Practical Python: Real-World Applications of Fundamental Concepts

Introduction

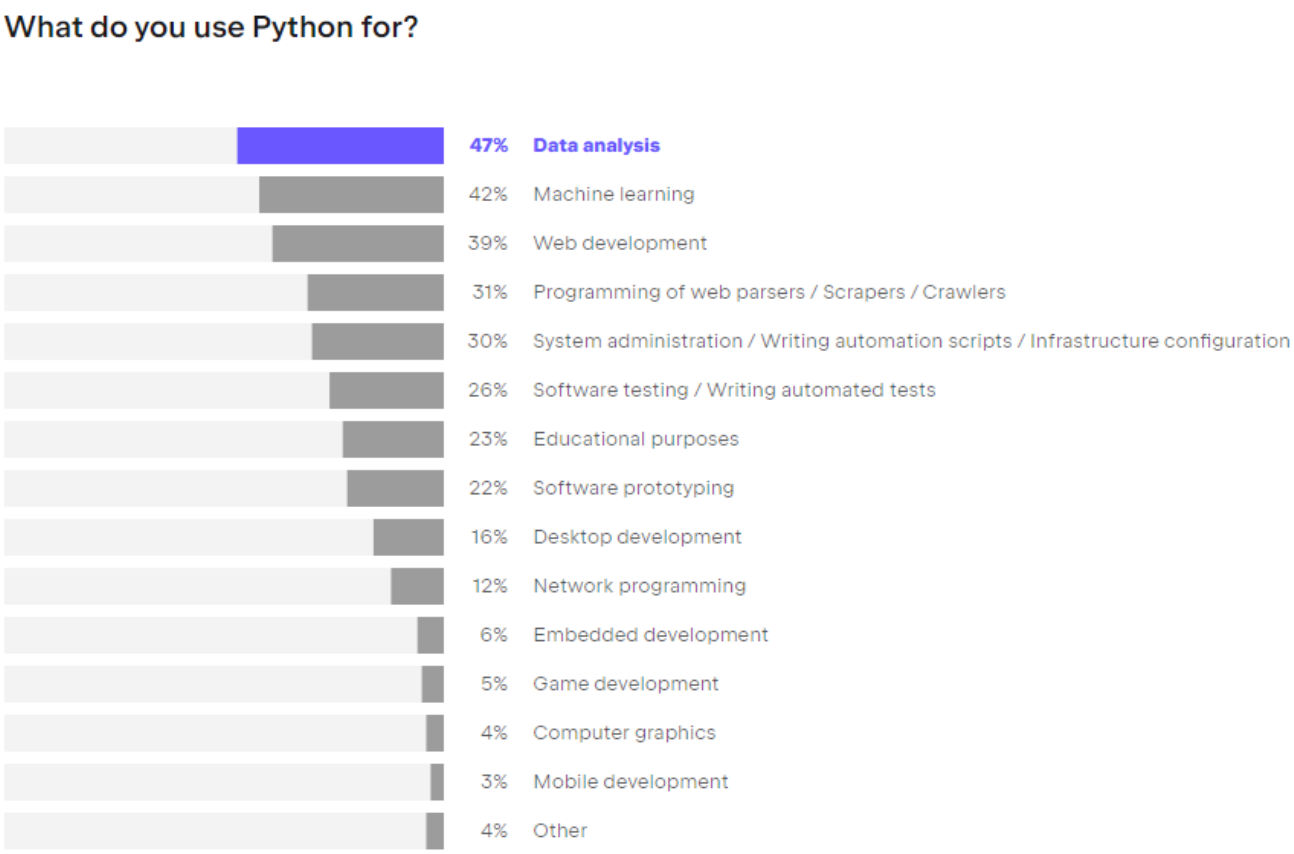
Learning Python is a long and arduous process. Throughout the process of learning we often felt lost on how to apply the Python we have learned to the real-world cases or better yet, to our jobs. But let's not get ahead of ourselves, we have just finished learning the Python Fundamentals, which could be done in as short as a single week. How can we test our skills in Python after just learning the Fundamentals? Doing so, the most approachable way, or sometimes even the only way to test and sharpen our Python technique is through a personal project.

Python fundamentals refer to the basic concepts and features of the Python programming language that are essential for understanding and writing Python code. In a short time of learning, new learners could understand these Fundamentals which in this article only includes the 8 most basic Python Fundamental:

1. Python Syntax and Semantics,
2. Data Types and Variables,
3. Operators,
4. Control Structures (conditional & loops),
5. Data Structures (List, Tuple, Dictionary, Sets),
6. Functions,
7. Module (most commonly used modules like' math', 'random', 'os', 'sys')
8. File Handling (reading & writing files like .csv)

There were a lot of Fundamental Python project available online, hundreds if not thousands of them but doing these projects often left the learners feels empty and not satisfied. The reason is because most of these projects have no usefulness to the user's actual life and no real use cases. Another "To Do List", "Calculator", "Guess the Number", "Word Count", "Hangman", "Password Generator", "Unit Converter" etc is all useless for actual use. Even worse by following this project pathway a lot of people were stuck in the dreaded "Tutorial Hell".

This is why, though contradicting, experienced python user would suggest new learner to just dive straight to their own personal passion project. Go straight to build your dream game, or mobile App. Even though these projects is almost impossible to complete with just the Python Fundamentals, just take whatever small steps you can take through your skill and push your way to that passion project little by little and learns the tools and technique needed along the way. This way, not only would new learners be more enthusiastic in doing their project, they would learn a lot more, and those lessons can all be applied straight in the real-world use case.



Source: JetBrains

The real-world use of Python by experienced and senior developer can be categorized into a small group of different use case. JetBrains yearly “Python Developer Survey Results” shows in what is the main use cases of Python in the real-world in 2023. In all of these real-world use cases of Python, almost all of them is unable to be completed with just the Python Fundamentals. But the surprising things

is that with the exception of ‘Educational Purposes’, very simple ‘Data Analysis’ and ‘Programming of web parsers / Scrapers / Crawlers’ are the few if not the only two that could be done with just the Python Fundamentals.

In this article, I will show you how with just the Python Fundamentals, I completed my personal projects which is Web Scrapping, easily and only in a short amount of time. This project is personally useful for me and I use this project to make a purchase decision that will stays with me for years to come. The project I’m talking about is Web Scrapping all the smartphones available and rate their Price-to-Performance ratio to find which is the best Phone Objectively according to our metrics, the best “bang for the buck” if you will.

Case: Finding the Best Phone Through Their Price-to-Performance Ratio

Nowadays we were showered with choices, there were so many phones that fits our needs and checks all the boxes for our personal use, so much so, that it is actually giving me a decision paralysis. Overchoice, also known as choice overload, is a paradox where having a wide array of options can negatively impact decision-making. This phenomenon arises when there are numerous equivalent choices to consider. The decision-making process becomes daunting because of the numerous possible outcomes and the fear of choosing incorrectly. Evaluating many similarly good options can be mentally exhausting, as each must be compared to the others to determine the best choice.

Tech Reviewers and Tech Websites are not helping either. Every single review is always ended by “it depends on the consumer’s needs” which technically correct, but if that was said to all the product reviewed it’ll make it even harder to choose. Those ranking like “Top 10 best smartphone you can buy for under $500” are not losing in making it harder to choose by their vagueness of the ranking benchmark. the metrics that they use in ranking the phones were not the same as my personal metrics. not to mention the price difference between the country. There were so many constraints and I haven’t found anybody making the price-to-performance ration of every phone yet, so I decided to make my own.

There is only one metrics that I will use, but this helps in data uniformity and give a measured constant. The metrics I'm using is the Antutu Benchmark of the phones Chipset, that is all. The camera, screen, body, size, and brand doesn't matter by 'my needs' since I rarely take a picture, always use the phone case protection, and Phone's screen, even the cheap ones is already more than good enough for everyday use. This not only makes the sorting and ranking easier, but it also gives it a more objective approach, since there are a 'hard' and 'exact' numbers for the chipset's benchmarks.

For their price to performance, we will be using the most straightforward way to measure them, which is by their ratio of Price-to-Performance by dividing the Antutu score with the phone's price. We can represent this later by how many Antutu score do we get for every one thousand rupiah that we spent for the phone. The first constraint we face is that we will only be getting phones that is available to purchase in Indonesia so and we will be using Rupiah as a currency. After that, we have to gather every single one of the phone's price, chipset, and Antutu score. The last step is to get the Price-to-Performance ratio and sort them from largest to smallest which could be easily done either with Python Pandas or just Microsoft Excel. Since we'll only be using Python Fundamental that is listed above, we'll use a little bit of excel but I'll be using the Python Pandas as visual to maintain the viewing consistency.

Web Scrapping

To starts with, we'll be using a Python library, which in a simple term, is a group of Python 'methods' that works together, usually inside a single folder. The way to use library is the same as using other module like the 'math' module, which is importing them with the keyword *import*. We will be using the most popular and the most widely used Python library to do Web Scrapping called BeautifulSoup. It understands the structure of the web pages, which are written in HTML language. BeautifulSoup is a Python library used for parsing HTML and XML documents because it handles complexities of web page formatting and let us pull out only the

information that we're interested in, saving a lot of time and effort compared to doing it manually.

```
1   import requests
2   from bs4 import BeautifulSoup
3   import csv
4   import time
```

Other than the BeautifulSoup library, we'll also be using the requests module, csv module, and time module.

Modules

1. requests
    - Purpose: This module allows us to send HTTP request inside Python code.
    - Usage in project:

      requests.get(url) : Used to fetch the content of web pages. It retrieves the HTML content of the URLs that we are scraping, both for the phone listing pages, and individual product detail pages.

```
45
46      page = requests.get(url)
47      soup = BeautifulSoup(page.content, 'html.parser')
48
```

2. bs4 (BeautifulSoup)
    - Purpose: For parsing HTML and XML documents. It creates parse trees from page source codes that can be used to extract data easily.
    - Usage in project:
        i. soup = BeautifulSoup(page.content, 'html.parser')
        Parses the HTML content fetched by the 'requests' module.
        ii. soup.find(...), soup.select(...), soup.select_one(...)
        Used to navigate and search the HTML parse tree to find elements and extract information such as product names, prices, and CPU details.

3. csv
    - Purpose: This module implements classes to read and write tabular data in CSV format so we can use them in Excel.
    - Usage in project:

        csv.DictWriter(...)

        Used to write the scrapped phone data into a CSV file. This allows us to save the collected data in a structured format that can be easily analysed or used later.

```
87  # Write to CSV
88  if all_products:
89      keys = all_products[0].keys()
90      with open('all_products_with_cpu.csv', 'w', newline='', encoding='utf-8') as output_file:
91          dict_writer = csv.DictWriter(output_file, keys)
92          dict_writer.writeheader()
93          dict_writer.writerows(all_products)
94      print("CSV file created successfully.")
95  else:
96      print("No products found. The scraping might have failed.")
```

4. time
    - Purpose: This module provides various time-related functions.
    - Usage in project:

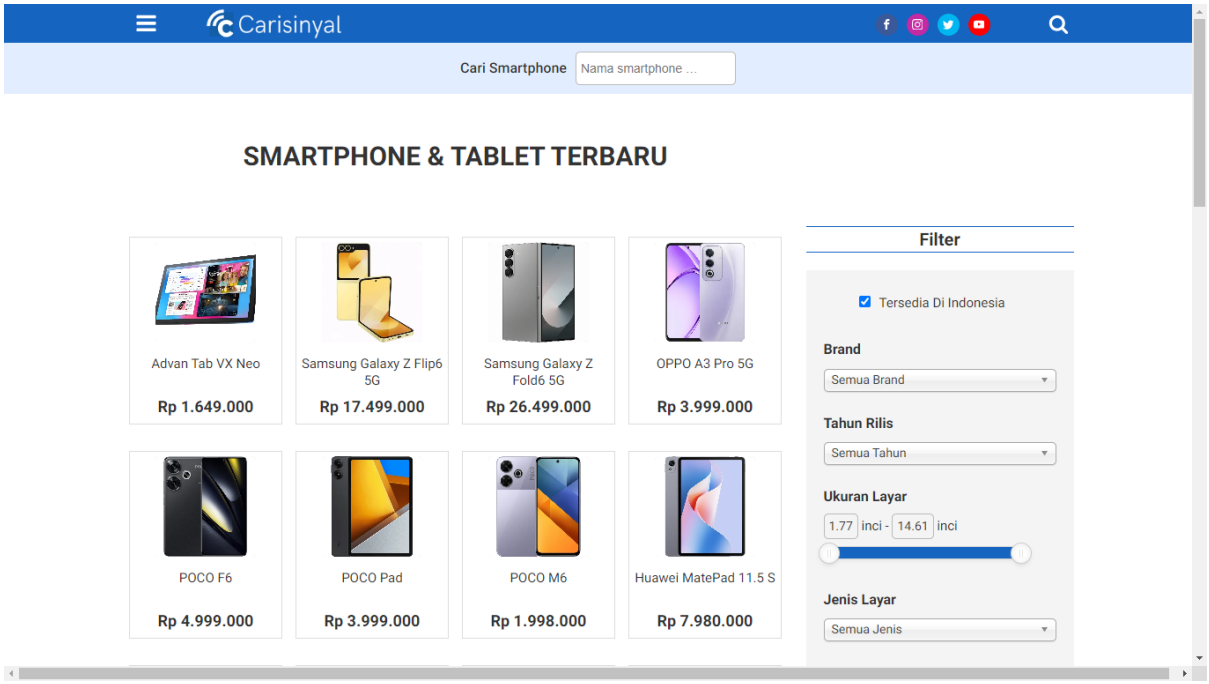
        time.sleep(*seconds*)

        Used to add delays in the scrapping process. This is *important* to avoid overwhelming the server with too many requests in a short period, which can lead to being blocked or banned from the side. Delays are added between individual phone detail requests and between page requests.

```
76
77                  # Add a delay to be respectful to the server
78                  time.sleep(2)
79              else:
80                  print(f"Couldn't find name for a product")
81
82          # Add a delay between pages
83          time.sleep(5)
84
```

Main Code

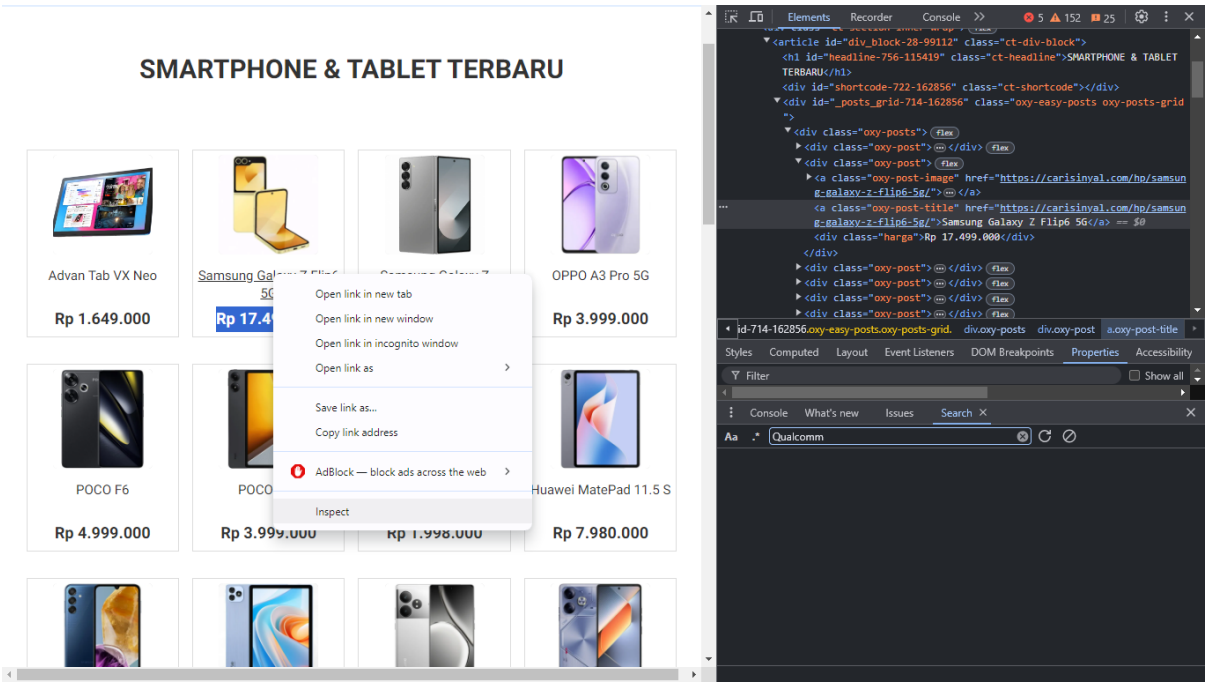
Now with the first four code, importing the module done we can start the main code. After importing the module, we go straight into deciding which webpages we are going to scrape. For the project I found the website that fits my requirements, not only it has every current phones available in Indonesia, it also include the price for their lowest memory options giving every phone the same baseline. the URL is https://carisinyal.com/

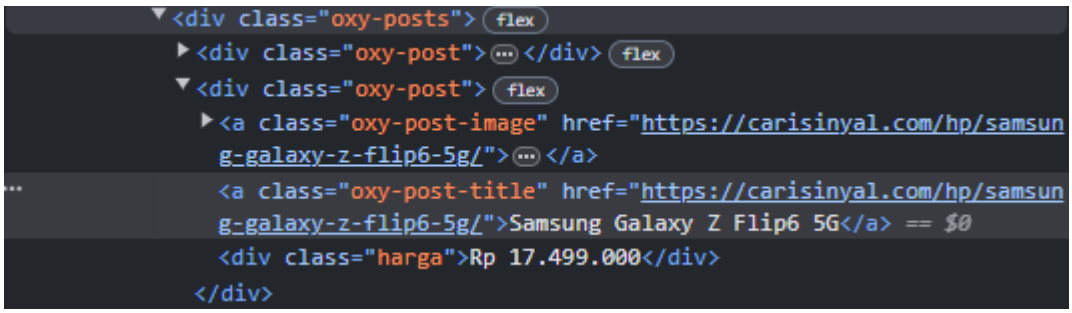


We can go straight to writing our code. import the 4 module, make a variable containing the main URL of the website we want to scrape and use BeautifulSoup to parse through the website HTML format. we then make another variable ‘products’ containing each of the phone on the webpage.

```
1  import requests
2  from bs4 import BeautifulSoup
3  import csv
4  import time
5
6  base_url = "https://carisinyal.com/hp/?_sfm_harga_status=Available&_sfm_layar_
7
8  soup = BeautifulSoup(base_url.content, 'html.parser')
9  products = soup.select('div.oxy-post')  # Adjust this selector if needed
```

To get each of the phone on the webpage, we use the soup.select() method so the BeautifulSoup scan through all the page information and get the information that we want. how do we know what to search from the webpage? That can easily be done by using the right click and 'inspect' the section we want to scrape





Here we can see that the Phone's names and the Phone's prices information on the webpage HTML format is inside the <a class ="oxy-post-title" href…> and the post-title is inside the div class "oxy-posts". The "oxy-posts" acts as a full container of each phones complete with their subclass of name, price and links that is why we get the whole container with:

products = soup.select('div.oxy-post)

```python
import requests
from bs4 import BeautifulSoup
import csv
import time

base_url = "https://carisinyal.com/hp/?_sfm_harga_status=Available&_sfm_layar_ukuran_laya

soup = BeautifulSoup(base_url.content, 'html.parser')
# Find all product containers
products = soup.select('div.oxy-post')  # Adjust this selector if needed
print(f"Found {len(products)} product containers on page {page_number}")

for product in products:
    name_elem = product.select_one('a.oxy-post-title')
    if name_elem:
        name = name_elem.text.strip()
        link = name_elem['href']
        price_elem = product.select_one('div.harga')
        price = price_elem.text.strip() if price_elem else 'N/A'
```

From the soup.select() BeautifulSoup gets the whole 20 occurrence of such line because there are 20 Phones in a single page from carisinyal.com. to get the name of each phone with their price and links to each product page, because there is no information about the chipset of each phones on this webpage. To get that information we have to open the individual phone pages one by one to get their chipset. the print function's purpose is just to inform us that the scrapping is working.

We iterate for each product, from the previous HTML inspect we found that the phone name is under the subclass "oxy-post-title". from that section, we get both the product name, which is the only text on the class, and the link so we can open each phone page separately. the price information is inside the same product, but different subclass which we have to write, it's under the "div.harga", the else N/A is for error handling so that the program doesn't terminate if it can't find the price.

After doing all that, the next step is the looping of each product, we visited each of the 20 products individual pages to get the chipset of the phones. we can put a function calling inside the product iteration so that in every product iteration the function gets called and we go straight inside to their individual pages, don't forget the error handling.

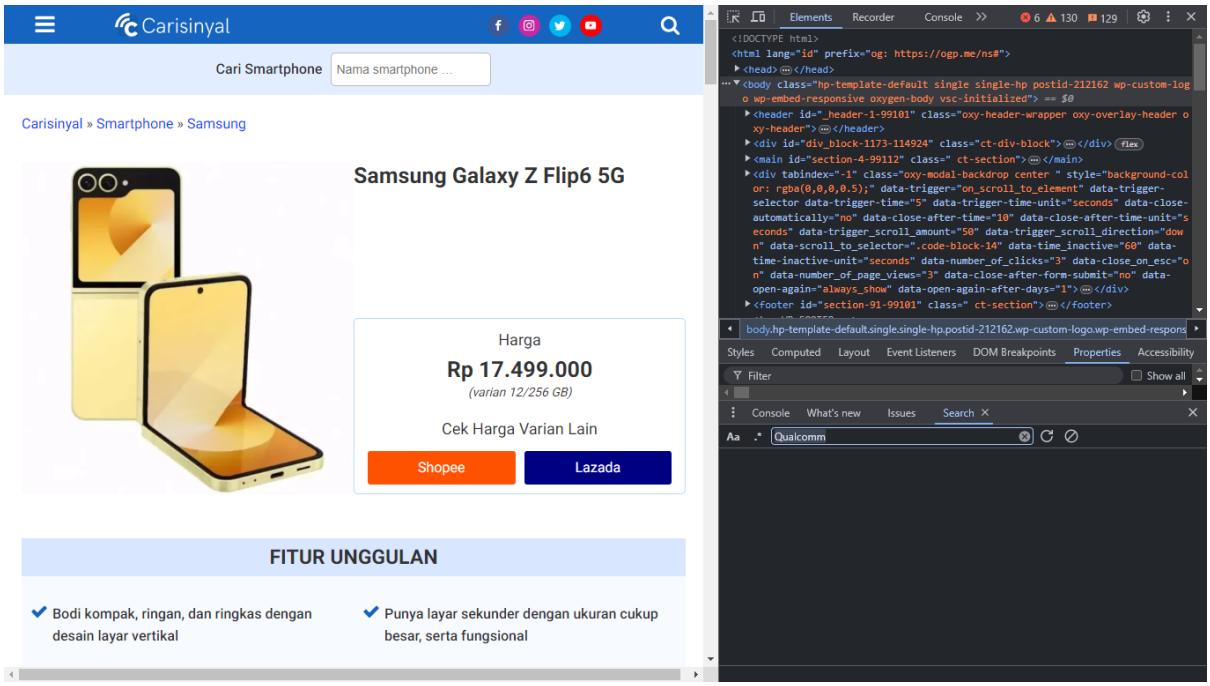
```
    for product in products:
        name_elem = product.select_one('a.oxy-post-title')
        if name_elem:
            name = name_elem.text.strip()
            link = name_elem['href']
            price_elem = product.select_one('div.harga')
            price = price_elem.text.strip() if price_elem else 'N/A'

        try:
            # Get CPU info from the product page
            cpu = get_product_details(link)
        except Exception as e:
            print(f"Error fetching details for {name}: {str(e)}")
            cpu = 'Error'
```

Now we'll construct the get_product_detail(link) function. we have to put the function above the code we are currently writing so that the function is constructed before we call them. the link we previously get will be the parameter and we put them as arguments for the get_product_detail(link) function.

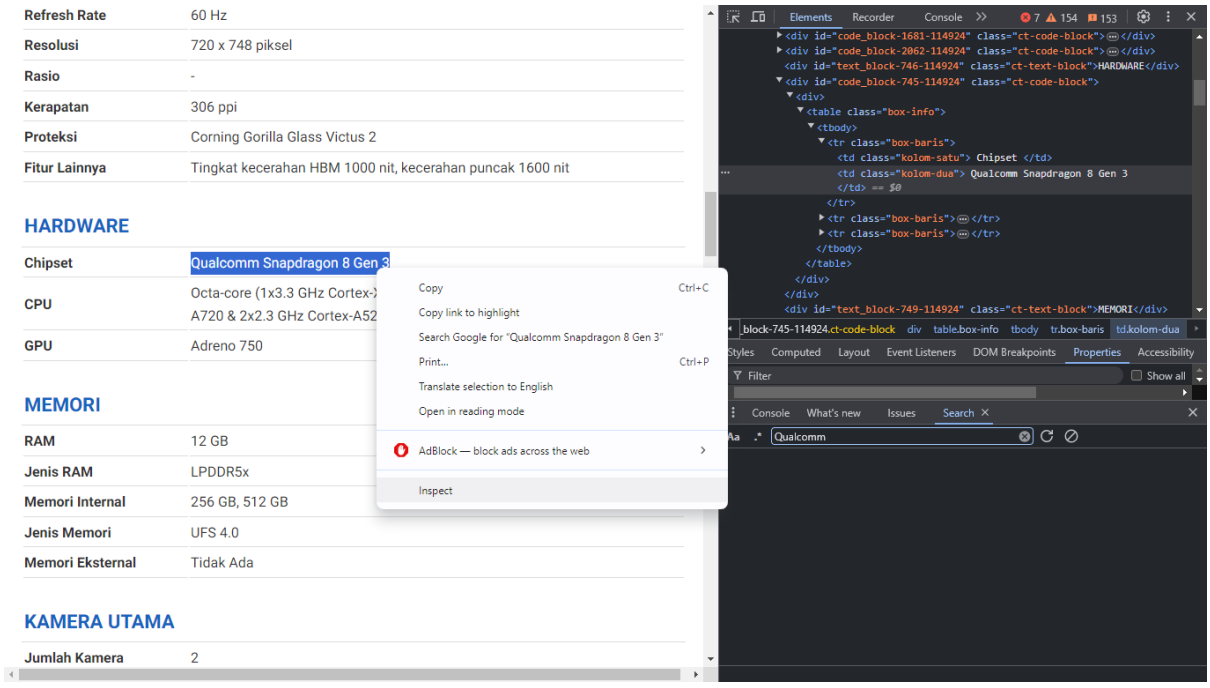
```
import time

def get_product_details(url):
    print(f"Fetching details from: {url}")
    page = requests.get(url)
    soup = BeautifulSoup(page.content, 'html.parser')

    cpu = 'None'
    hardware_section = soup.find('div', id='code_block-745-114924')

    if hardware_section:
        rows = hardware_section.find_all('tr')
        for row in rows:
            cols = row.find_all('td')
            if len(cols) == 2:
                key = cols[0].text.strip().lower()
                value = cols[1].text.strip()
                if 'chipset' in key:
                    cpu = value
                    print(f"Found CPU: {cpu}")
                    break

    if cpu == 'None':
        print("CPU information not found")

    return cpu

base_url = "https://carisinyal.com/hp/?_sfm_harga_status=Available&_sfm_layar_ukuran_layar=1.77+14.61&_sfm_bate
```

First we get notified that the function is called successfully, we repeat the first process of using the requests module so the program get the URLs HTML, then we put the HTML content inside the soup with BeautifulSoup module just like previously.

Now we are inside each of the product pages. the soup variable contains all the pages information.



We have to find the information that we want to scrape and inspect them.



We use soup.find() method this time which works the same way as the .select() method just for different format. The select() is for CSS selector and return all matching text, while tag() is for HTML tag and returns only the first finding. If there is such tag that we are looking for, we split the whole product page into each

section. In HTML <tr> and <td> is tag used within tables to define rows and cells respectively. This is HTML format, after first getting the container tag, we split them into their own respective table row or “tr” and inside the table row is the table data which is the “td”. From the inspection we find that inside the container, in their own table rows, the first line is for the “Chipset” text, and the second line is the “Snapdragon 8 Gen 3” that we wanted to scrape. We write the if conditional inside the iterator, after iterating each table rows, if the program found the table data(td) with “Chipset” text inside, we are scraping the text in the line right after it.

```
15          rows = hardware_section.find_all('tr')
16          for row in rows:
17              cols = row.find_all('td')
18              if len(cols) == 2:
19                  key = cols[0].text.strip().lower()
20                  value = cols[1].text.strip()
21                  if 'chipset' in key:
22                      cpu = value
23                      print(f"Found CPU: {cpu}")
24                      break
```

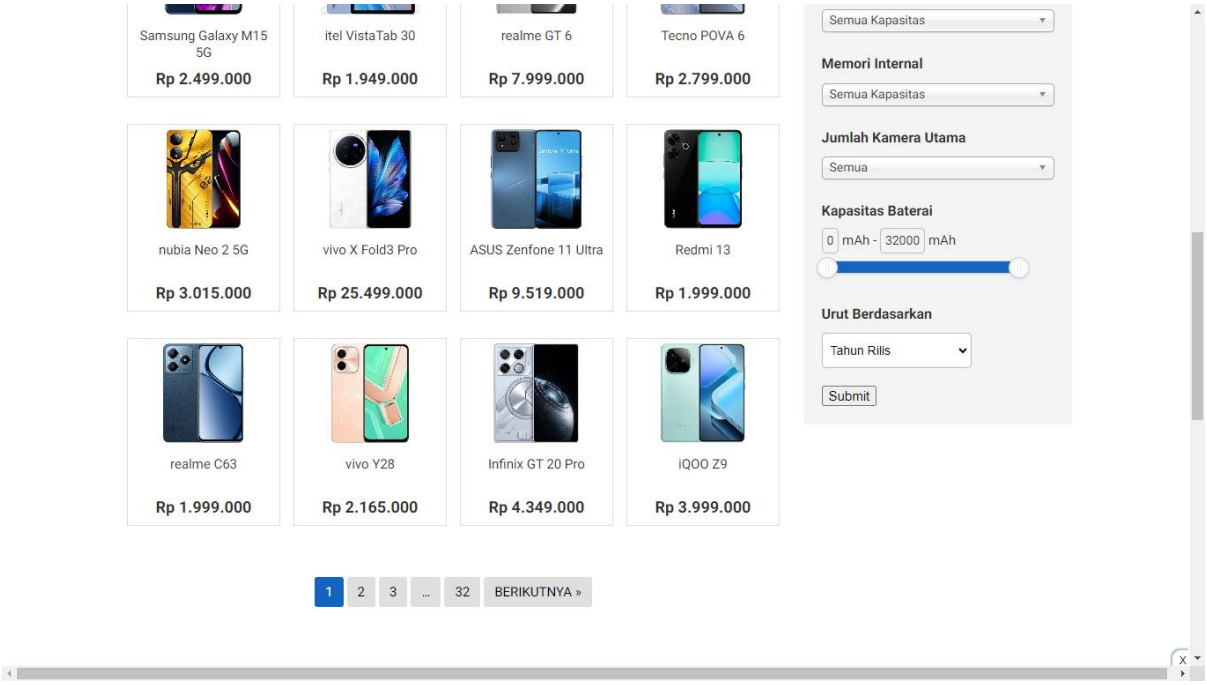
We set the scrapped information inside the cpu variable and break out of the loop. then we use a little bit of error handling in case there’s no cpu information, and return the cpu variable if we found one.

```
26      if cpu == 'None':
27          print("CPU information not found")
28
29      return cpu
```

Now we’re going back to the main page, the whole scraping is already done, we’ve got all the information we needed from this website, which is the Phone Name that is available to be purchased in Indonesia, it’s price and also, it’s chipset or CPU. We only need to append them to out collection data types, of which I used the list because we need a mutable datatype. First, we make an empty list in the global scope, so I put them above the product iteration code. I name the list all_products and we append all the information of a singular product we have scraped from the page inside the iterator, so in each iteration of product, we append a single phone information to the list and get notified with print statement. we also put the delay below so that for every iteration there is a delay in scraping for each product.

```python
    try:
        # Get CPU info from the product page
        cpu = get_product_details(link)
    except Exception as e:
        print(f"Error fetching details for {name}: {str(e)}")
        cpu = 'Error'

    all_products.append({
        "name": name,
        "price": price,
        "cpu": cpu,
        "link": link
    })

    print(f"Scraped: {name}")

    # Add a delay to be respectful to the server
    time.sleep(2)
```

Now we are pretty much done, we'll be scraping the whole webpage of 20 products, but this is only one of the 32 pages of phones. So now we just have to use the same code for every single page, we just need to make an iterator for each page and put the previous code inside that iterator. What we need to first find out is how we can move from a page to the next page over and over until we visited all 32 pages. Thankfully the solution is simple because each page URLs is unique only in the end, so in every iteration, we only have to change the URL slightly.



This is the URL for the second page
https://carisinyal.com/hp/?_sfm_harga_status=Available&_sfm_layar_ukuran_layar=1.77+14.61&_sfm_baterai_kapasitas_baterai=0+32000&sf_paged=2

And this is the URL for the third page
https://carisinyal.com/hp/?_sfm_harga_status=Available&_sfm_layar_ukuran_layar=1.77+14.61&_sfm_baterai_kapasitas_baterai=0+32000&sf_paged=3

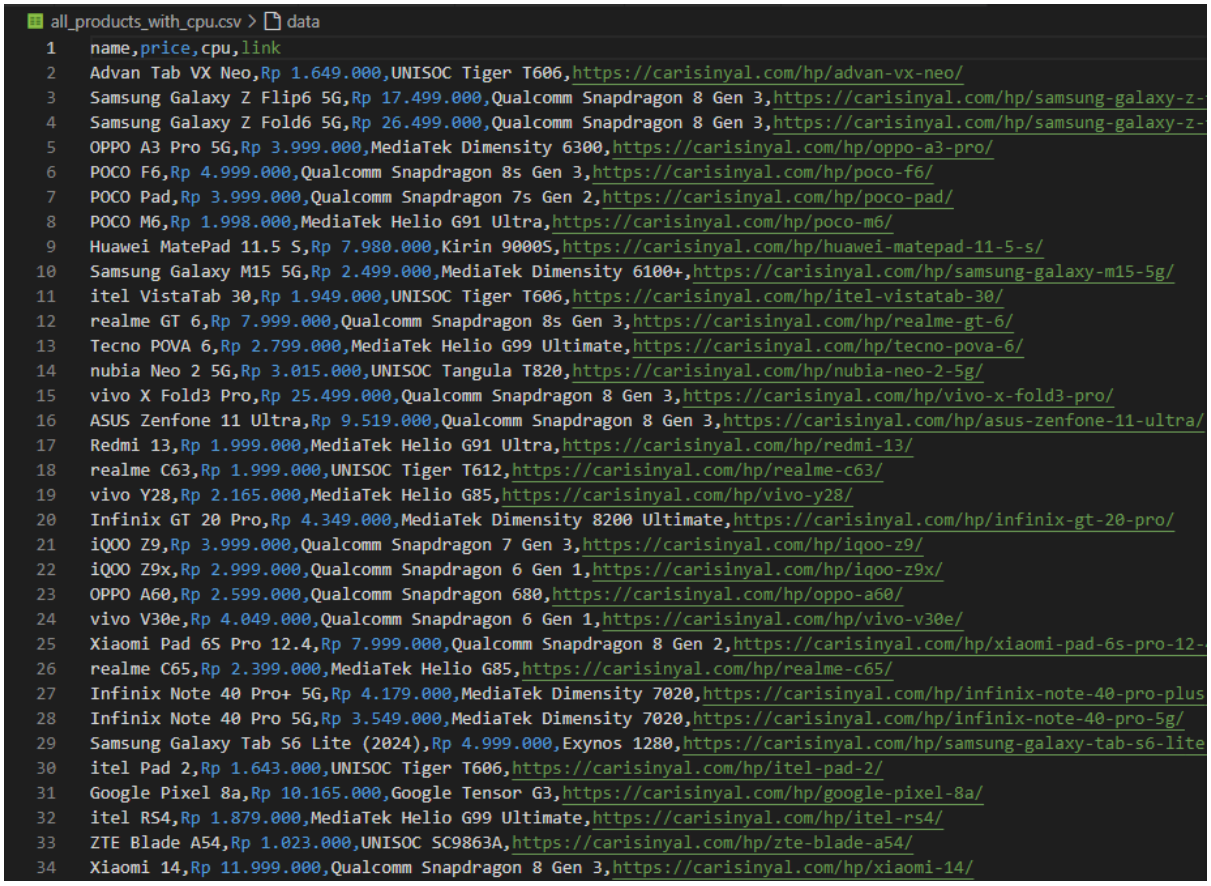
They have the same URLs with only the last digit being different, so we can easily iterate over a range of 32 and change the URLs ending according to the iteration.

```
31
32  base_url = "https://carisinyal.com/hp/?_sfm_harga_status=Available&_sfm_layar_ukuran_layar=1.77+14.61&
33  total_pages = 32
34  all_products = []
35
36  for page_number in range(1, total_pages + 1):
37      url = base_url if page_number == 1 else f"{base_url}&sf_paged={page_number}"
38      print(f"Scraping page {page_number} of {total_pages}")
39      print(f"URL: {url}")
40
41      soup = BeautifulSoup(base_url.content, 'html.parser')
42      # Find all product containers
43      products = soup.select('div.oxy-post')  # Adjust this selector if needed
44      print(f"Found {len(products)} product containers on page {page_number}")
45
46      for product in products:
```

We just put the iterator above the code and indent all the code below so the main code will run on every single page. we put the total pages inside a variable and make an iterator of page_number which will increase one until it reaches the last page. At the end of the iteration, we also put another delay with time method so that we stop for a few second in every page. Now the code is fully done, we only have to save the data from the all_product list into a CSV so we can easily analyse the data. To save into CSV, we can use the code already shown above on the module part.

```
79
80  # Write to CSV
81  if all_products:
82      keys = all_products[0].keys()
83      with open('all_products_with_cpu.csv', 'w', newline='', encoding='utf-8') as output_file:
84          dict_writer = csv.DictWriter(output_file, keys)
85          dict_writer.writeheader()
86          dict_writer.writerows(all_products)
87      print("CSV file created successfully.")
88  else:
89      print("No products found. The scraping might have failed.")
```

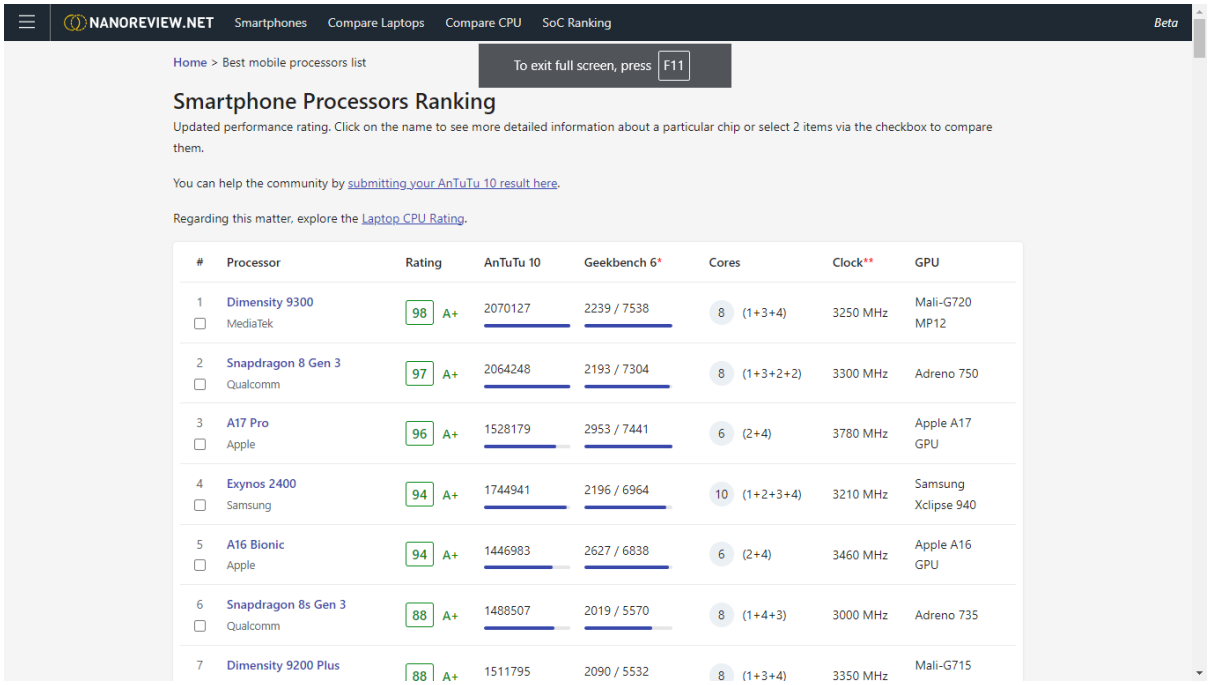
Now we have got all the phone data inside the CSV and if you open it inside a code editor, it'll look like this:


all_products_with_cpu.csv > data

```
name,price,cpu,link
Advan Tab VX Neo,Rp 1.649.000,UNISOC Tiger T606,https://carisinyal.com/hp/advan-vx-neo/
Samsung Galaxy Z Flip6 5G,Rp 17.499.000,Qualcomm Snapdragon 8 Gen 3,https://carisinyal.com/hp/samsung-galaxy-z-
Samsung Galaxy Z Fold6 5G,Rp 26.499.000,Qualcomm Snapdragon 8 Gen 3,https://carisinyal.com/hp/samsung-galaxy-z-
OPPO A3 Pro 5G,Rp 3.999.000,MediaTek Dimensity 6300,https://carisinyal.com/hp/oppo-a3-pro/
POCO F6,Rp 4.999.000,Qualcomm Snapdragon 8s Gen 3,https://carisinyal.com/hp/poco-f6/
POCO Pad,Rp 3.999.000,Qualcomm Snapdragon 7s Gen 2,https://carisinyal.com/hp/poco-pad/
POCO M6,Rp 1.998.000,MediaTek Helio G91 Ultra,https://carisinyal.com/hp/poco-m6/
Huawei MatePad 11.5 S,Rp 7.980.000,Kirin 9000S,https://carisinyal.com/hp/huawei-matepad-11-5-s/
Samsung Galaxy M15 5G,Rp 2.499.000,MediaTek Dimensity 6100+,https://carisinyal.com/hp/samsung-galaxy-m15-5g/
itel VistaTab 30,Rp 1.949.000,UNISOC Tiger T606,https://carisinyal.com/hp/itel-vistatab-30/
realme GT 6,Rp 7.999.000,Qualcomm Snapdragon 8s Gen 3,https://carisinyal.com/hp/realme-gt-6/
Tecno POVA 6,Rp 2.799.000,MediaTek Helio G99 Ultimate,https://carisinyal.com/hp/tecno-pova-6/
nubia Neo 2 5G,Rp 3.015.000,UNISOC Tangula T820,https://carisinyal.com/hp/nubia-neo-2-5g/
vivo X Fold3 Pro,Rp 25.499.000,Qualcomm Snapdragon 8 Gen 3,https://carisinyal.com/hp/vivo-x-fold3-pro/
ASUS Zenfone 11 Ultra,Rp 9.519.000,Qualcomm Snapdragon 8 Gen 3,https://carisinyal.com/hp/asus-zenfone-11-ultra/
Redmi 13,Rp 1.999.000,MediaTek Helio G91 Ultra,https://carisinyal.com/hp/redmi-13/
realme C63,Rp 1.999.000,UNISOC Tiger T612,https://carisinyal.com/hp/realme-c63/
vivo Y28,Rp 2.165.000,MediaTek Helio G85,https://carisinyal.com/hp/vivo-y28/
Infinix GT 20 Pro,Rp 4.349.000,MediaTek Dimensity 8200 Ultimate,https://carisinyal.com/hp/infinix-gt-20-pro/
iQOO Z9,Rp 3.999.000,Qualcomm Snapdragon 7 Gen 3,https://carisinyal.com/hp/iqoo-z9/
iQOO Z9x,Rp 2.999.000,Qualcomm Snapdragon 6 Gen 1,https://carisinyal.com/hp/iqoo-z9x/
OPPO A60,Rp 2.599.000,Qualcomm Snapdragon 680,https://carisinyal.com/hp/oppo-a60/
vivo V30e,Rp 4.049.000,Qualcomm Snapdragon 6 Gen 1,https://carisinyal.com/hp/vivo-v30e/
Xiaomi Pad 6S Pro 12.4,Rp 7.999.000,Qualcomm Snapdragon 8 Gen 2,https://carisinyal.com/hp/xiaomi-pad-6s-pro-12-
realme C65,Rp 2.399.000,MediaTek Helio G85,https://carisinyal.com/hp/realme-c65/
Infinix Note 40 Pro+ 5G,Rp 4.179.000,MediaTek Dimensity 7020,https://carisinyal.com/hp/infinix-note-40-pro-plus
Infinix Note 40 Pro 5G,Rp 3.549.000,MediaTek Dimensity 7020,https://carisinyal.com/hp/infinix-note-40-pro-5g/
Samsung Galaxy Tab S6 Lite (2024),Rp 4.999.000,Exynos 1280,https://carisinyal.com/hp/samsung-galaxy-tab-s6-lite
itel Pad 2,Rp 1.643.000,UNISOC Tiger T606,https://carisinyal.com/hp/itel-pad-2/
Google Pixel 8a,Rp 10.165.000,Google Tensor G3,https://carisinyal.com/hp/google-pixel-8a/
itel RS4,Rp 1.879.000,MediaTek Helio G99 Ultimate,https://carisinyal.com/hp/itel-rs4/
ZTE Blade A54,Rp 1.023.000,UNISOC SC9863A,https://carisinyal.com/hp/zte-blade-a54/
Xiaomi 14,Rp 11.999.000,Qualcomm Snapdragon 8 Gen 3,https://carisinyal.com/hp/xiaomi-14/
```

Overall there's hundreds of different and unique Phones. now for the Antutu part, luckily such list is easily available in a lot of website, and we don't have to scrape the data at all, just simply copy and paste the data into an excel file. One such website is https://nanoreview.net/en/soc-list/rating

from the page we just have to highlight all the data and copy paste them into an open excel sheet. I'll use Python Pandas as a visual help but It'll be similar in excel.



Now we have 2 table of information, the name, price, chipset table, and the chipset, Antutu table. We just have to join them by their chipset, I'll use the pandas join method, while on excel we can use the VLOOKUP, mine is

=IFERROR(VLOOKUP(E7;$K$5:$L$179;2;FALSE);"")

You should match the cell number and the whole chipset cell. we'll get a table like this from that

	name	price	cpu	antutu
0	Samsung Galaxy Z Flip6 5G	Rp 17.499.000	Qualcomm Snapdragon 8 Gen 3	2069048
1	Samsung Galaxy Z Fold6 5G	Rp 26.499.000	Qualcomm Snapdragon 8 Gen 3	2069048
2	OPPO A3 Pro 5G	Rp 3.999.000	MediaTek Dimensity 6300	447617
3	POCO F6	Rp 4.999.000	Qualcomm Snapdragon 8s Gen 3	1489956
4	POCO Pad	Rp 3.999.000	Qualcomm Snapdragon 7s Gen 2	595319
...	...	...	...	...
461	Ulefone Armor 7	Rp 6.500.000	MediaTek Helio P90	300868
462	Huawei P30 Lite	Rp 0	Kirin 710	250165
463	Apple iPhone 11	Rp 6.249.000	Apple A13 Bionic	889911
464	Doogee S68 Pro	Rp 3.900.000	MediaTek Helio P70	233983
465	vivo Y12	Rp 0	MediaTek Helio P22	145765

Now add the Price-to-Performance table by dividing the Antutu score to the price. On pandas it'll only use the

*DataFrame[new_column] = DataFrame['antutu']*1000 / DataFrame ['price"]*

DataFrame.sort_values(column, sort)

Now we have the table

	name	price	cpu	antutu	Price-to-Performance
63	itel P55 5G	1419000	MediaTek Dimensity 6080	435520	306.920366
3	POCO F6	4999000	Qualcomm Snapdragon 8s Gen 3	1489956	298.050810
228	Sharp Aquos V6 5G	1388000	MediaTek Dimensity 700	389401	280.548271
275	POCO M5	1499000	MediaTek Helio G99	418747	279.350901
242	Infinix Hot 20 5G	1899000	MediaTek Dimensity 810	429221	226.024750
189	Redmi 12C	1175000	MediaTek Helio G85	265260	225.753191
432	vivo Y20s G	1099000	MediaTek Helio G80	247951	225.615105
84	Samsung Galaxy Tab A9 (WiFi)	1920000	MediaTek Helio G99	418747	218.097396
9	ASUS Zenfone 11 Ultra	9519000	Qualcomm Snapdragon 8 Gen 3	2069048	217.359807
276	POCO M5s	1899000	MediaTek Helio G95	409894	215.847288
289	OPPO Reno8 5G	3649900	MediaTek Dimensity 1300	786855	215.582619
118	Infinix GT 10 Pro 5G	3540000	MediaTek Dimensity 8050	754997	213.275989
319	Redmi 10C	1449900	Qualcomm Snapdragon 680	308738	212.937444
245	Tecno POVA 4	1999000	MediaTek Helio G99	418747	209.478239
11	iQOO Z9	3999000	Qualcomm Snapdragon 7 Gen 3	823249	205.863716
317	Redmi 10 5G	1899000	MediaTek Dimensity 700	389401	205.055819
126	Tecno POVA 5	2049000	MediaTek Helio G99	418747	204.366520
27	vivo Y03	1299000	MediaTek Helio G85	265260	204.203233
103	vivo Y17s	1299000	MediaTek Helio G85	265260	204.203233
41	Infinix Hot 40 Pro	2059000	MediaTek Helio G99	418747	203.373968

I personally added the *1000 code so that the Price-to-Performance column represent how many points of Antutu did the phones gives from each thousand rupiah(seribu rupiah) of price, just for easier explanation. Now the project is fully done. We use only one metrics because we're only looking at the objective measurable performance, and that is what we got. This is my personal project and I will be using this information to make an informed decision to a purchase of a new phone. For your own project, you can choose your own URLs and find the data place with inspect, and edit the code slightly. Good Luck on your future project.