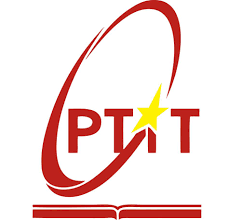
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

**KHOA CÔNG NGHỆ THÔNG TIN 1**



**BÁO CÁO BÀI TẬP LỚN PYTHON**

**ỨNG DỤNG PYTHON SCRAPING**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Nhóm học phần:*** | ***Nhóm 5*** | | |
| ***Nhóm bài tập lớn:*** | ***Nhóm 16*** | | |
| ***Sinh viên thực hiện:*** |  |  | |
|  | ***Trần Quang Huy :***  ***(Trưởng nhóm)*** | | ***B20DCCN327*** |
|  | ***Nguyễn Cảnh Hưng:*** | | ***B20DCCN339*** |
|  | ***Nguyễn Tiến Anh:*** | | ***B20DCCN005*** |
|  | ***Nguyễn Hồng Hiếu:*** | | ***B20DCCN252*** |
| ***Giảng viên giảng dạy:*** | ***Thầy Nguyễn Hoàng Anh*** | | |

**HÀ NỘI-2022**

**Mục lục**

[**I.** **Giới thiệu sơ lược về sản phẩm của nhóm** 1](#_Toc119166318)

[**1.** **Sơ đồ khối chức năng ứng dụng** 1](#_Toc119166319)

[**2.** **Danh sách các lớp đã xây dựng** 2](#_Toc119166320)

[**3.** **Sơ đồ thực thể quan hệ ERD** 4](#_Toc119166321)

[**II.** **So sánh nội dung với github được giao** 4](#_Toc119166322)

[**III.** **Kết quả ứng dụng** 4](#_Toc119166323)

[**1.** **Giải thích code quan trọng cho từng phần** 4](#_Toc119166324)

[**a)** **File chứa lớp product: Product.py** 4](#_Toc119166325)

[**b)** **File khởi tạo GUI: init.py** 5](#_Toc119166326)

[**c)** **Module shopeeFunctions.py dùng để xây dựng các hàm lấy về dữ liệu từ shopee** 8](#_Toc119166327)

[**d)** **Module imageFunctions.py dùng để xây dựng các hàm download ảnh từ 1 trang web** 18](#_Toc119166328)

[**e)** **File chứa hàm để chạy: main.py** 21](#_Toc119166329)

[**2.** **Cài đặt triển khai ứng dụng** 21](#_Toc119166330)

[**3.** **Kết quả thực nghiệm** 22](#_Toc119166331)

[**4.** **Kết luận, hạn chế và phương hướng phát triển trong tương lai** 26](#_Toc119166332)

[**5.** **Tài liệu tham khảo** 26](#_Toc119166333)

[**IV.** **ĐÓNG GÓP CỦA CÁC THÀNH VIÊN** 27](#_Toc119166334)

**Danh sách hình vẽ và hình ảnh**

Hình 1. Sơ đồ khối chức năng của ứng dụng…………………………………………...2

Hình 2. Quan hệ giữa các lớp được xây dựng…………………………………………..3

Hình 3. Các module functions được xây dựng…………………………………………. 3

Hình 4. Sơ đồ thực thể quan hệ ER…………………………………………………….. 4

Hình 5. Xây dựng lớp product…………………………………………………………. 5

Hình 6. Các thư viện sử dụng trong init.py…………………………………………….. 5

Hình 7. Xây dựng lớp Window………………………………………………………… 6

Hình 8. GUI của tab thứ nhất……………………………………………………………7

Hình 9. GUI của tab thứ 2………………………………………………………………7

Hình 10. Các thư viện dùng trong shopeeFunctions.py………………………………...8

Hình 11. Đường dẫn tới Chromedrive………………………………………………….8

Hình 12. Hàm xác định vị trí frame trên màn hình……………………………………..9

Hình 13. Ví dụ về link bản phím cơ…………………………………………………….9

Hình 14. Ví dụ về link dầu gội đầu……………………………………………………..9

Hình 15. Hàm tạo link dẫn tới trang shopee……………………………………………10

Hình 16. Hàm trả về source code web dạng BeautifulSoup……………………………11

Hình 17. Minh họa cho source code trang web………………………………………...12

Hình 18. Code cho phần phân tích bằng BeautifulSoup………………………………..13

Hình 19. Minh họa cho code đánh giá sản phẩm……………………………………….14

Hình 20. Hàm run để lấy từ nhiều trang………………………………………………..14

Hình 21. Các hàm cho TreeView……………………………………………………….15

Hình 22. Các hàm sort productList……………………………………………………..15

Hình 23. Hàm để lưu vào file CSV……………………………………………………..16

Hình 24. Hàm kết nối tới shopee……………………………………………………….16

Hình 25. Các thư viện sử dụng trong imageFunctions.py……………………………...17

Hình 26. Try except đường dẫn nhập vào………………………………………………18

Hình 27. Sử dụng selenium để lấy source code………………………………………..18

Hình 28. Lấy về danh sách đường link ảnh……………………………………………..19

Hình 29. Lấy ảnh về bằng requests …………………………………………………….19

Hình 30. Download ảnh về folder………………………………………………………19

Hình 31. Tiến hành download nhiều file cùng lúc bằng thread………………………...20

Hình 32. File main………………………………………………………………………20

Hình 33. Demo lấy dữ liệu từ shopee…………………………………………………..21

Hình 34. Demo xử lý dữ liệu từ shopee………………………………………………..22

Hình 35. Demo hiển thị dữ liệu trên màn hình ………………………………………..22

Hình 36. Demo tải ảnh từ 1 website……………………………………………………23

Hình 37. Demo load trang và xử lý dữ liệu trong quá trình tải ảnh……………………24

Hình 38. Demo kết quả sau khi tải ảnh…………………………………………………24

1. **Giới thiệu sơ lược về sản phẩm của nhóm**

**Mô tả ứng dụng**

* Tên ứng dụng: Python scraping
* Nhận thấy được người dùng đang dành nhiều thời gian cho việc mua sắm online, (Cụ thể là sàn TMĐT Shopee). Quyết định mua sản phẩm của họ thường dựa trên một số yếu tố nhất định: Shop bán đồ đó có uy tín hay không, số lượng sản phẩm đã bán ra có nhiều hay không và tỷ lệ yêu thích sản phẩm đó như thế nào? … Nhận thấy được điều đó, nhóm 16 có triển khai thực hiện 1 ứng dụng giúp việc mua sắm trở nên dễ dàng hơn. Bằng việc lấy toàn bộ dữ liệu cơ bản về sản phẩm từ ngay trang 1 của shopee.vn. Nhóm 16 đã lọc ra những thông tin cần thiết đủ để khách hàng có thể đưa ra quyết định nên chọn sản phẩm nào.
* Thêm vào đó, nhóm cũng tạo tính năng giúp người dùng có thể lấy số lượng ảnh theo yêu cầu các link từ 1 trang web cho trước.
* Phân tích thiết kế chi tiết ứng dụng:

1. **Sơ đồ khối chức năng ứng dụng**

Python scraping

Get danh sách sản phẩm từ shopee

Get các ảnh từ 1 trang web bất kì

Lưu và hiển thị các ảnh đã lưu trong thư mục

Sắp xếp danh sách sản phẩm

Lưu danh sách sản phẩm vào file CSV

Hiển thị danh sách sản phẩm lên màn hình

Sắp xếp theo đánh giá sản phẩm

Sắp xếp theo giá giảm dần

Sắp xếp theo doanh số giảm dần

Hình 1. Sơ đồ khối chức năng của ứng dụng

1. **Danh sách các lớp đã xây dựng**

* Lớp **product** để mô tả các sản phẩm lấy từ shopee.vn

Bao gồm: Tên, giá nhỏ nhất, giá lớn nhất, đánh giá sản phẩm, tổng số lượt bán ra của shop đó và cuối cùng là link dẫn đến sản phẩm đó trên shopee

* Lớp Window kế thừa từ tkinter.Tk(): Để khởi tạo GUI

Biểu đồ:

fdProduct

* Name: str
* Min price: int
* Max price: int
* Rating: float
* Sales: int
* Link: str
* \_\_str\_\_()
* \_\_iter\_\_()

Window(tk.Tk)

* Tk: tkinter.Tk()
* productList: list
* \_\_init\_\_()
* Intro()
* initPage()
* run()

Hình 2. Quan hệ giữa các lớp được xây dựng

* Ngoài ra, nhóm 16 còn tạo ra 2 file py chứa các hàm được sử dụng trong chương trình.
  + File1: shopeeFunctions.py: chứa các hàm liên quan đến Chức năng thứ nhất: Lấy danh sách sản phẩm từ shopee theo tên tìm kiếm
  + File2: ImageFunctions.py chứa các hàm liên quan đến Chức năng thứ hai: Lấy các ảnh từ 1 website rồi lưu vào thư mục đã được định sẵn

shopeeFunctions.py

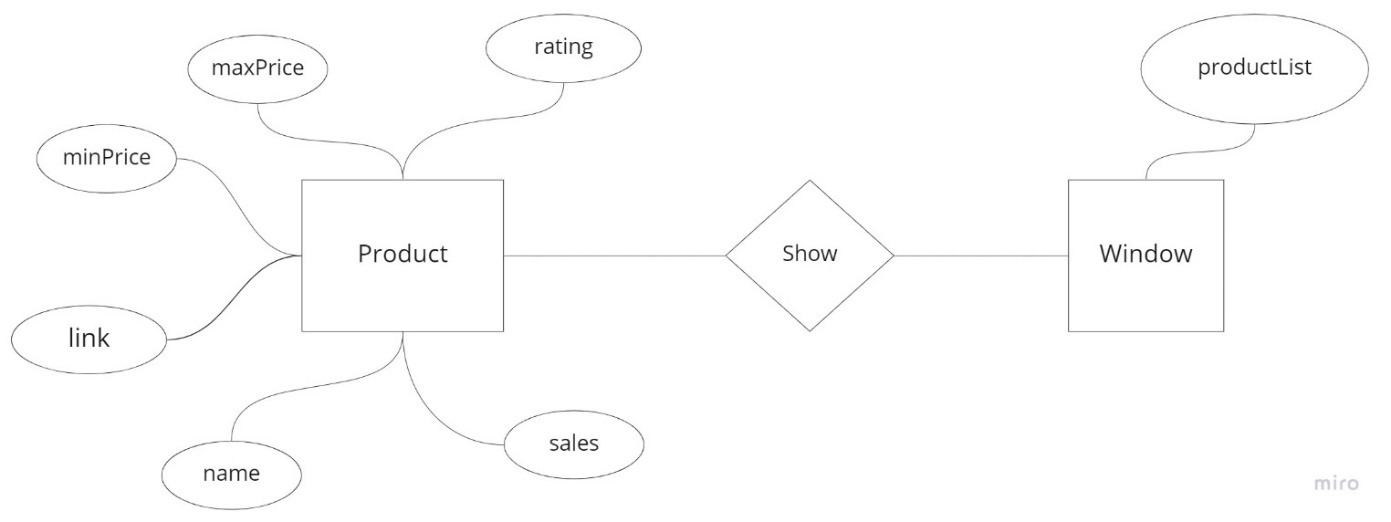
* accessToGithub()
* getPosition()
* generateLink()
* getHtml()
* showProgressBar()
* endProgressBar()
* fillProductList()
* run()
* writeToFile()
* clearTreeView()
* fillTreeView()
* showProducts()
* accessToShopee()

imageFunctions.py

* get\_url()
* showProgressBar()
* endProgressBar()
* download()
* getImage\_run()
* showFolder()

Hình 3. Các module functions được xây dựng

1. **Sơ đồ thực thể quan hệ ERD**



Hình 4. Sơ đồ thực thể quan hệ ERD

1. **So sánh nội dung với github được giao**

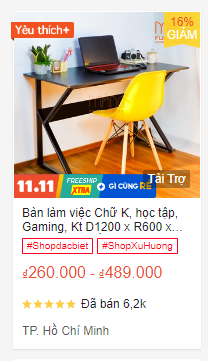
* Sau khi tìm hiểu về git: <https://github.com/REMitchell/python-scraping.git>. Team 16 nhận thấy danh sách source code được push lên là code hướng dẫn của cuốn sách “web scraping with Python” của tác giả Ryan Mitchell. Sách gồm 18 chương, mỗi chương là 1 phần hướng dẫn về cách crawl dữ liệu từ web. Đi cùng với đó là code sample demo cho từng phần.
* Vì lí do đó team 16 quyết định sẽ phát triển mới 1 ứng dụng để crawl dữ liệu có vận dụng kiến thức đã được đề cập trong sách.

1. **Kết quả ứng dụng**
   1. **Giải thích code quan trọng cho từng phần**
   * **Các thư viện mà tổng thể chương trình đã sử dụng**

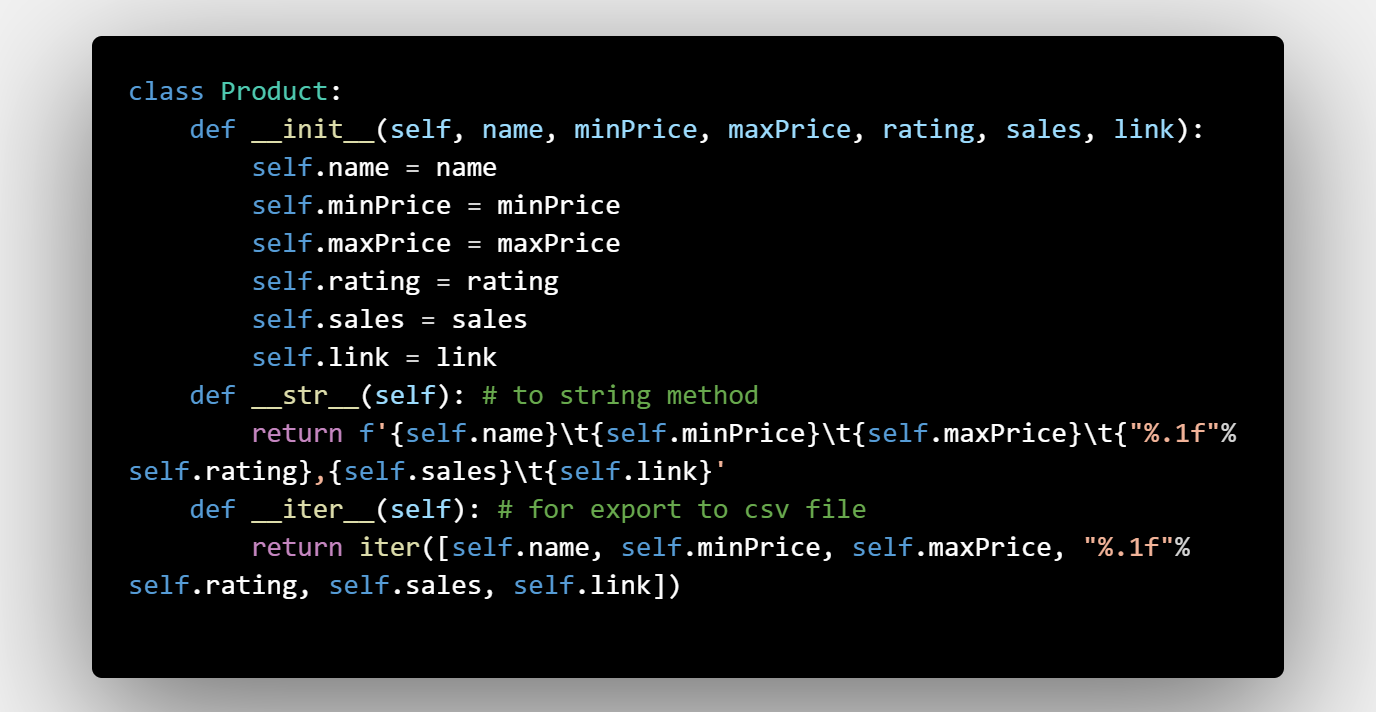
Tkinter, BeautifulSoup, selenium, webbrowser, futures, urllib.request, urllib.error, time, requests, os, tqdm, PIL, csv

Trong đó, thư viện phục vụ chính cho việc lấy dữ liệu là **BeautifulSoup** và **selenium**

1. **File chứa lớp product: Product.py**

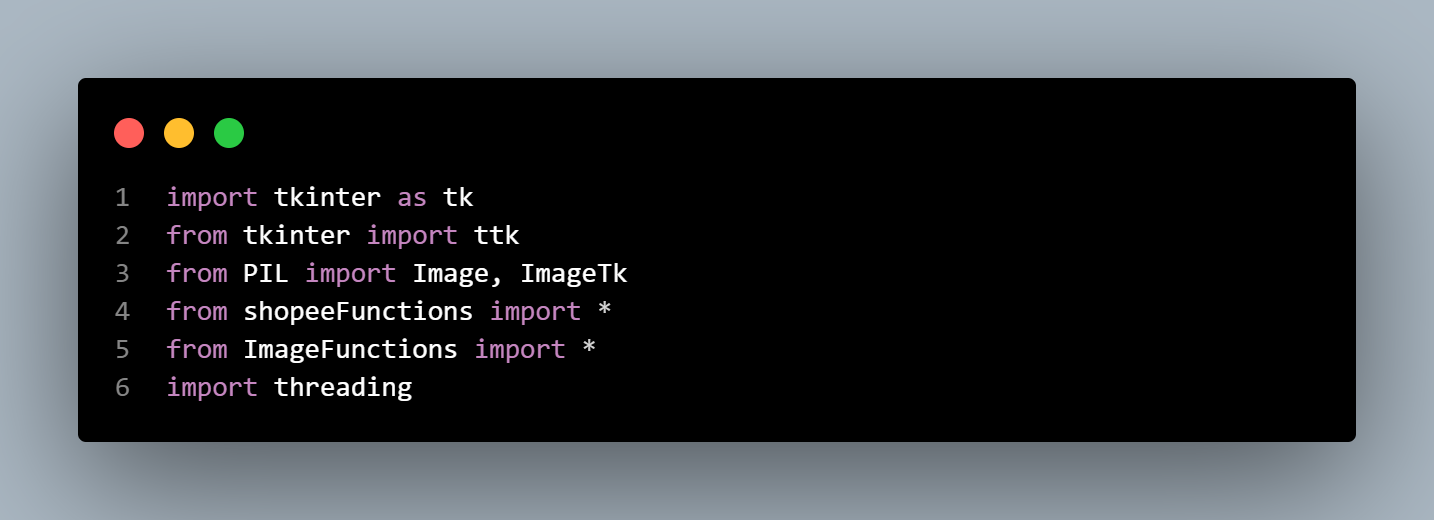
Từ phân tích sơ lược trang web shopee.vn, nhóm 16 đã rút ra được 5 thuộc tính cơ bản mà người dùng có thể quan tâm khi mua sắm trên web.

Từ 1 sản phẩm ngẫu nhiên: ta có thể thấy được các thông tin cơ bản như tên sản phẩm, giá nhỏ nhất, giá lớn nhất, lượt đánh giá (sao), doanh số sản phẩm và 1 thuộc tính ẩn là đường link dẫn đến sản phẩm đó  
- Từ đó, nhóm xây dựng lớp product như sau

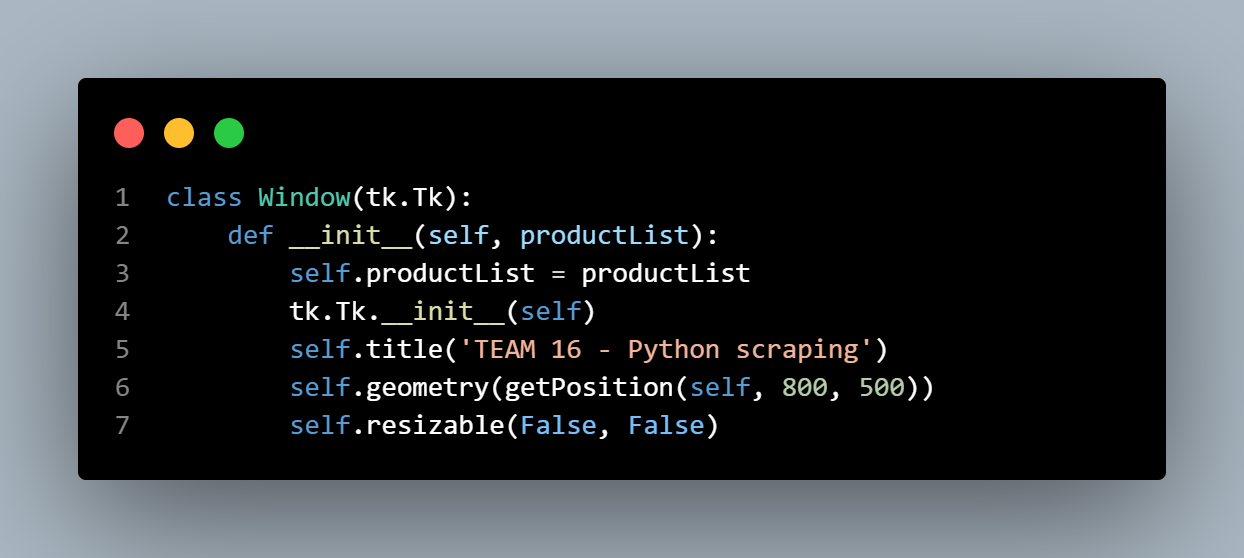
Hình 5. Xây dựng lớp product

* Với 2 phương thức: \_\_str\_\_() dùng cho việc ghi sản phẩm ra màn hình console, \_\_iter\_\_() dùng cho việc ghi sản phẩm vào file csv.

1. **File khởi tạo GUI: init.py**

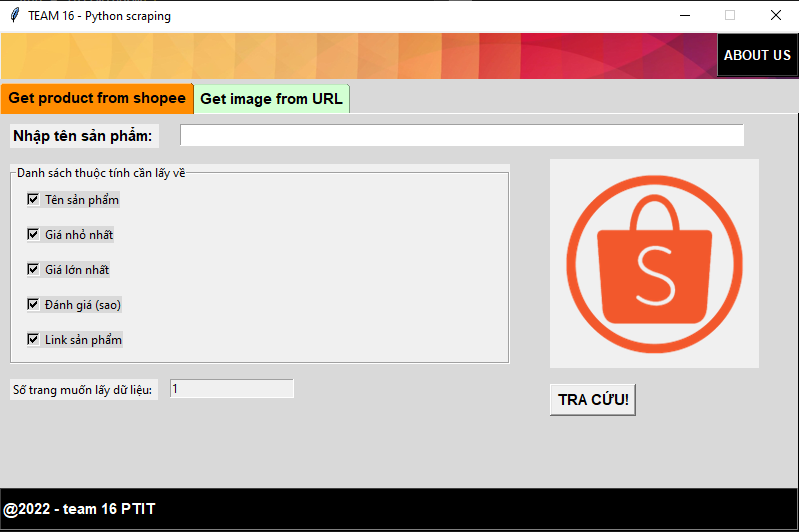
* Các thư viện đã sử dụng: tkinter bao gồm tk và ttk, PIL, threading
* Các module tự xây dựng: shopeeFunctions và imageFunctions

Hình 6. Các thư viện sử dụng trong init.py

* Nhóm đã xây dựng lớp Window kế thừa từ tk.Tk() với hàm khởi tạo chỉnh các yếu tố về vị trí và kích thước cho lớp:

Hình 7. Xây dựng lớp Window

* Trong lớp Window, nhóm đã xây dựng hàm initPage() với mục đích thêm các widget con vào trong Tk() frame ban đầu.
  + Label: background, hiện thị các yêu cầu
  + Button: các button dùng để tra cứu và đến github của nhóm
  + Tab widget: chia thành 2 tab trên màn hình
  + Style: xây dựng style dùng trong app
  + Checkbox: các checkbox chọn dữ liệu

Sau khi hoàn thiện, GUI của tab 1 sẽ như sau

Hình 8. GUI của tab thứ nhất

* Tab thứ 2 được xây dựng đơn giản hơn với 2 label và 1 entry

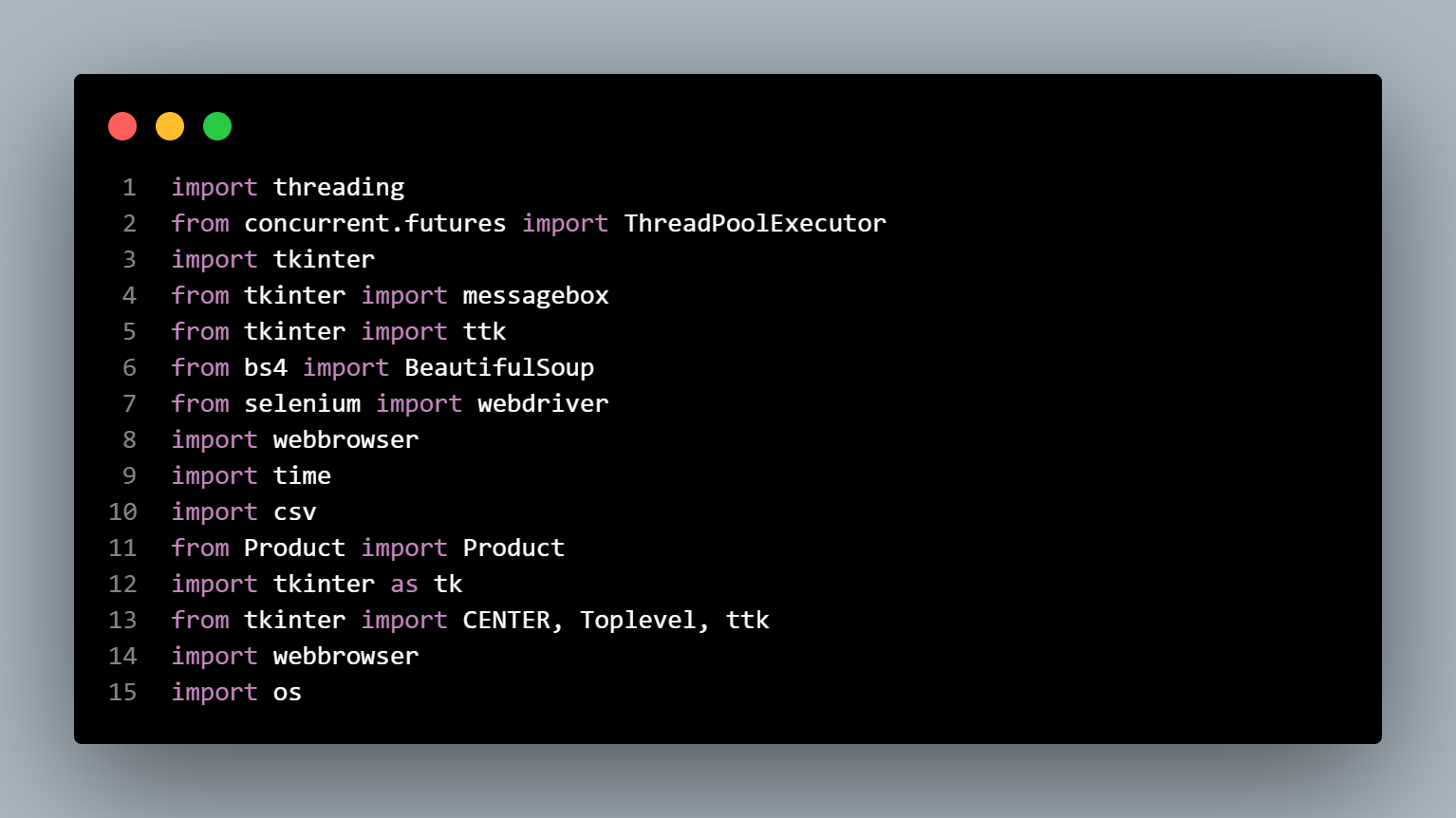
Sau khi xây dựng, tab2 sẽ có giao diện như sau



Hình 9. GUI của tab thứ 2

* Sau khi đã xây dựng xong GUI, ta gọi hàm mainloop() của Tk() để hiển thị lên màn hình

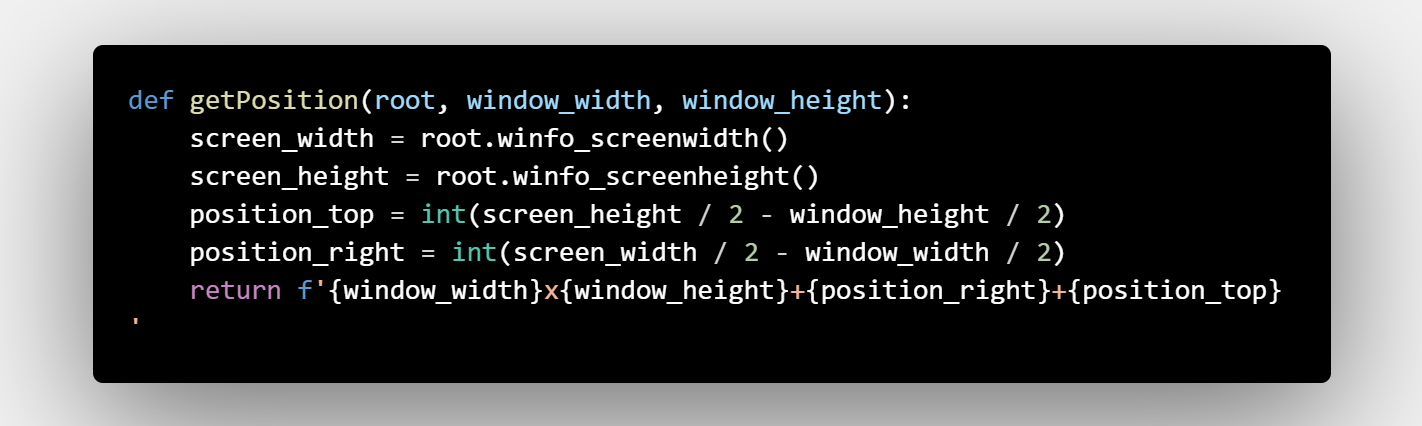
1. **Module shopeeFunctions.py dùng để xây dựng các hàm lấy về dữ liệu từ shopee**

* Các thư viện và module dùng để lấy data về bao gồm

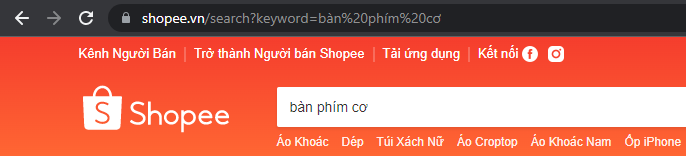
Hình 10. Các thư viện dùng trong shopeeFunctions.py

* Vì app có dùng thư viện selenium, do đó cần đưa đường dẫn tới Chromedriver vào biến PATH

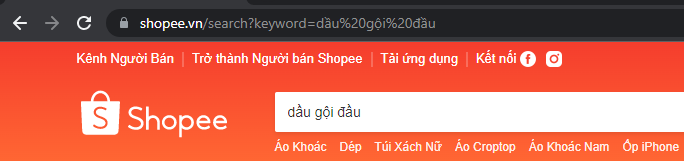
Hình 11. Đường dẫn tới Chromedriver

* Khi Frame xuất hiện, có thể xuất hiện ở vị trí mà người dùng không mong muốn, nhóm đã xây dựng hàm getPosition() để xác định vị trí của Frame ở giữa trên màn hình. Hàm này đúng ra là xây dựng ở init.py nhưng trong quá trình code, nhóm nhận thấy khi xây dựng ở init.py sẽ dẫn đến lỗi ImportError do hiện tượng 2 file import lẫn nhau dẫn đến tình trạng interpreter không biết là hàm nào được xây dựng trước.

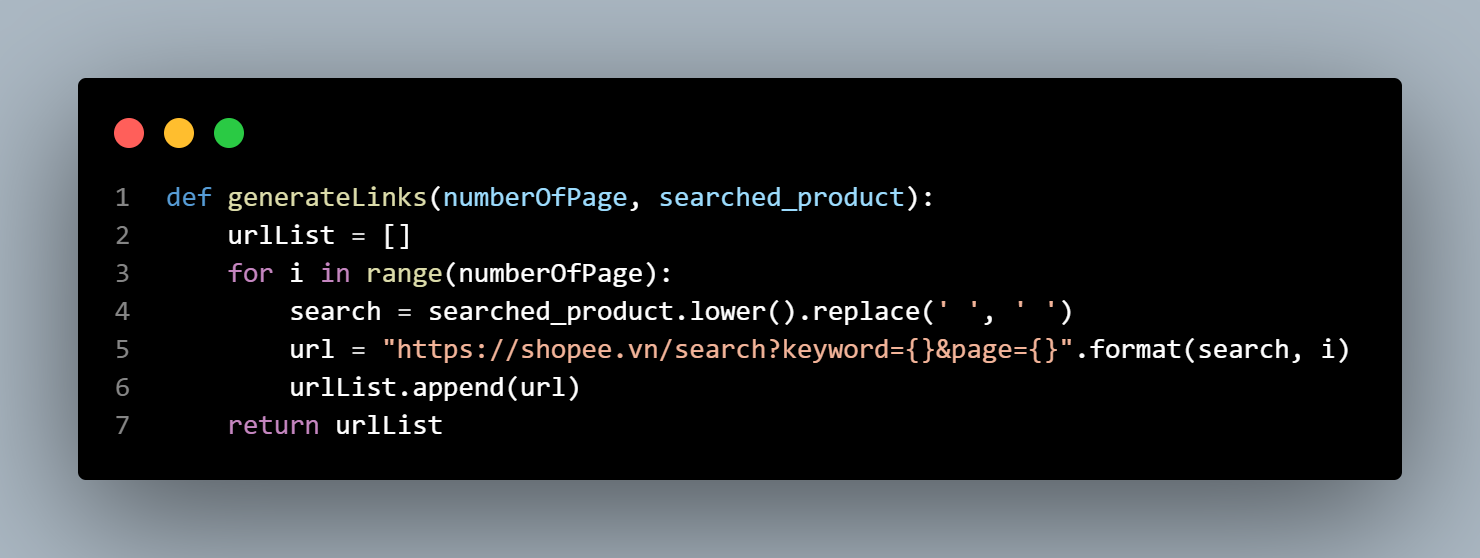
Hình 12. Hàm xác định vị trí frame trên màn hình

* **Phân tích đường link shopee**:
* Sau khi thực hiện tra cứu nhiều sản phẩm từ shopee thông qua tên tìm kiếm, nhận thấy rằng shopee tạo ra những link tìm kiếm có quy luật. Ví dụ

Hình 13. Ví dụ về link bản phím cơ

* Với sản phẩm là “bàn phím cơ”, sau khi ấn tìm kiếm, shopee tự động tạo ra link có dạng <https://shopee.vn/search?keyword=bàn%20phím%20cơ> , nhận thấy tên sản phẩm tìm kiếm xuất hiện trong link tìm kiếm với 1 chút thay đổi là thay vì dấu cách thì sẽ là “%20”
  + Hay với 1 sản phẩm khác như dầu gội đầu

Hình 14. Ví dụ về link dầu gội đầu

* + Cũng được dẫn đến kết quả tương tự
  + Thêm vào đó, số trang của tìm kiếm cũng được gắn vào trong link.
  + Với thứ tự trang có dạng i-1 với i là trang muốn tìm kiếm, ví dụ tìm kiếm trong trang 1 thì page=0, trang 2 thì page=1 …
* Kết luận về link:
  + Bắt đầu bằng: <https://shopee.vn/search?keyword=>
  + Tiếp theo, tên sản phẩm chứa dấu cách sẽ được thay bằng “%20” sau đó nối vào link
  + Thứ tự trang i có dạng “&page=” + i-1 sau đó được nối vào link
* Do đó, nhóm đã tạo ra 1 hàm dùng để tạo ra các link dẫn đến các trang web với số lượng trang được nhập vào

Hình 15. Hàm tạo link dẫn tới trang shopee

* Sau khi phân tích trang web, nhận thấy dùng thư viện BeautifulSoup không thể lấy được dữ liệu về sản phẩm từ trang web. Lý do là vì shopee.vn ngoài HTML, CSS còn được viết bằng Javascript, một ngôn ngữ lập trình động. Chính vì lý do đó, các thẻ chứa các sản phẩm chỉ được hiện lên khi người dùng cuộn đến vị trí của sản phẩm đó trên màn hình. Vì vậy, nhóm sẽ dùng thêm thư viện selenium, kết hợp với BeautifulSoup để lấy toàn bộ dữ liệu cần thiết

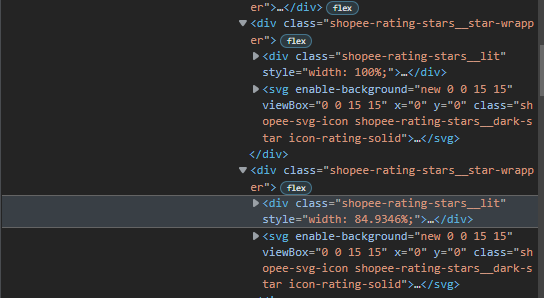
Hình 16. Hàm trả về source code web dạng BeautifulSoup

* Nhóm dùng thư viện selenium để mở trang web đó bằng trình duyệt Google Chrome với đường dẫn về Chromedriver đã được đề cập ở Hình 11. Sau khi mở trang web, chờ đợi 3s cho các thẻ ban đầu load. Sau đó, tự động cuộn trang web xuống 450 đơn vị, tự động lặp lại 15 lần cho đến khi đến cuối trang web. Lúc này, toàn bộ các thẻ đã được hiện ra kể cả những thẻ được viết bằng JS. Ta chỉ cẩn lấy source code của web và đưa nó về biến BeautifulSoup với định dạng ‘lxml’ là được. Ở đây, nhóm thực hiện bọc quá trình trong try except phòng khi lỗi xảy ra: PATH bị sai, có lỗi mạng, …. Hàm trả về dữ liệu dạng BeautifulSoup nếu không xảy ra lỗi, None nếu ngược lại
* **Phân tích source code trang web**

Hình 17. Minh họa cho source code trang web

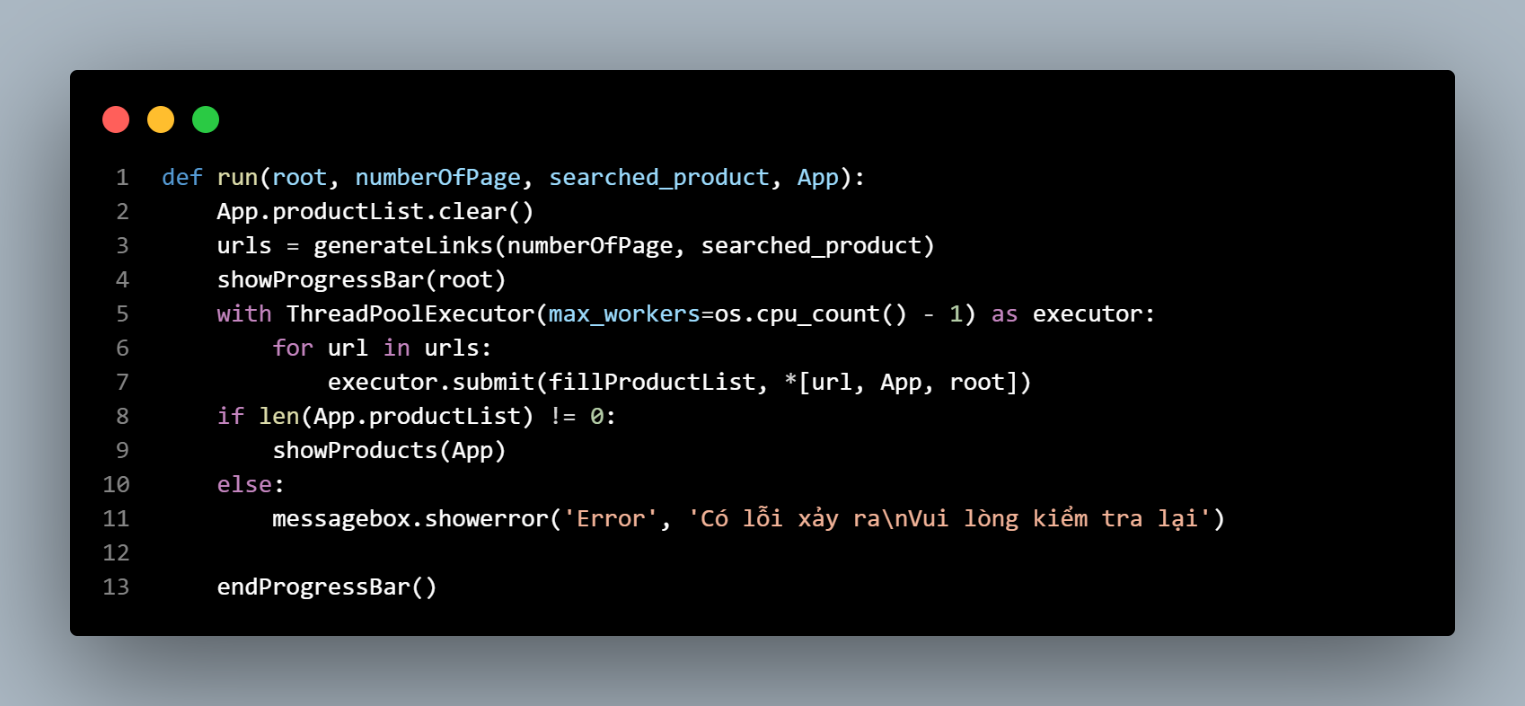
* Dùng công cụ devtools trong trang web, nhóm phân tích từng thành phần của trang web
  + Thẻ chứa sản phẩm: với thẻ div có class = “col-xs-2-4 shopee-search-item-result\_\_item” chính là thẻ chứa sản phẩm
  + Trong thẻ chứa sản phẩm, nhóm tìm ra:
    1. Thẻ chứa tên sản phẩm: div có class = “ie3A+n bM+7UW Cve6sh”
    2. Thẻ chứa giá nhỏ nhất, lớn nhất: span có class = “ZEgDH9”
    3. Lượt đánh giá sản phẩm của shop hiển thị không ở dạng số mà ở dạng hình ngôi sao được lấp đầy bởi màu vàng. Các ngôi sao này ở thẻ div có class = “shopee-rating-stars\_\_lit”.
    4. Thẻ chứa doanh số sản phẩm: div có class = “r6HknA uEPGHT”. Khác với các thuộc tính trên, doanh số có thể có hoặc không có, nếu không có, mặc định doanh số bằng 0
    5. Thẻ chứa link sản phẩm: a có thuộc tính “herf”
* Sau khi phân tích đầy đủ, nhóm sử dụng thư viện BeautifulSoup để xây dựng hàm fillProductList, dùng để thêm sản phẩm vào App.productList

Hình 18. Code cho phần phân tích bằng BeautifulSoup

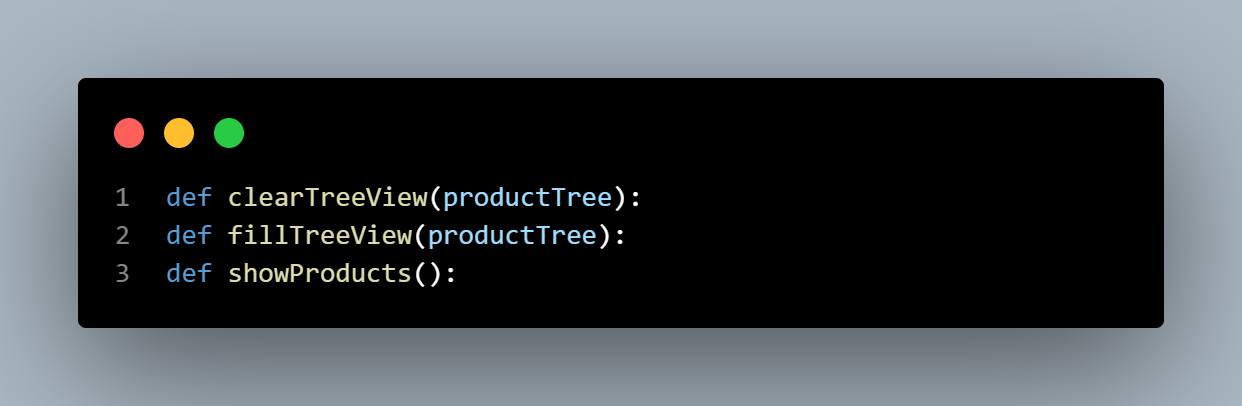
* + Danh sách sản phẩm được tìm bằng phương thức findAll(tag, attr), kết quả trả về là 1 list thuộc BeautifulSoup
  + Từ 1 phần tử, thuộc tính name, linkItem có thể dễ dàng lấy bằng phương thức find()
  + Một sản phẩm có thể có giá lớn nhất và nhỏ nhất hoặc chỉ có 1 giá, nhóm tìm tất cả giá của sản phẩm đó, nếu sản phẩm có 2 giá thì phần tử đầu tiên là giá nhỏ nhất, còn lại là lớn nhất. Ngược lại, giá lớn nhất bằng giá nhỏ nhất bằng giá trị tìm được
  + Sản phẩm có thể có đánh giá hoặc không. Nếu có thì đánh giá sản phẩm là 1 thuộc tính ẩn, thể hiện qua các ngôi sao, tỉ lệ bằng tỉ lệ của màu vàng của ngôi sao đó. Nhóm tìm tất cả các ngôi sao của sản phẩm đó, sau đó nhận thấy tỉ lệ mỗi ngôi sao nằm trong thuộc tính “style”, do đó lấy phần số của thuộc tính, ta được tỉ lệ 1 ngôi sao, cộng 5 ngôi sao lại ta được đánh giá của sản phẩm đó

Hình 19. Minh họa cho code đánh giá sản phẩm

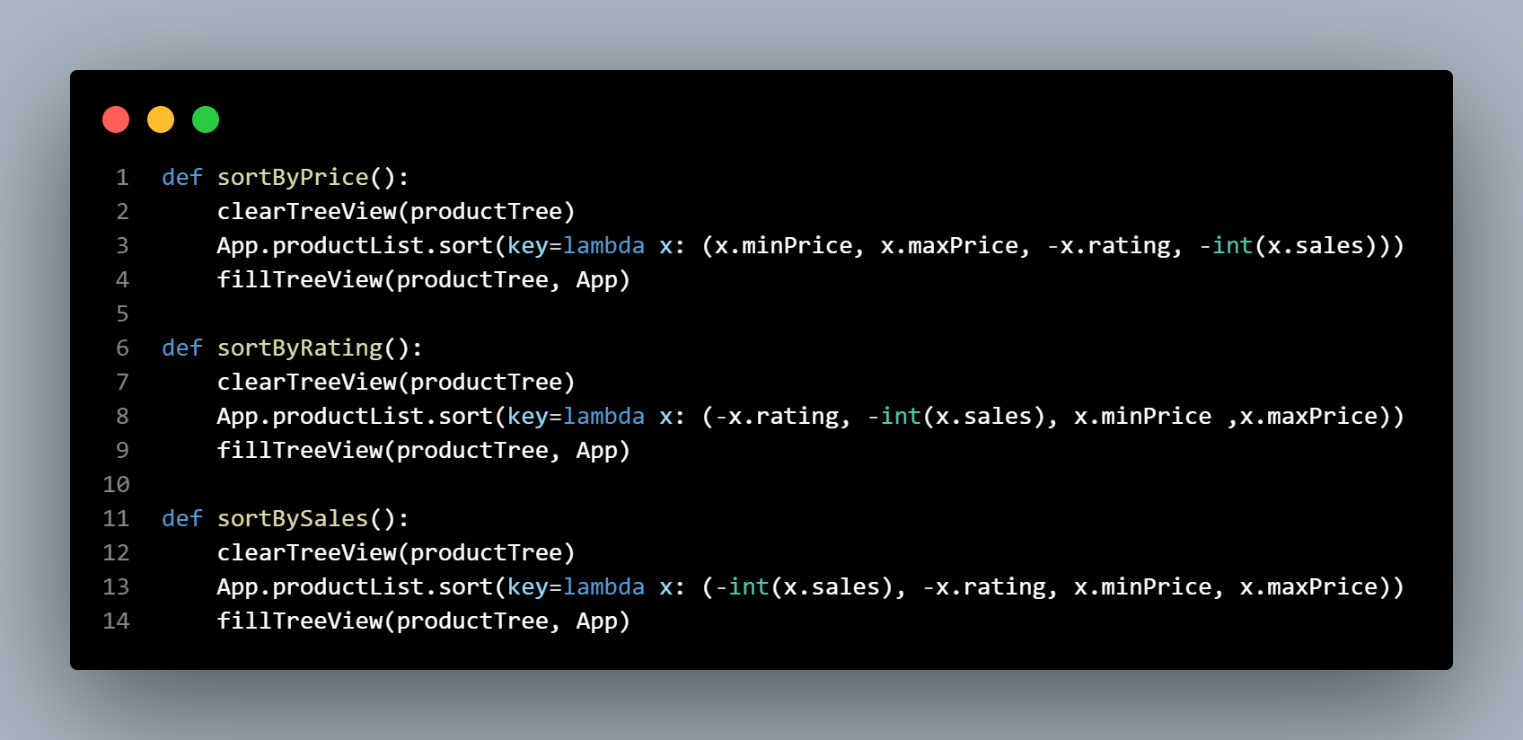
* + Tương tự, doanh số sản phẩm cũng có thể có hoặc không. Nếu có, kết quả trả về có thể ở dạng string (1,2k, 2,4k, …). Ta xóa dấu “,” và thay thế k bằng 000 sẽ thu được kết quả định dạng số cần tìm

* Khi đã xây dựng đủ các hàm phục vụ cho quá trình lấy thông tin sản phẩm, nhóm tiến hành xây dựng hàm để chạy các phương thức đó.

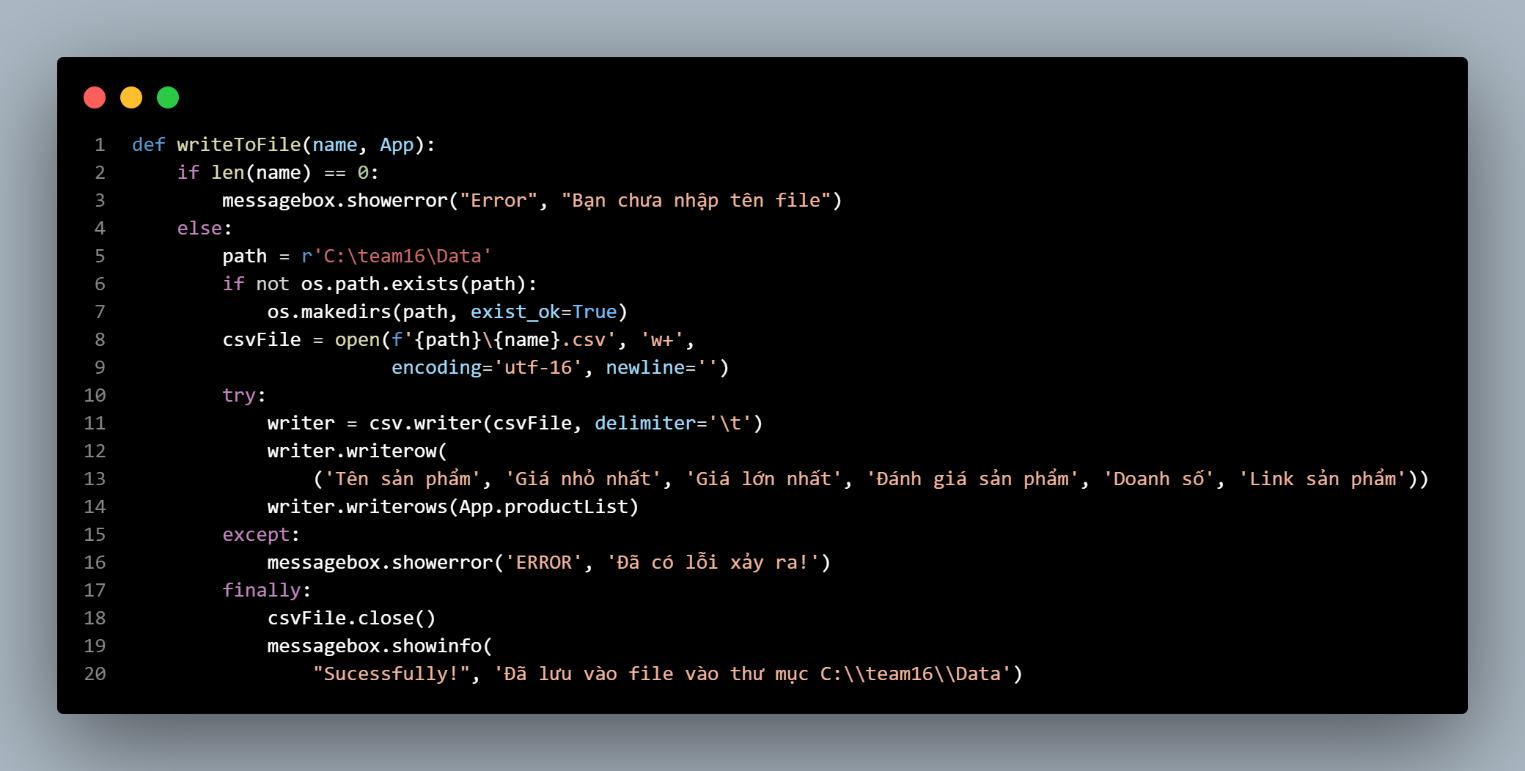
Hình 20. Hàm run để lấy từ nhiều trang

* + Vì ứng dụng có dùng thư viện selenium để mở trang web để lấy source code, thời gian mở từng web có thể sẽ rất lâu. Nhóm tối ưu thời gian đó bằng cách sử dụng ThreadPoolExecutor. Khi chạy, app sẽ tự động bật lên một số tab đồng thời. Số lượng phụ thuộc vào số nhân của CPU máy sử dụng. Ở đây, nhóm cho chạy số tab bằng số nhân CPU – 1, điều này nhằm đảm bảo bộ nhớ RAM và tốc độ CPU ổn định để có thể lấy dữ liệu thuận lợi. Thêm vào đó, sử dụng ThreadPoolExecutor sẽ chạy các tab ở các luồng khác nhau, từ đó tăng tốc độ lấy sản phẩm và cũng tránh được tình trạng not responding ở màn hình chính
* Sau khi có được danh sách sản phẩm từ shopee, nhóm xây dựng GUI để hiển thị danh sách lên màn hình. Nhóm sử dụng cấu trúc TreeView như 1 bảng hiển thị sản phẩm, xây dựng hàm clearTreeView() để xóa toàn bộ sản phẩm trên TreeView, fillTreeView() để thêm sản phẩm vào TreeView và hàm showProducts() để hiển thị TreeView lên màn hình.

Hình 21. Các hàm cho TreeView

* Trong đó, khi hiện thị danh sách sản phẩm lên có 3 tùy chọn để sắp xếp sản phẩm. Nhóm cũng đã xây dựng 3 hàm để sắp xếp list: sắp xếp theo giá, sắp xếp theo lượt đánh giá của sản phẩm và sắp xếp theo doanh số:

Hình 22. Các hàm sort productList

* Sau khi đã hiển thị được các sản phẩm lên màn hình, nhóm tiếp tục xây dựng hàm để lưu file vào file csv

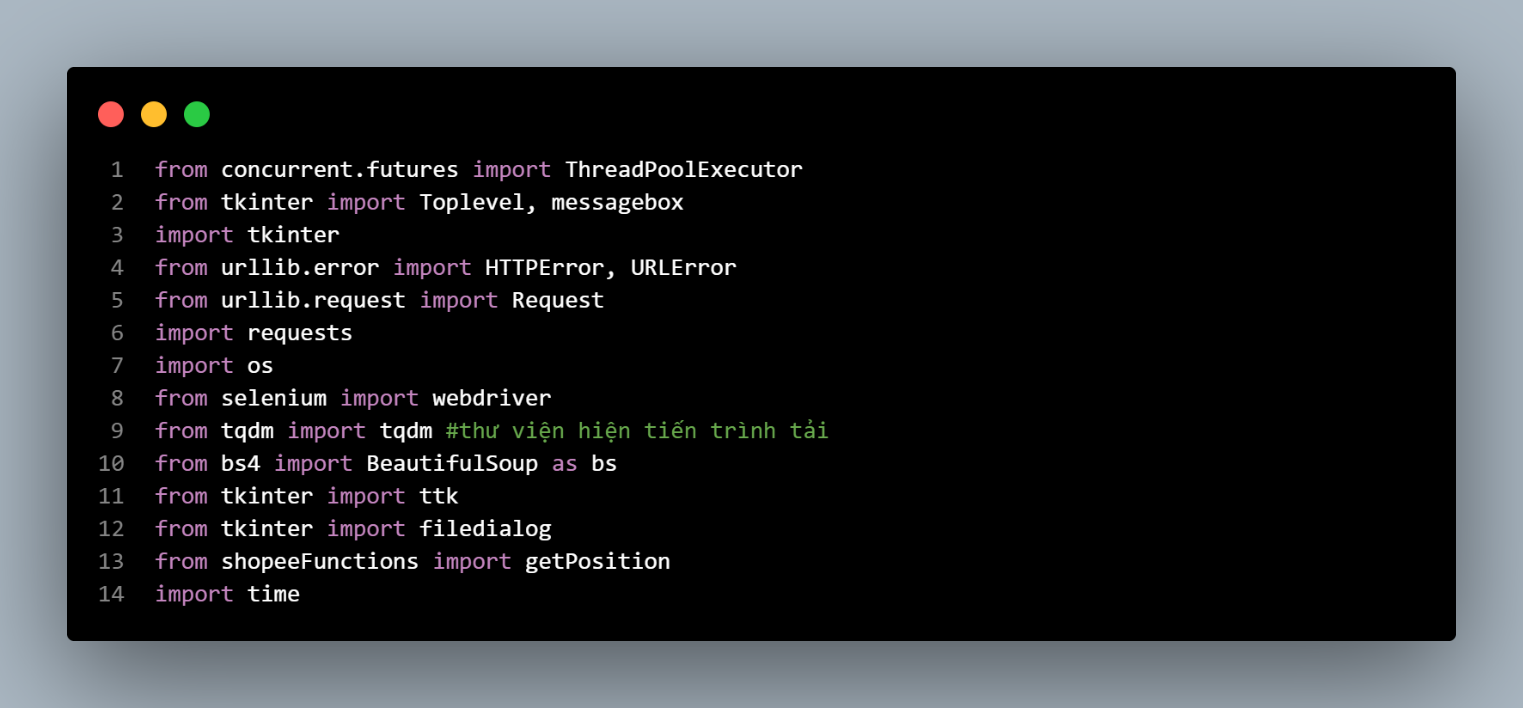
Hình 23. Hàm để lưu vào file CSV

Danh sách sản phẩm được lưu vào file ở thư mục C:\team16\Data. Nếu thư mục chưa tồn tại thì sẽ tự động tạo ra. File được mã hóa ở dạng “utf-16” vì có chứa kí tự tiếng Việt.

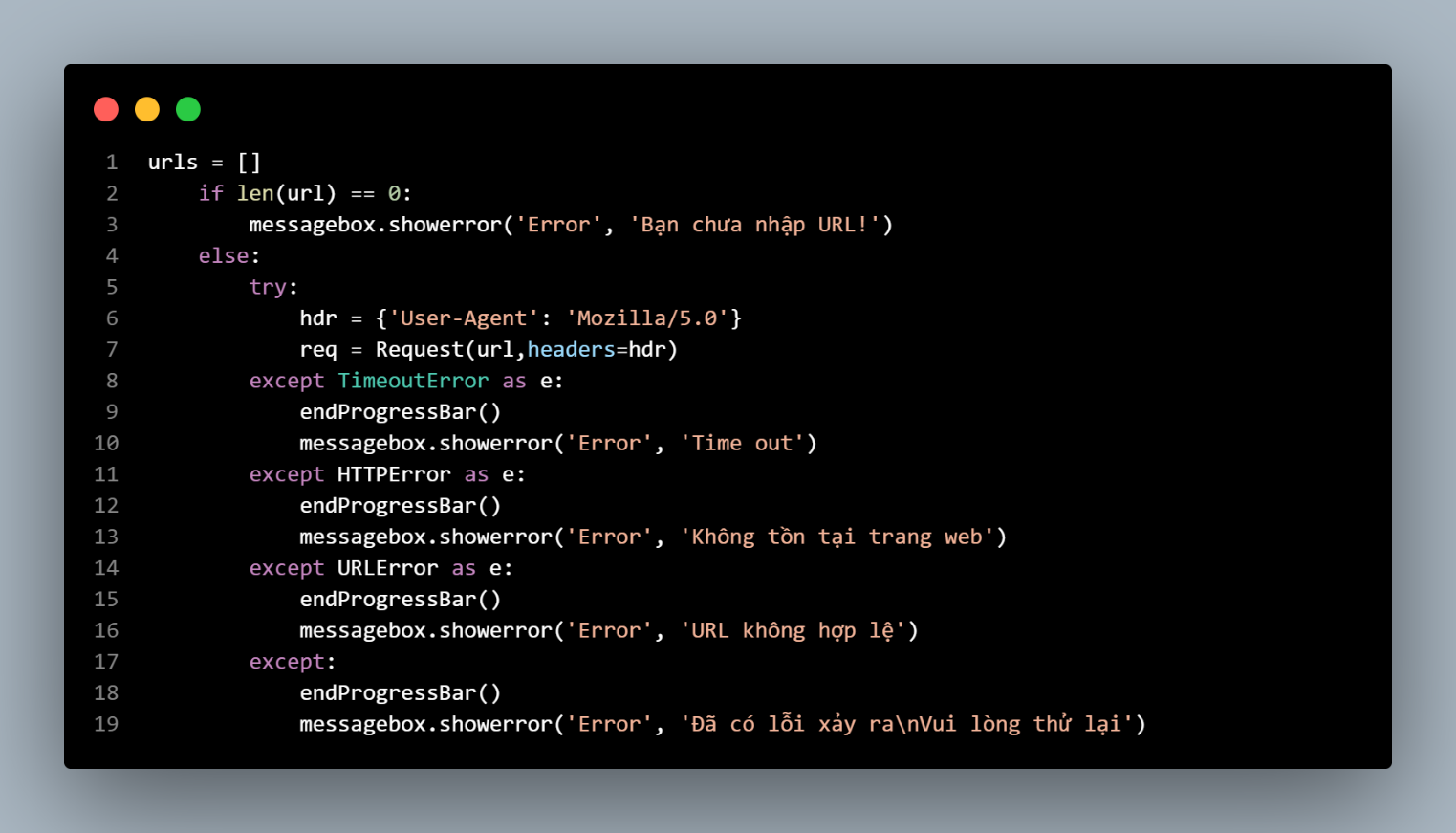
* Hàm accessToShopee() được xây dựng để gọi hàm run()

Hình 24. Hàm kết nối tới shopee

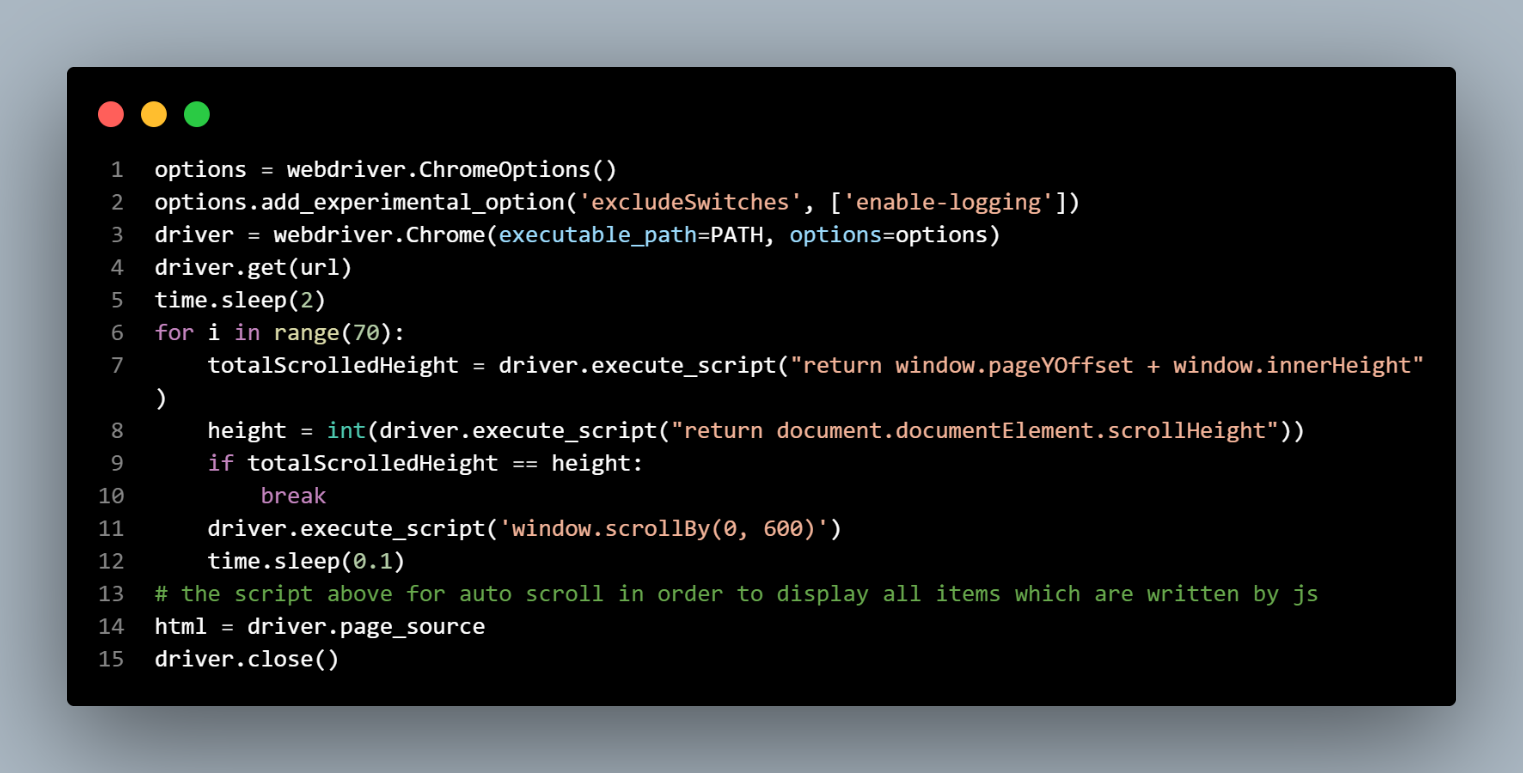
* + Hàm chứa các tham số đầu vào gồm: root là frame mẹ, numberOfPage là xâu lấy được từ entry eNumberOfPage từ init.py chứa số lượng trang muốn lấy, searchedProduct là tên sản phẩm muốn lấy, App chứa productList.
  + Vì vấn đề tốc độ nên nhóm chỉ giới hạn số trang ở 10.
  + Nếu chạy run ở dạng gọi hàm bình thường thì sẽ dẫn đến tình trạng tkinter bị not responding, do đó cần gọi hàm ở 1 thread khác để phần chương trình không bị tình trạng đơ, lag.

1. **Module imageFunctions.py dùng để xây dựng các hàm download ảnh từ 1 trang web**
   * Các thư viện và module sử dụng trong file

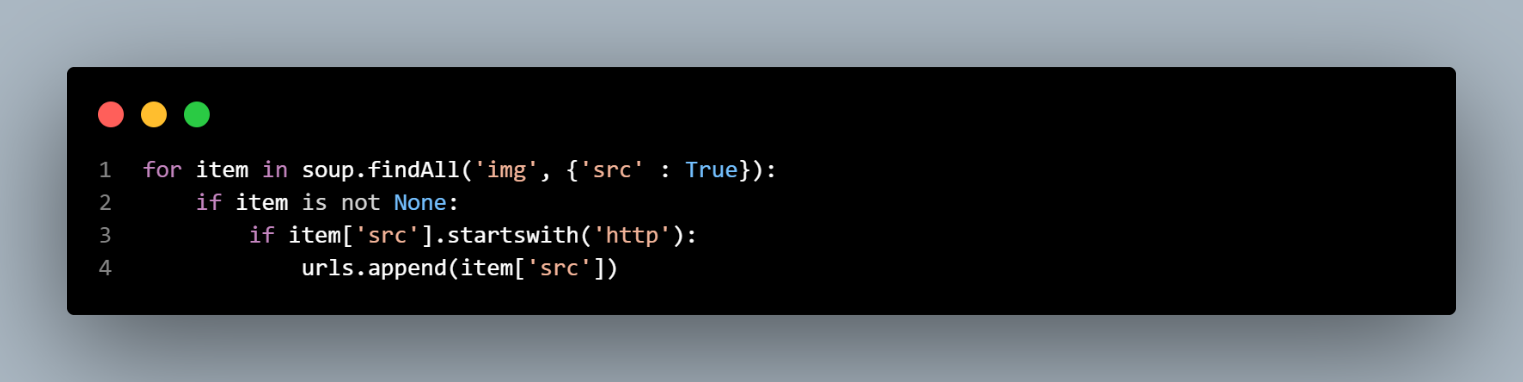
Hình 25. Các thư viện sử dụng trong imageFunctions.py

* Vì chức năng này lấy file theo đường link nên cần kiểm tra xem đường link có lỗi gì không, thực hiện bằng try except

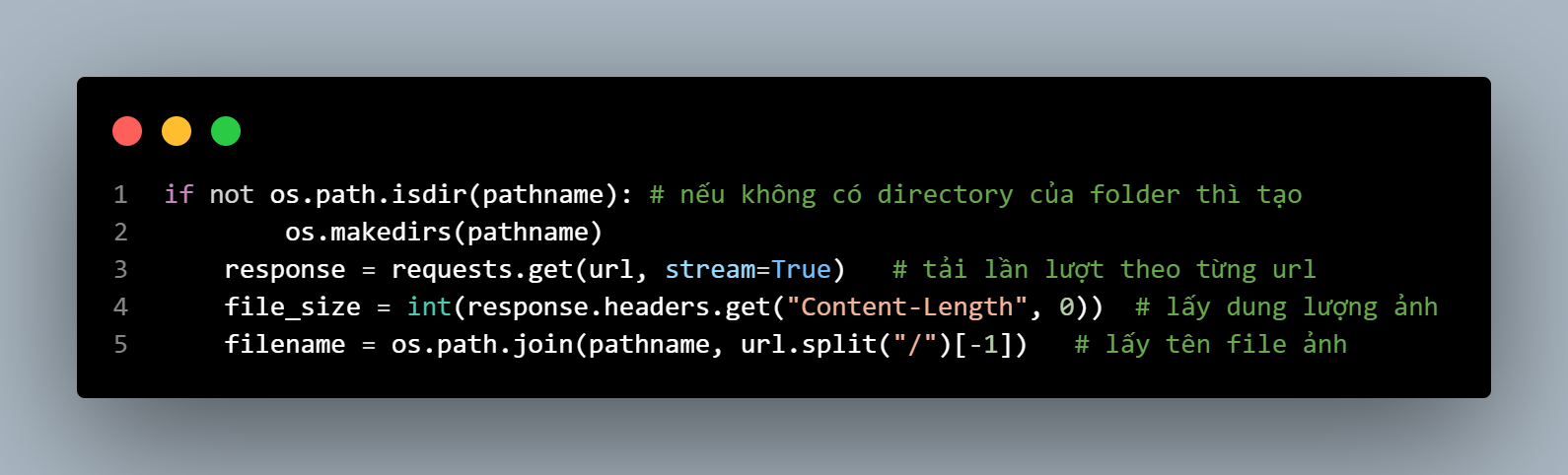
Hình 26. Try except đường dẫn nhập vào

* + Tương tự như phần lấy dữ liệu từ shopee, ngày nay rất nhiều trang web được viết bằng Javascript nên nhóm tiếp tục sử dụng selenium để lấy source code trang web.

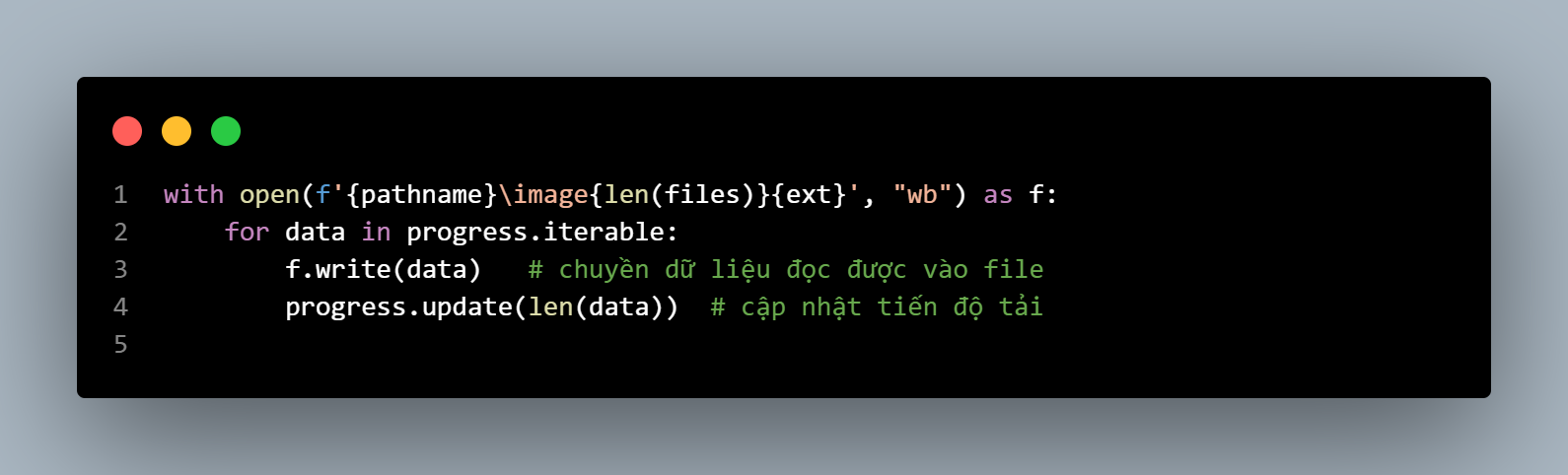
Hình 27. Sử dụng selenium để lấy source code

* + Để lấy được các ảnh, nhóm thực hiện tải trang web và tự động cuộn xuống 70 lần, mỗi lần 600 đơn vị. Thời gian chờ mỗi lần cuộn là 0.1s. Đồng thời trong lúc đó, nhóm cũng lấy ra chiều cao của page và vị trí của con trỏ ở page đó, nếu con trỏ bằng chiều cao page thì kết thúc vòng lặp. Sở dĩ nhóm chọn lặp 70 lần vì để hạn chế các web như linkedin, facebook, … các page có độ dài vô hạn sẽ làm chương trình không thể dừng nếu muốn lấy tất cả các ảnh từ web. 70 lần là con số phù hợp với số lượng ảnh từ 100 đến 200.
* Sau khi phân tích, nhận thấy các đường link ảnh thường nằm ở các thẻ img với attribute “src”.

Hình 28. Lấy về danh sách đường link ảnh

* Sau khi đã có url của các ảnh, thực hiện tải ảnh bằng requests

Hình 29. Lấy ảnh về bằng requests

* Rồi download ảnh về, phần extension của file được xác định trong hàm get\_url()

Hình 30. Download ảnh về folder

* Khi đã có list các url image, ta tiến hành download tất cả về.

Hình 31. Tiến hành download nhiều file cùng lúc bằng thread

* + Tương tự phần lấy dữ liệu từ shopee, ở đây nhóm cũng sử dụng ThreadPoolExecutor để lấy ảnh về nhắm tối ưu thời gian chờ của các ảnh. Hạn chế tình trạng not responding của app.

### **File chứa hàm để chạy: main.py**

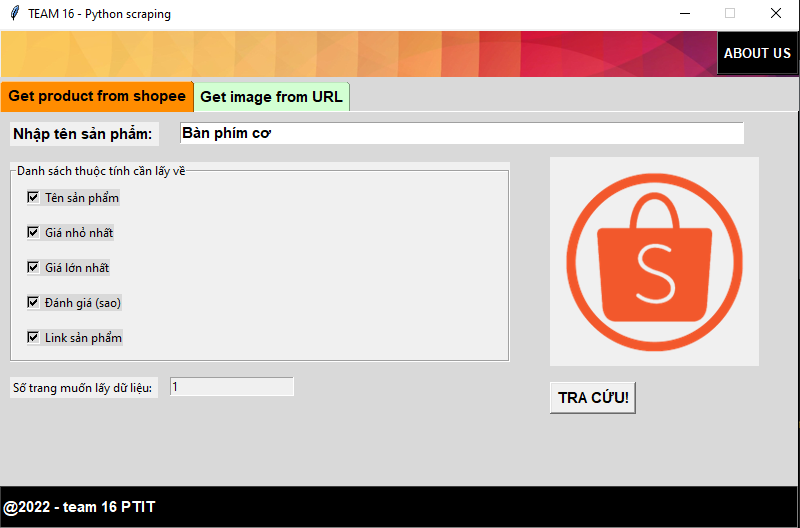
****

Hình 32. File main

1. **Cài đặt triển khai ứng dụng**

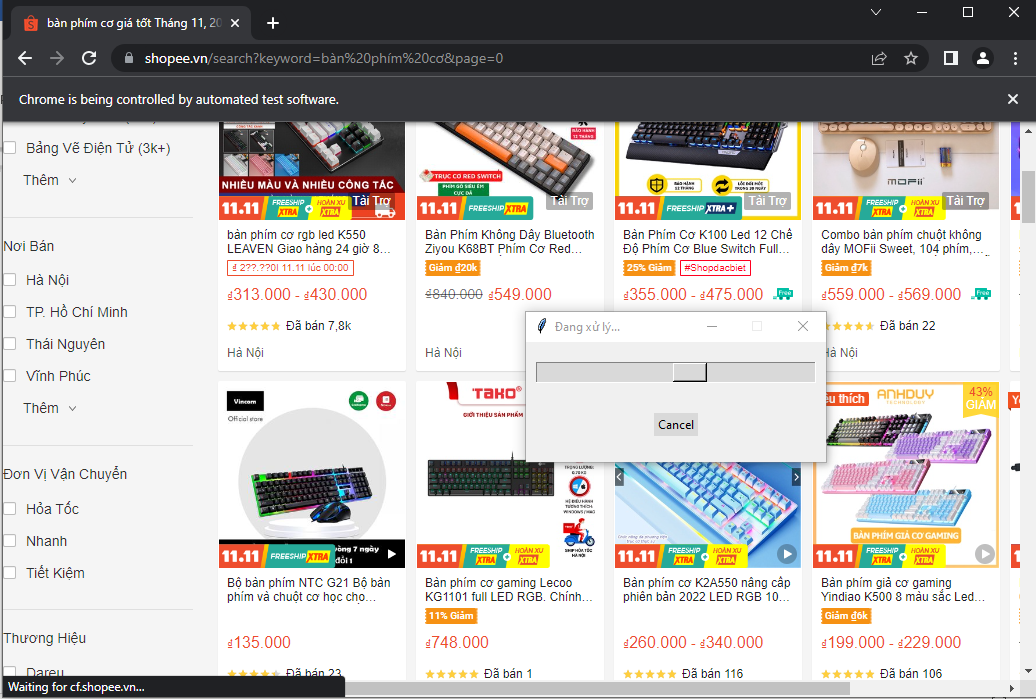
* Sau khi đã đảm bảo đủ các file, tiến hành cài đặt các thư viện ngoài, download Chromedriver và copy đường dẫn vào biến PATH ở 2 file shopeeFunctions.py và imageFunctions.py
* Cuối cùng chạy file main.py
* Hướng dẫn cụ thể đã được nhóm up lên github ở file README.md

1. **Kết quả thực nghiệm**

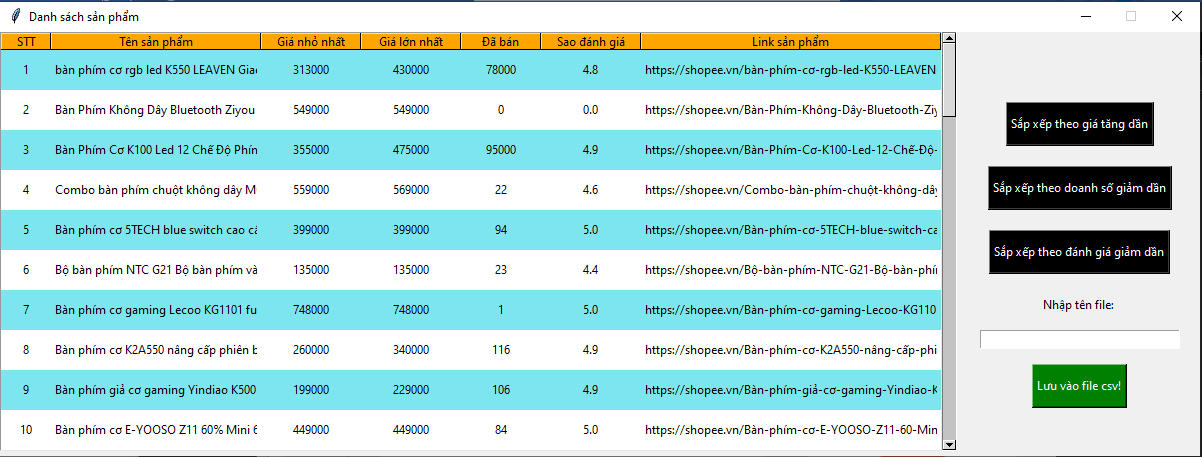
* **Tính năng 1:** Lấy danh sách sản phẩm từ shopee, Lấy danh sách bàn phím cơ

Hình 33. Demo lấy dữ liệu từ shopee

* Quá trình load trang web và xử lý dữ liệu

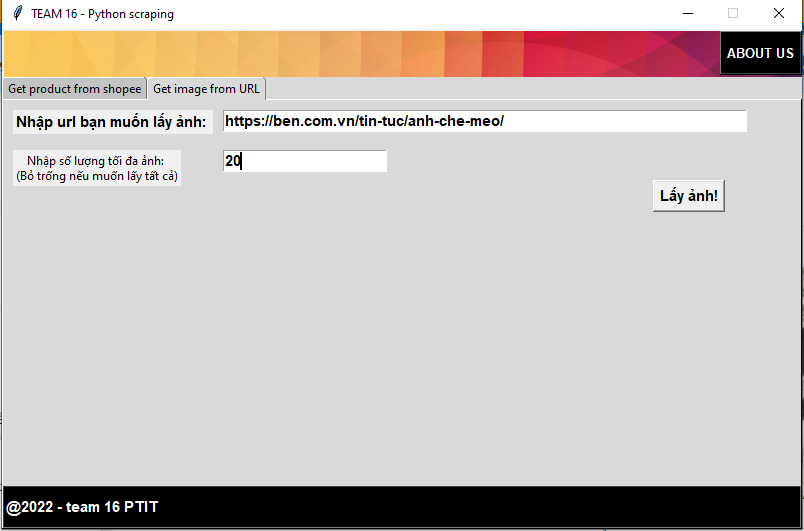


Hình 34. Demo xử lý dữ liệu từ shopee

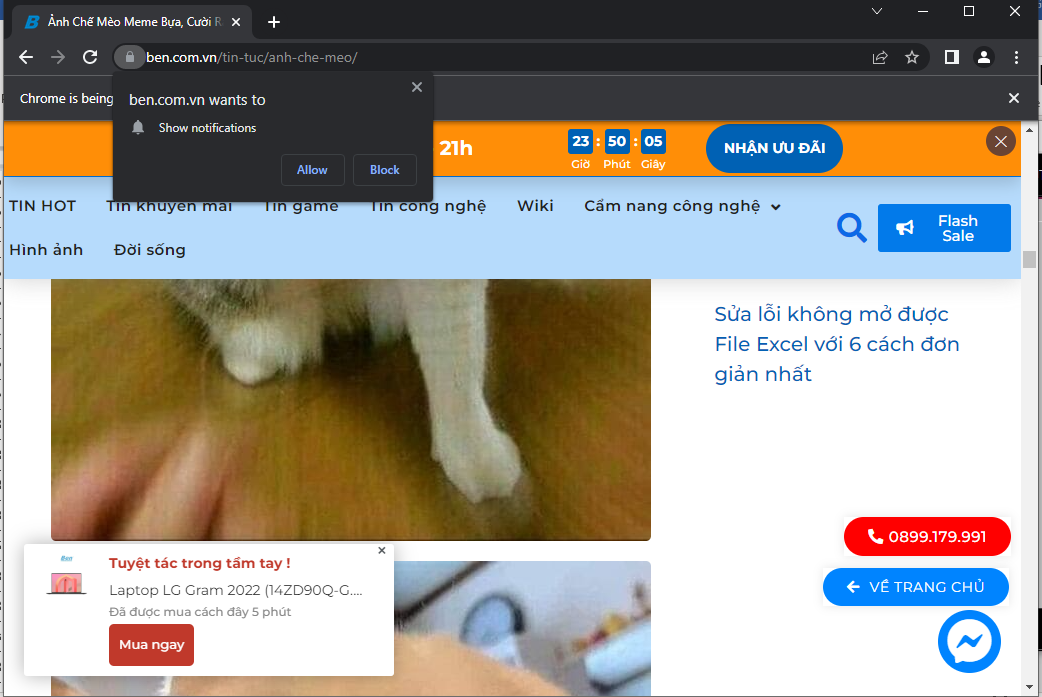
* Hiển thị dữ liệu lên màn hình

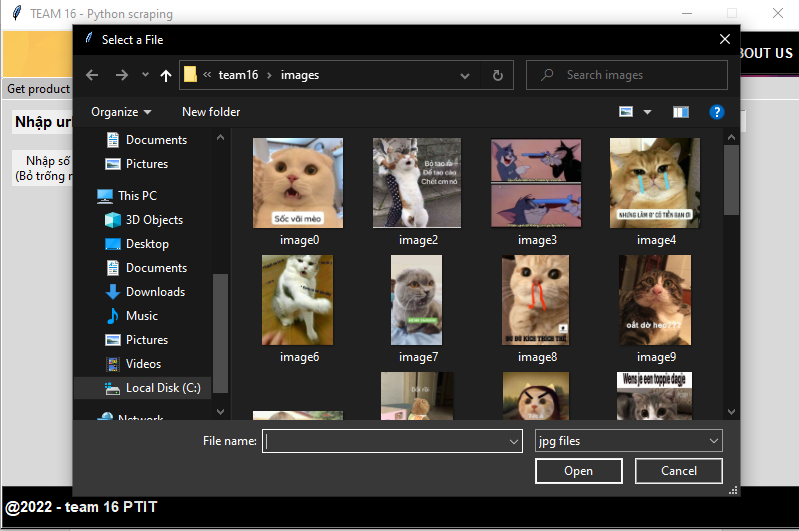
Hình 35. Demo hiển thị dữ liệu trên màn hình

* **Tính năng 2:** Tải ảnh từ 1 trang web lưu về folder C:\team16\images

****Hình 36. Demo tải ảnh từ 1 website

* Quá trình load trang web và tải ảnh

Hình 37. Demo load trang và xử lý dữ liệu trong quá trình tải ảnh

* Hiển thị kết quả tải ảnh

Hình 38. Demo kết quả sau khi tải ảnh

1. **Kết luận, hạn chế và phương hướng phát triển trong tương lai**

* **Kết luận:** App đã thực hiện được những chức năng cơ bản đã đặt ra
* **Hạn chế:**
  + Quá trình lấy dữ liệu còn tương đối lâu
  + Chưa chọn lọc được dữ liệu mà người dùng muốn lấy về
  + Chưa thực sự hiệu quả đối với những trang web sử dụng javascript
  + Còn nhiều exception chưa handle hết
* **Phương hướng phát triển trong tương lai:**
  + Tăng tốc lấy dữ liệu bằng các thư viện khác (scrapy, JSON, …)
  + Chọn lọc được dữ liệu mà người dùng muốn lấy về
  + Xây dựng tương thích với javascript
  + Sử dụng multiprocessing kết hợp với threading để tăng tốc độ và tăng số lượng sản phẩm

1. **Tài liệu tham khảo**

* <https://github.com/REMitchell/python-scraping.git>
* [(54) Khóa học Tkinter - Tạo giao diện người dùng đồ họa trong Hướng dẫn sử dụng Python - YouTube](https://www.youtube.com/watch?v=YXPyB4XeYLA)
* <https://stackoverflow.com>
* [Creating Tabbed Widget With Python-Tkinter - GeeksforGeeks](https://www.geeksforgeeks.org/creating-tabbed-widget-with-python-tkinter/?id=discuss)

# **ĐÓNG GÓP CỦA CÁC THÀNH VIÊN**

1. Trần Quang Huy B20DCCN327
   * Code phần backend: shopeeFunctions.py
   * Code phần frontend: init.py
   * Là người trực tiếp tổng hợp các file trước khi push lên github
   * Chịu trách nhiệm làm notebook
2. Nguyễn Cảnh Hưng B20DCCN339
   * Code phần hiển thị TreeView
   * Cùng làm noteboook
   * Tham gia chỉnh sửa file imageFunction.py
   * Tìm hiểu BeautifulSoup
3. Nguyễn Tiến Anh B20DCCN005
   * Tìm hiểu phần selenium
   * Code phần backend: imageFunctions.py
   * Thuyết trình
4. Nguyễn Hồng Hiếu B20DCCN252
   * Tìm hiểu scraping
   * Làm Slide
   * Chỉnh sửa file notebook