## 1. وارد کردن کتابخانه‌ها و واریابل‌های پیکربندی

```python

import locale

from selenium import webdriver

from flask import Flask, jsonify, request

import time

import re

import json

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.chrome.options import Options

from selenium.webdriver.common.by import By

from datetime import datetime

```

- \*\*locale\*\*: برای فرمت‌بندی اعداد و تاریخ‌ها در قالب‌های محلی

- \*\*webdriver\*\*: برای کنترل مرورگر وب

- \*\*Flask\*\*: برای ایجاد وب‌سرور و API

- \*\*time\*\*: برای استفاده از توقف‌ها (delay‌ها)

- \*\*re\*\*: برای استفاده از عبارات باقاعده جهت پردازش رشته‌ها

- \*\*json\*\*: برای کار با فایل‌ها و داده‌های JSON

- \*\*Options\*\*: برای پیکربندی مرورگر Chrome

- \*\*By\*\*: برای استفاده از استراتژی‌های جستجو در Selenium

- \*\*datetime\*\*: برای کار با تاریخ و زمان

## 2. تعریف توابع

### 2.1. `parse\_posted\_date`

```python

def parse\_posted\_date(posted\_text):

if 'دیروز' in posted\_text:

return 1

elif 'امروز' in posted\_text:

return 2

elif 'ساعت' in posted\_text:

return 3

elif 'روز پیش ۳' in posted\_text:

return 4

elif 'پریروز' in posted\_text:

return 5

else:

return None

```

این تابع تاریخ ارسال آگهی را مقایسه می‌کند و یک کد عددی بازمی‌گرداند که به ترتیب نشان‌دهنده زمان ارسال آگهی است.

### 2.2. `get\_driver`

```python

chrome\_options = Options()

chrome\_options.add\_argument('--headless')

chrome\_options.add\_argument('--user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36')

chrome\_options.add\_argument('--disable-gpu')

chrome\_options.add\_argument('--no-sandbox')

driver = None

def get\_driver():

global driver

if driver is None:

driver = webdriver.Chrome(options=chrome\_options)

return driver

```

این تابع یک نمونه از مرورگر Chrome را ایجاد می‌کند و اگر یک نمونه از آن وجود نداشته باشد، یک نمونه جدید ایجاد می‌کند.

### 2.3. `car\_Options`

```python

@app.route('/edit/<lost\_color>/<car\_assident>', methods=['POST'])

def car\_Options(lost\_color, car\_assident):

try:

with open('Lost\_Price.json') as file:

prices\_lost = json.load(file)

prices\_lost["Color\_Lost"] = lost\_color

prices\_lost["Car\_assident"] = car\_assident

with open('Lost\_Price.json', 'w') as file:

json.dump(prices\_lost, file, indent=4)

time.sleep(3)

with open('Lost\_Price.json', 'r') as file:

prices\_l = json.load(file)

# Parse prices from string to float

locale.setlocale(locale.LC\_ALL, '')

color\_l = locale.atof(prices\_l['Color\_Lost'])

assident\_l = locale.atof(prices\_l['Car\_assident'])

# Calculate final price

Final\_price = color\_l + assident\_l

prices\_l["Final\_price"] = int(Final\_price)

with open('Lost\_Price.json', 'w') as file:

json.dump(prices\_l, file, indent=4)

time.sleep(5)

result = {"message": "Your operation was successful."}

return jsonify(result)

except Exception as e:

print("error in write or read data json file: ", e)

```

این تابع اطلاعات را از یک فایل JSON می‌خواند، آنها را به روزرسانی می‌کند و سپس اطلاعات به روزرسانی شده را در فایل JSON ذخیره می‌کند.

### 2.4. `main`

```python

@app.route('/<city>/<name\_product>', methods=['POST'])

def main(city, name\_product):

try:

# Get driver instance

driver = get\_driver()

driver.get("https://divar.ir/s/" + city)

# Search for the product

search\_input = driver.find\_element(By.CSS\_SELECTOR, ".kt-nav-text-field\_\_input")

search\_input.send\_keys(name\_product)

search\_input.send\_keys(Keys.RETURN)

# Scroll and collect ads

start\_time = time.time()

all\_ads = []

while (time.time() - start\_time) < 30:

driver.execute\_script("window.scrollBy(0, 200);")

time.sleep(0.5)

ads = driver.find\_elements(By.CSS\_SELECTOR, ".post-list\_\_widget-col-c1444 article.kt-post-card")

for ad in ads:

ad\_data = {}

# Collect ad data

ad\_data['title'] = ad.find\_element(By.CSS\_SELECTOR, ".kt-post-card .kt-post-card\_\_title").text

ad\_data['description'] = ad.find\_element(By.CSS\_SELECTOR, ".kt-post-card .kt-post-card\_\_description:nth-of-type(1)").text

ad\_data['price'] = ad.find\_element(By.CSS\_SELECTOR, ".kt-post-card .kt-post-card\_\_description:nth-of-type(2)").text

try:

IMG = ad.find\_element(By.CSS\_SELECTOR, ".kt-post-card .kt-post-card-thumbnail img")

ad\_data['img'] = IMG.get\_attribute('src')

```python

except:

ad\_data['img'] = 'Null'

Link = ad.find\_element(By.XPATH, ".//ancestor-or-self::a")

ad\_data['link'] = Link.get\_attribute('href')

posted\_element = ad.find\_element(By.CSS\_SELECTOR, ".kt-post-card .kt-post-card\_\_bottom-description")

# Parse and categorize posted date

ad\_data['posted'] = parse\_posted\_date(posted\_element.text)

if ad\_data['posted'] is not None and ad\_data['posted'] <= 5:

ad\_data['original\_posted'] = posted\_element.text

driver.implicitly\_wait(10)

all\_ads.append(ad\_data)

print(ad\_data)

# Set locale for Persian numbers and currency

locale.setlocale(locale.LC\_ALL, 'fa\_IR')

# Extract prices from ads

all\_prices = [int(re.search(r'(\d[\d,]\*)', ad['price']).group(1).replace(',', '')) for ad in all\_ads]

# Calculate average price

average\_price = sum(all\_prices) / len(all\_prices)

average\_price\_iranian = locale.currency(average\_price, grouping=True)

for ad in all\_ads:

js\_d = json.dumps(ad, ensure\_ascii=False)

# Format the average price

av\_price = f"میانگین قیمت تمام آگهی‌ها: {average\_price\_iranian}"

final\_res = json.dumps(av\_price, ensure\_ascii=False)

# Read the final price from a JSON file

with open('Lost\_Price.json', 'r') as file:

final\_price\_min = json.load(file)

# Calculate the real value of the product

real\_val = average\_price - final\_price\_min['Final\_price']

realsval = locale.format\_string("%.2f", real\_val, grouping=True)

x = f"ارزش واقعی محصول شما : {realsval} ريال "

fin = json.dumps(x, ensure\_ascii=False)

return jsonify(final\_res=final\_res, fin=fin, js\_d=all\_ads[:])

except Exception as e:

driver.quit()

return jsonify({"message": "\nScrapper stopped, launching again in 4 seconds..."})

time.sleep(4)

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

```

### 2.5. `main` (ادامه)

این تابع داده‌های آگهی‌های جمع‌آوری شده را پردازش می‌کند. ابتدا به صفحه مورد نظر می‌رود، سپس جستجوی محصول مورد نظر را انجام می‌دهد و آگهی‌های مرتبط را جمع‌آوری می‌کند.

پس از جمع‌آوری داده‌های آگهی‌ها، قیمت‌ها را استخراج می‌کند و میانگین قیمت‌ها را محاسبه می‌کند. همچنین اطلاعات درباره‌ی زمان ارسال آگهی‌ها را بررسی کرده و آگهی‌های قدیمی‌تر از 5 روز را حذف می‌کند.

در نهایت، قیمت‌ها و محاسبات را در فرمت مناسب برای نمایش به کاربر آماده می‌کند و به عنوان پاسخ به درخواست‌های وب‌سرویس خروجی می‌دهد.

## 3. اجرای برنامه

```python

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

```

این بخش اجازه اجرای برنامه را فراهم می‌کند.