



Amazon Product Price Tracker

Developed With: Python, HTML, CSS

# Introduction

AMZSPY is an Amazon product price-tracking software program that aims to allow consumers to get notifications on Amazon products they would like to purchase in the future. This program aims to solve a simple consumer issue in today’s commerce; to bridge the gap between searching for attractive deals on products consumers desire and allow the product deals to go directly to them in their email inbox. This application only checks for price drops on products you specify and requires a URL and email to send to. There is a website implementation and automated real-time database connected to a Python scraping program. AMZSPY will solve the issue of having to buy products you desire at full price and instead offers you a chance to buy the same exact product at a lower price, in exchange for an email to store, a product to track, and a commission that is compensated towards the developer at no cost to the user.

# Resources Used For This Project:

UX/UI:

<https://getbootstrap.com/docs/4.4/getting-started/introduction/>

Optional Proxy:

<https://blog.scrapinghub.com/python-requests-proxy>

Python Scraper Inspiration:

<https://hackernoon.com/scraping-amazon-product-information-with-python-and-beautifulsoup-yn4s3tgr>

# Example Usage of AMZSPY

Situation 1:

A consumer would like to be notified of an Amazon product price drop without having to search through the internet themselves, which could be tedious and unsuccessful with time wasted. AMZSPY will scrape the product hourly until the price drops and sends an email directly to the consumer with no time spent on the user’s behalf other than a relatively quick website submit form.

Situation 2:

A consumer is looking for a lower price on various Amazon products they desire but cannot find any deals and do not have the required time to search for every single product’s potential deals. The user can enter their email and URL and have full assurance that they’ll be notified whenever the price drops on whatever products they would like.

Situation 3:

A reseller is looking to find out which products and competitors are currently dropping product prices or keeping their prices the same or higher in real-time in order to decide which products to list. Then through the products they choose to list, their products can be placed at a desirable price for their profit margins on in-demand products that consumers are currently buying.

# Current Code

Github Repository:

<https://github.com/helloandysok/helloandysok.github.io>

Software code as of now:

import requests  
import traceback  
import smtplib  
import time  
from bs4 import BeautifulSoup  
from lxml.html import fromstring  
from itertools import cycle  
# For Realtime database  
from firebase import firebase  
# Initialize Admin SDK from Firebase  
import firebase\_admin  
from firebase\_admin import credentials, firestore  
  
cred = credentials.Certificate('amzspy-ed060-firebase-adminsdk-ombif-5fa1965a13.json')  
default\_app = firebase\_admin.initialize\_app(cred)  
db = firestore.client()  
  
# Get a URL from the user  
URL = input("Copy & Paste The Amazon URL You Would Like To Price Track: ")  
  
# For utilizing the Chrome Browser  
headers = {"User-Agent": 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_15\_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.117 Safari/537.36'}  
  
# User info:  
email\_list = input("Enter Your Email: (Ex: andysok@gmail.com) ")  
  
# Realtime database creation  
firebase = firebase.FirebaseApplication("https://amzspy-ed060.firebaseio.com/", None)  
data = {  
 'Email': email\_list  
}  
  
  
# OOP Component of Polymorphism  
def discount\_conversion(a, b):  
 return a / b  
  
  
# Track price for user  
def price\_tracker():  
  
 # Get the page you're searching for  
 page = requests.get(URL, headers=headers)  
  
 # Used to bypass anti-scraping  
 soup\_old = BeautifulSoup(page.content, "html.parser")  
  
 # Used to gather data on Amazon product page  
 soup = BeautifulSoup(soup\_old.prettify(), "html.parser")  
  
 # Find the title  
 title = soup.find(id="productTitle").get\_text()  
  
 # Find the full price  
 full = soup.find(class\_="priceBlockStrikePriceString a-text-strike").get\_text()  
  
 # Find the price  
 price = soup.find(id="priceblock\_ourprice").get\_text()  
  
 # Strip the price of non-numbers  
 stripped = price.strip("$ ,")  
 stripped\_full = full.strip("$ ,")  
  
 # Replace the commas with empty character  
 replacer = stripped.replace(",", "")  
 replace\_full = stripped\_full.replace(",", "")  
  
 # Find the dot in the price  
 find\_dot = replacer.find(".")  
 find\_full = replace\_full.find(".")  
  
 # Convert the price by removing the cents  
 changer = replacer[0:find\_dot]  
 change\_full = replace\_full[0:find\_full]  
  
 # The price, fully converted  
 converter = float(changer)  
 convert\_full = float(change\_full)  
  
 # Store the current price for comparison  
 current\_price = converter  
 current\_full = convert\_full  
  
 # Print the original price, product title, and current sale price  
 print("Your Product: ")  
 print(title.strip())  
 print("Your List Price: ")  
 print(full)  
 print("Your Price: ")  
 print(price)  
  
 # input, Lambda & polymorphism to determine discount  
 discount = float(input("How big of a discount are you looking for? (Ex: 10 for 10% off) "))  
 get\_discount = (lambda a, b, c, d: ((a/b) - c) \* d)(current\_price, current\_full, 1, -1)  
 percentage\_shift = discount\_conversion(float(discount), 100)  
  
 # Check hourly for price shifts  
 while get\_discount <= percentage\_shift:  
 print("Your product is not at the desired discount yet. We will check hourly to see any price differences.")  
 print("You will receive an email in your inbox once your discount is available..")  
 time.sleep(3600)  
  
 if get\_discount >= percentage\_shift:  
 sender(email\_list)  
  
  
def sender(email\_list):  
  
 # Connect email server and email account  
 server = smtplib.SMTP('smtp.gmail.com', 587)  
 server.ehlo()  
 # Starts the encrypted communication with the server  
 server.starttls()  
 server.ehlo()  
 server.login('endlesskhaiandy@gmail.com', 'ivdnvcsckaporjco')  
  
 # Email message to send to list / Add the URL variable in the body of the email  
 subject = 'New Amazon Product Price Alert!'  
 body = 'Hey Smart Shopper, a product you want has a lower price! ' \  
 'Check it out by clicking the link: ', URL  
 msg = f"Subject: {subject}\n\n{body}"  
  
 # Send the mail!  
 server.sendmail('endlesskhaiandy@gmail.com', email\_list, msg)  
  
 # Email Confirmation  
 print('Thanks for your patience; Check your email! ')  
  
 # Terminate  
 server.quit()  
  
# Run the price tracker  
price\_tracker()  
  
  
# Create an CRUD database using firebase, but switch around variables to complement this integration

# Final Code Report:

How to use:

1. Go to Ngrok or Manage.py directory
2. Runserver or Ngrok server

# URLS.PY

from django.urls import path  
from django.conf.urls import url  
from . import views  
  
urlpatterns = [  
 path('', views.home, name="home"),  
 url(r'^home', views.home, name="home")  
]

# VIEWS.PY

from django.shortcuts import render, redirect  
from .models import User  
from .forms import UserForm  
from django.contrib import messages  
from django.http import HttpResponse  
  
  
# Create your views here.  
  
  
def home(request):  
 import requests  
 import traceback  
 import smtplib  
 import time  
 from bs4 import BeautifulSoup  
 from lxml.html import fromstring  
 from itertools import cycle  
 # For Realtime database  
  
 if request.method == 'POST':  
  
 # Get a URL from the user  
 URL = request.POST.get('amzurl', None)  
  
 # For utilizing the Chrome Browser  
 headers = {  
 "User-Agent": 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_15\_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.117 Safari/537.36'}  
  
 # User info:  
 email\_list = request.POST.get('email', None)  
  
 # OOP Component of Polymorphism  
 def discount\_conversion(a, b):  
 return a \* b  
  
 def price\_check():  
 # Get the page you're searching for  
 new\_page = requests.get(URL, headers=headers)  
  
 # Used to bypass anti-scraping  
 new\_soup\_old = BeautifulSoup(page.content, "html.parser")  
  
 # Used to gather data on Amazon product page  
 new\_soup = BeautifulSoup(soup\_old.prettify(), "html.parser")  
  
 # Find the title  
 new\_title = soup.find(id="productTitle").get\_text()  
  
 # Find the full price  
 new\_full = soup.find(class\_="priceBlockStrikePriceString a-text-strike").get\_text()  
  
 # Find the price  
 new\_price = soup.find(id="priceblock\_ourprice").get\_text()  
  
 # Strip the price of non-numbers  
 new\_stripped = new\_price.strip("$ ,")  
 new\_stripped\_full = new\_full.strip("$ ,")  
  
 # Replace the commas with empty character  
 new\_replacer = new\_stripped.replace(",", "")  
 new\_replace\_full = new\_stripped\_full.replace(",", "")  
  
 # Find the dot in the price  
 new\_find\_dot = new\_replacer.find(".")  
 new\_find\_full = new\_replace\_full.find(".")  
  
 # Convert the price by removing the cents  
 new\_changer = new\_replacer[0:new\_find\_dot]  
 new\_change\_full = new\_replace\_full[0:new\_find\_full]  
  
 # The price, fully converted  
 new\_converter = float(new\_changer)  
 new\_convert\_full = float(new\_change\_full)  
  
 # Store the current price for comparison  
 new\_current\_price = new\_converter  
 new\_current\_full = new\_convert\_full  
 return new\_current\_price  
  
 def sender(email\_list):  
  
 # Connect email server and email account  
 server = smtplib.SMTP('smtp.gmail.com', 587)  
 server.ehlo()  
 # Starts the encrypted communication with the server  
 server.starttls()  
 server.ehlo()  
 server.login('endlesskhaiandy@gmail.com', 'ivdnvcsckaporjco')  
  
 # Email message to send to list / Add the URL variable in the body of the email  
 subject = 'New Amazon Product Price Alert!'  
 body = 'Hey Smart Shopper, a product you want has a lower price! ' \  
 'Check it out by clicking the link: ', URL  
 msg = f"Subject: {subject}\n\n{body}"  
  
 # Send the mail!  
 server.sendmail('endlesskhaiandy@gmail.com', email\_list, msg)  
  
 # Terminate  
 server.quit()  
  
 # Get the page you're searching for  
 page = requests.get(URL, headers=headers)  
  
 # Used to bypass anti-scraping  
 soup\_old = BeautifulSoup(page.content, "html.parser")  
  
 # Used to gather data on Amazon product page  
 soup = BeautifulSoup(soup\_old.prettify(), "html.parser")  
  
 # Find the title  
 title = soup.find(id="productTitle").get\_text()  
  
 # Find the full price  
 full = soup.find(class\_="priceBlockStrikePriceString a-text-strike").get\_text()  
  
 # Find the price  
 price = soup.find(id="priceblock\_ourprice").get\_text()  
  
 # Strip the price of non-numbers  
 stripped = price.strip("$ ,")  
 stripped\_full = full.strip("$ ,")  
  
 # Replace the commas with empty character  
 replacer = stripped.replace(",", "")  
 replace\_full = stripped\_full.replace(",", "")  
  
 # Find the dot in the price  
 find\_dot = replacer.find(".")  
 find\_full = replace\_full.find(".")  
  
 # Convert the price by removing the cents  
 changer = replacer[0:find\_dot]  
 change\_full = replace\_full[0:find\_full]  
  
 # The price, fully converted  
 converter = float(changer)  
 convert\_full = float(change\_full)  
  
 # Store the current price for comparison  
 current\_price = converter  
 current\_full = convert\_full  
  
 temp\_price = current\_price  
  
 # input, Lambda & polymorphism to determine discount  
 # discount = float(input("How big of a discount are you looking for? (Ex: 10 for 10% off) "))  
 get\_discount = (lambda a, b, c, d: ((a / b) - c) \* d)(current\_price, current\_full, 1, -1)  
 percentage\_shift = round(discount\_conversion(float(get\_discount), 100), 2)  
  
 # Call price check function  
 price\_check()  
  
 # Check hourly for price shifts  
 while price\_check() >= temp\_price:  
 messages.success(request, {'The Product: ': title})  
 messages.success(request, {'The Current Price: ': price})  
 messages.success(request, {'The Current Discount In Percentage: ': percentage\_shift})  
 messages.success(request, {'Tracker Used. If The Price Drops, We Will Email You!'})  
 time.sleep(3600)  
 price\_check()  
  
 # Send email if the price is lowered  
 if price\_check() < temp\_price:  
 sender(email\_list)  
 messages.success(request, {'The Product: ': title})  
 messages.success(request, {'The Current Price: ': price})  
 messages.success(request, {'The Current Discount In Percentage: ': percentage\_shift})  
 return render(request, "home.html", messages.success(request, 'Product Successfully Imported! Check Your Inbox For Any Notifications!'))  
  
 return render(request, "home.html", messages.success(request, 'Tracker Used. If The Price Drops, We Will Email You!'))  
 else:  
 return render(request, "home.html", {})

# BASE.HTML

<!-- GetBootstrap template by getbootstrap.com, Design by Andy Sok-->  
<!doctype html>  
<html lang="en">  
 <head>  
 <!-- Required meta tags -->  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">  
  
 <!-- Bootstrap CSS -->  
 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">  
  
 <!-- Page name -->  
 <title>AMZSPY - Amazon Product Price Tracker By Andy Sok</title>  
  
 </head>  
 <!-- Navbar -->  
<nav class="navbar navbar-expand-lg navbar-dark" style="background-color:#222e3e;">  
 <a class="navbar-brand" href="{% url 'home' %}">AMZSPY By Andy Sok</a>  
 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">  
 <span class="navbar-toggler-icon"></span>  
 </button>  
 <div class="collapse navbar-collapse" id="navbarNavAltMarkup">  
 <div class="navbar-nav">  
 <a class="nav-item nav-link active" href="{% url 'home' %}">Home <span class="sr-only">(current)</span></a>  
 <a class="nav-item nav-link" href="#how">How It Works</a>  
 <a class="nav-item nav-link" href="https://github.com/helloandysok/helloandysok.github.io">Documentation</a>  
 <a class="nav-item nav-link" href="mailto:contact@amzspy.us">Need Help? Email us: contact@amzspy.us</a>  
 </div>  
 </div>  
</nav>  
 <div class="container">  
 <br>  
 <h2 class="text-primary">WELCOME TO AMZSPY</h2>  
 <h4>The Newest Amazon Price Tracking Tool!</h4>  
 <br>  
 <p>Simple & Easy To Use!</p>  
 <p>Just Enter Your Amazon Product URL & Email Down Below:</p>  
 </br>  
  
 <!--img src="AMZ.png" class="img-fluid" alt="amzspy.png"-->  
 <form method="POST">  
 {% csrf\_token %}  
 <div class="form-group">  
 <label for="inputurl">Enter Your Amazon Product URL: </label>  
 <input type="amzurl" name="amzurl" class="form-control" id="inputurl" aria-describedby="amz">  
 <small id="amz" class="form-text text-muted">Copy & Pasting Makes It 100% Easier For Both of Us.</small>  
 </div>  
 <div class="form-group">  
 <label for="email">Enter Your Email: </label>  
 <input type="email" name="email" class="form-control" id="email">  
 <small id="emailmsg" class="form-text text-muted">We will NEVER spam you. We HATE spam.</small>  
  
 </div>  
 <div class="form-group form-check">  
 <input type="checkbox" class="form-check-input" id="check">  
 <label class="form-check-label" for="check">Check here to get discounts on similar items sent to your inbox.</label>  
 </div>  
 <button onclick="location.href='{% url 'home' %}'" type="submit" class="btn btn-primary">Track My Amazon Product</button>  
 </form>  
 <div style="padding-top:25px;">  
 {% block content %}  
 {% endblock %}  
 </div>  
  
 </br>  
  
 <!-- In-app help section -->  
 <div class="p-3 mb-2 text-white" style="background-color:#222e3e;" >  
 <br>  
 <div class="container">  
 <h4 id="how">How AMZSPY Works:</h4>  
 <p>1. You copy & paste your Amazon Product URL.</p>  
 <p>2. We scrape the data for you.</p>  
 <p>3. You input your desired discount amount.</p>  
 <p>4. Enter your email.</p>  
 <p>5. We send you an email when your product has a lower price! VOILA!</p>  
 </div>  
 </div>  
 <p class="text-muted" style="padding-top:25px;">By submitting a form, we may be compensated for sales derived through this website at no extra cost to you.</p>  
 </div>  
  
 <!-- Optional JavaScript -->  
 <!-- jQuery first, then Popper.js, then Bootstrap JS -->  
 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin="anonymous"></script>  
 <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>  
 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>  
 </body>  
</html>

# HOME.HTML

{% extends 'base.html' %}  
  
{% block content %}  
  
 <!-- bootstrap flashing message -->  
 {% if messages %}  
 {% for message in messages %}  
 <div class="alert alert-warning alert-dismissable" role=""alert>  
 <button class="close" data-dismiss="alert">  
 <small><sup>Close</sup></small>  
 </button>  
 {{message}}  
 </div>  
 {% endfor %}  
{% endif %}  
  
  
  
  
{% endblock %}

# Testing Plans and Documentation

How/What AMZSPY will be tested:

1. Rapid entry form submission.
2. Human error and malicious intent via user input.
3. Checking on back-end database processes via Firebase to ensure continually updated price values and correctly submitted user entries.
4. Outputting required values via console to verify working scraper.
5. Implementing simple calculations and price checking formulas to verify working conditional statements.
6. Connecting website to database and code to ensure fully automated process not requiring a GUI open and running.

# Diagrams

A close up of a map

Description automatically generated

# A screenshot of a cell phone Description automatically generated

A close up of a map

Description automatically generated

# FAQs and Support

**Contact for Any Questions or Concerns:** [**andy@andysok.com**](mailto:andy@andysok.com)

**Q: Is there any protection against malicious user intent on the submit form?**

A: There is currently not any exception handling or human error precautions.

**Q: Is Amazon the only supported store?**

A: Yes. This software can only search for Amazon products using URL’s.

**Q: Will you spam the emails with price drops?**

A: No. There will be one email sent per price drop to avoid any SPAM issues or concerns. There can be a continual price drop notification dependent on the reuse purposes.

**Q: Are short links supported?**

A: As long as there is a redirection to an Amazon product URL, any links will be supported.

**Q: Are any emails supported?**

A: All email formats are supported to avoid any issues concerning SPAM.

**Q: Is this program free to use?**

A: 100% free forever for all users. We are able to continue operation via Amazon partnership in their referral program, which allows us to be compensated with a very small commission at no extra cost or hassle to you!

**Q: How do I use this software?**

A: Simply copy and paste an Amazon URL and enter your desired email to receive a notification for a price drop!

**Q: How can I contact you for any issues?**

A: Please feel free to contact us at [andy@andysok.com](mailto:andy@andysok.com)

**Q: This product isn’t working? What is wrong?**

A: The only reasons why you may not be receiving a price drop are due to a few simple reasons:

1. The email entered is not corrected due to misspelling or URL entered is incorrect due to misspelling or non-Amazon URL.
2. The notification is in your Junk mail.
3. The product does not have any price drops.

# Instructions On How To Use AMZSPY:

1. Enter the desired Amazon Product URL via copy & paste.
2. Enter the desired email for the price drop notification.
3. Simply wait for an email in your inbox!

# Licensing Information: There will be no licensing. This product will remain free to view and use.

Known issues as of February 13, 2020:

1. There is an error for any incorrect URL’s.
2. There could be an access block on Amazon’s behalf in which a proxy is implemented, or VPN is activated to fix the issue.
3. The Amazon Referral program is purposefully not installed into the email due to low traffic in development.
4. Website is currently static and cannot support any buttons or submit forms.
5. Database is not fully connected and will not update or function properly.
6. The Amazon price drop calculation is not fully implemented and is still in the testing phase. The price check will be based off of a discount entry.

# Final Conclusion and Reporting

So, after intensive testing and remodeling of the program, there has been a switch in implementation of the website and database. The website is now created and running on a Django development webserver with a NGROK public webserver redirect link to the local host. The database is a DBSQLLITE3 SQL database provided via Django and only stores the User’s Email and Amazon URL link as a model for the website, and uses the submit form to call the python script to enable price tracking. With this slight change, the product has significantly gotten a better UX/UI design being able to implement GetBootstrap’s template code for CSS and HTML. As a final report, this does what it sets out to do and the only feature purposefully not implemented is the Amazon Referral Program, despite the website mentioning compensation and commission.