interface and Error Handling**:

- Implement a user-friendly interface and provide clear instructions.
- Include error handling to manage invalid inputs, failed downloads, or other potential issues.

**Testing Your Program**:

- Test your program with different webpages to ensure it can handle various structures and types of content.
- Ensure that the program saves the images correctly and handles errors gracefully.

### *Extra Tips*

- Ensure that your web scraping activities comply with the website's terms of service and legal considerations.
- Consider the variety of image formats and potential issues with downloading large files.

### *Deliverables*

Submit your Python script (.py file) with clear comments, along with a brief report explaining your approach, the challenges faced during web scraping and file handling, and how you overcame these challenges.