Amazon Scraper(Selenium, BeautifulSoup)

1. The Scraper starts through looping over platform codes in a f-string based url and then it uses selenium webdriver to locate the pincode box.

for code in code\_df['platform\_code']:  
 url = f"https://www.amazon.in/dp/{code}"

1. Then it enters the pincode fetched from the excel sheet and loads the driver.

location.send\_keys(pin, Keys.ENTER)

1. It checks whether the product is available or not.
2. If the product is unavailable then, certain data points are assigned as "Unavailable", because they aren't located in the DOM.
3. If the product is available, then it is loaded into a BeautifulSoup object to fetch all elements.
4. For delivery days, the site displays date in format(Wednesday, 30 October). So it sorts by checking through an 'if else conditions'. This is to solve the delivery days exception of different years of delivery and order date.

date\_object = datetime.strptime(del\_date, "%A, %d %B")  
if date\_object.month == 1 and Present\_date.month != 1:  
 d\_obj = date\_object.replace(year=(datetime.today().year + 1))  
else:  
 d\_obj = date\_object.replace(year=datetime.today().year)  
days = (d\_obj.date() - Present\_date).days

1. At the end of the script, all data points are loaded as a dictionary and then appended into a list('data\_list').

BigBasket Scraper (Requests, BeautifulSoup, BigBasket API)

1. The Scraper starts with looping over keywords with page = 0 and rank = 1 and end at page 2.

slugs = ['kelloggs', 'chips']  
data\_list = []  
for slug in slugs:  
 page = 0  
 rank = 1

1. It mimicked a f-string API with query params i.e., slug & page to hit & retrieve nicely formatted JSON string.

page += 1  
url = f"https://www.bigbasket.com/listing-svc/v2/products?type=ps&slug={slug}&page={page}"

1. It then, increments the page value and runs the requests object with the API.
2. Then, it loops through all products list(var : products) and assigning specific key's values to variable append in a dictionary.

Purple.com Scraper(Requests, BeautifulSoup)

1. It starts with removing duplicate urls and pincodes.
2. Then it extracts the ‘script’ element to store the basic data points the product like sku and product\_id etc.
3. Then it loops through all location pincodes and uses the sku as product\_id in different APIs to fetch pincode availability, pricing and reviews&ratings data etc.

for pin in locat\_df['location']:

pincode\_api = f"https://www.purplle.com/neo/cart/pincode-check?pincode={pin}&productid={sku}"  
 price\_api = f"https://www.purplle.com/neo/catalog/retrieveprice?productId={sku}&pincode={pin}"  
 review\_api = f"https://www.purplle.com/neo/catalog/reviews?productId={sku}&page=1&sortBy=mh"

1. It adds headers for pincode requests object to not cause “Invalid API” error.
2. It then adds 2 lambda function to check the status of “In-Stock” or not.

Other for checking delivery days. If it has no ‘delivery days’ key then it displays the

‘message’ key.

1. It then collects individual response objects and extracts data points through data manipulation and append all into a dataframe.