**Dokumentasi Projek Web Scraping**

**Exercise FTUI 2021**

### 

### 

### 

**Oleh :**

Project Division

Exercise FTUI

Universitas Indonesia

### 

Daftar Isi Dokumentasi

[**Cara penggunaan**](#_dm7x044iito3) **3**

[Instalasi](#_fktrvmoet0q0) 3

[Cara Penggunaan](#_hnsxdzptr94t) 3

[**Source Code**](#_dy3vx64lcgyd) **5**

[Library](#_ozszkktlu4t6) 5

[Scrapping Code](#_ikhdre1l1xpf) **6**

[Variabel Global](#_pavjpsk5r1q3) 6

[Fungsi mengecek URL](#_s6na7cyxxpfi) 7

[Fungsi Mencari dengan tipe data-testid](#_nyqejzcbppl9) 7

[Fungsi Mencari dengan tipe class](#_r2h0aevmzq8s) 7

[Fungsi mendapatkan dengan link toko untuk Scrapping](#_w0868caqwx07) 8

[Fungsi untuk mendapatkan Ulasan](#_gtqx4wf8y0pd) 8

[Fungsi utama Run (berguna untuk mendapatkan data keseluruhan produk)](#_2tbipo5reenm) 8

[Fungsi Start (Menginisialisasinya mulai proses scrapping di web driver)](#_mlzg65mha6ro) 10

[Fungsi Random Generator (Mengenerate nama file random)](#_nd513erdfwsb) 11

[Fungsi change (Untuk mengganti nama file dan letak penyimpanan)](#_5elqpl6en5on) 11

[Fungsi Download()](#_58vll8idtrz8) 12

[GUI Code](#_o1u9da595xst) **12**

[**Performance Speed**](#_cc0e6e2d6px) **16**

[Tabel](#_28hyqg1gnpjm) 16

[Grafik](#_oa6f13plmxlz) 17

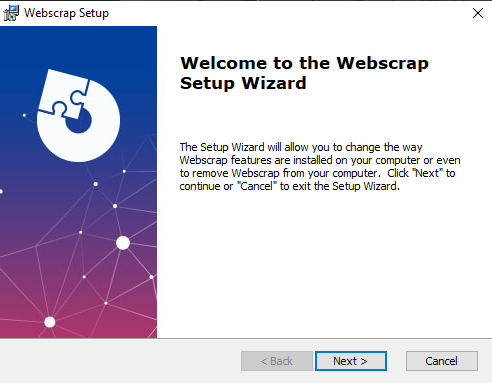
## **Cara penggunaan**

### **Instalasi**

Bentuk file yang disampaikan berupa format .zip yang perlu di extract terlebih dahulu untuk menjalankan program Webscrapping tersebut. Berikut merupakan langkah-langkah instalasinya:

Bentuk file yang disampaikan berupa format .zip yang perlu di extract terlebih dahulu untuk menjalankan program Webscrapping tersebut. Berikut merupakan langkah-langkah instalasinya:

1. Install terlebih dahulu aplikasi Webscrapping dengan menjalankan file Webscrap.msi dan ikuti proses instalasi hingga selesai



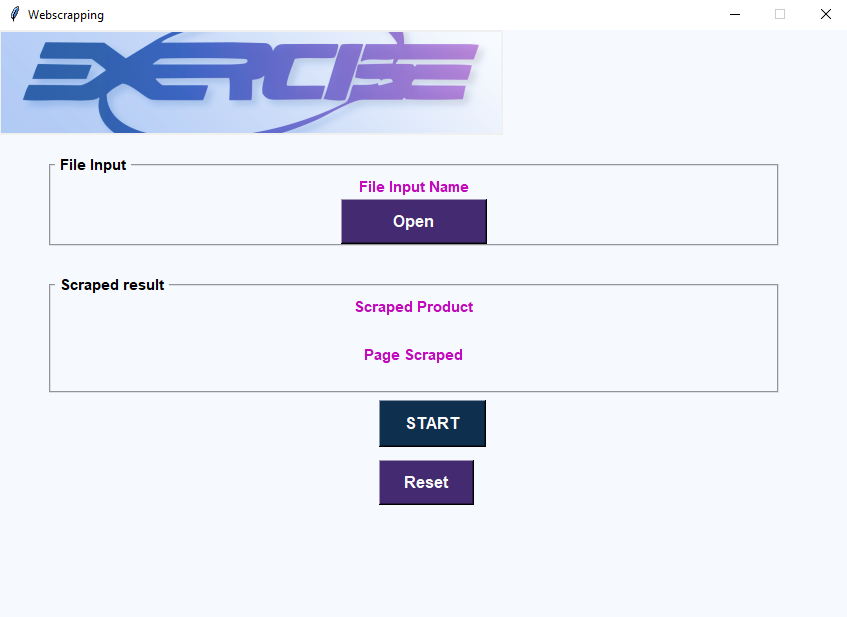
1. Setelah instalasi telah selesai dilakukan, maka pada halaman desktop akan ditampilkan shortcut aplikasi Webscrap Tokped yang dapat dijalankan untuk melakukan scraping



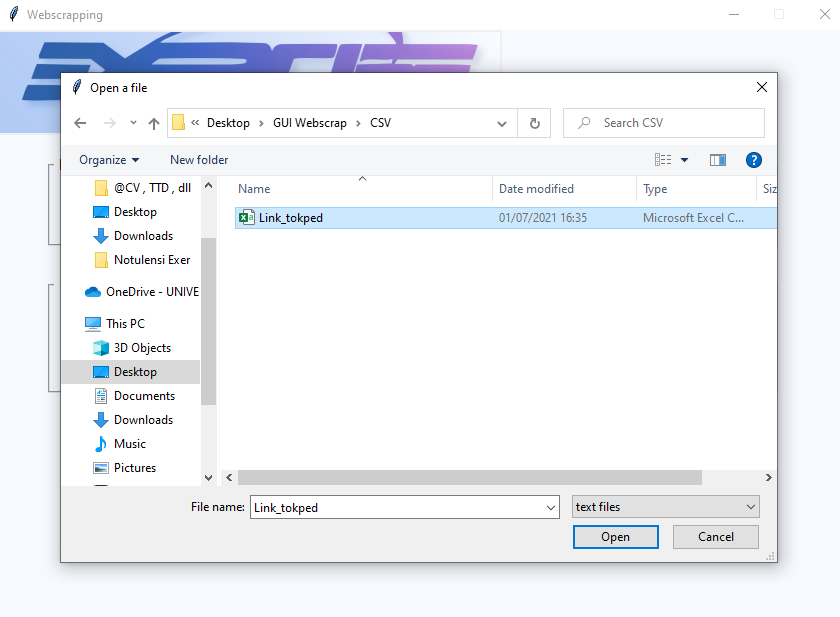
### **Cara Penggunaan**

Jika aplikasi telah selesai di install, selanjutnya dapat dimulai proses scrapping dengan mengikuti langkah berikut:

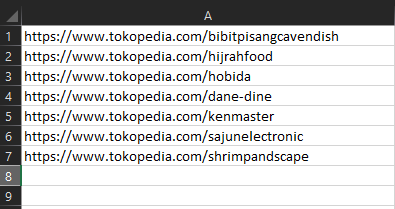
1. Buka file Webscrap Tokped.exe yang akan memberikan tampilan utama aplikasi program Webscrap



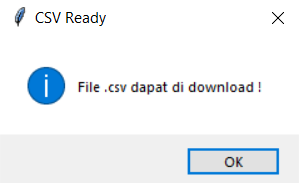
1. Selanjutnya untuk memulai proses scraping, tekan tombol Open dan pilih file CSV yang berisi alamat URL dari toko Tokopedia yang ingin di scrap



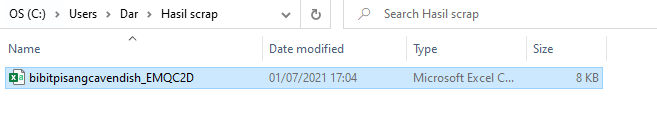
Isi dari CSV tersebut adalah sebagai berikut:



1. Program akan dimulai ketika tombol START telah ditekan, dimana proses scrapping akan dijalankan pada background atau *headless mode*.
2. Ketika proses scraping sudah selesai, maka terdapat *messagebox* yang menampilkan informasi bahwa proses scrapping telah selesai dilakukan



1. Setelah mencapai tahap tersebut, maka hasil webscrap telah dikonversi menjadi csv dengan format nama <nama Toko>\_<6 Digit Unique code> yang akan disimpan pada default path berupa users /<nama user> / Hasil scrap



## Source Code

### **Library**

|  |
| --- |
| from tkinter import \* from tkinter import StringVar from tkinter import messagebox |

→ Library dari tkinter untuk membuat GUI berbasis pemrograman python. Penggunaan library ini ditujukan untuk membuat beberapa widget yang dapat ditampilkan serta dihubungkan pada variabel dari bahasa python.

→ StringVar digunakan untuk menerimpa input dalam bentuk string yang akan di assign pada variabel python

→ Messagebox digunakan untuk menampilkan pesan popup

|  |
| --- |
| from bs4 import BeautifulSoup, NavigableString, Tag |

→ Library dari beautiful soup untuk membantu mengambil format class css yang tidak bisa dilakukan oleh selenium

|  |
| --- |
| from selenium import webdriver from selenium.webdriver.common.by import By from selenium.webdriver.chrome.options import Options |

→ Library selenium untuk menjalankan browser yang dapat diautomasi agar dapat melakukan web scraping pada website dinamis

|  |
| --- |
| import pandas as pd |

→ Library pandas yang digunakan untuk melakukan konversi ke bentu .csv

|  |
| --- |
| import os |

→ Library untuk melakukan penempatan path file dan mengambil letak dari chromedriver

|  |
| --- |
| import time |

→ Library untuk melakukan delay dan juga menghitung lamanya program berjalan

|  |
| --- |
| import string |

→ Library untuk menggunakan beberapa instruksi yang berkaitan dengan string

|  |
| --- |
| import random |

→ Library untuk membuat men*generate* random file name jika tidak dispesifikan pada aplikasi

|  |
| --- |
| import csv |

→ Library untuk mengolah data dengan metode csv

### **Scrapping Code**

#### **Variabel Global**

|  |
| --- |
| daftarToko = [] (string) |

→ Merupakan list dimana program akan menyimpan sementara nama nama toko dari input csv untuk dilakukan proses scrapping

|  |
| --- |
| results=[] (string) |

→ Untuk menyimpan data (nama produk, harga produk, terjual, *rating,* link URL, hatga sebelum diskon dan Jumlah Ulasan) hasil scrapping

|  |
| --- |
| pageCounter=0 (Int) |

→ Merupakan variabel yang berfungsi menentukan hasil *page*(halaman) yang sudah berhasil discrapping

|  |
| --- |
| global jumlahScraped  jumlahScraped=len(results)(Int) |

→ Merupakan variabel penghitung banyaknya produk yang sudah berhasil discrap

|  |
| --- |
| checkerResult=False(boolean) |

→ Parameter yang digunakan untuk menentukan apakah sudah terdapat result sebelum dilakukan proses scrapping.

#### 

**Pengaturan Chrome Drive**

|  |
| --- |
| chrome\_options = Options() chrome\_options.headless = True  → Inisialisasi pengaturan chrome driver untuk menyatakan bahwa digunakan chrome driver bersifat headless  user\_agent = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36' chrome\_options.add\_argument(f'user-agent={user\_agent}') driver = webdriver.Chrome(  executable\_path="chromedriver.exe"  ,chrome\_options=chrome\_options)) |
|  |

→ Mendefinisikan penggunaan web driver chrome headless beserta user agentnya agar terdeteksi sebagai user yang mengakses toko

#### **Fungsi mengecek URL**

|  |
| --- |
| def checker(namaToko): |

→ Untuk mengecek kecocokan input URL pada aplikasi terhadap format webscrapping.

|  |
| --- |
| if len(urlTkinter.get()) == 0:  messageUrl()  return False |

→ Jika input kosong maka ditampilkan message untuk memasukkan input URL toko

|  |
| --- |
| substring = "https://www.tokopedia.com/"  if substring not in namaToko:  mssgUrlError()  return False |

→ Jika input tidak sesuai format maka akan ditampilkan pesan ketidakcocokan

#### **Fungsi Mencari dengan tipe data-testid**

|  |
| --- |
| def findByTestId(element, tag, identifier): |

→ Untuk mencari data yang terkandung dalam <div> data-testid

|  |
| --- |
| data = element.find(tag, attrs={"data-testid": identifier})  return data.text if data else "" |

→ Menyimpan hasil seluruh scrapping pada halaman tersebut dalam variabel data yang akan di return pada instruksi pemanggil

#### **Fungsi Mencari dengan tipe class**

|  |
| --- |
| def findByClassId(element, tag, identifier): |

→ Untuk mencari data yang terkandung dalam <div> class

|  |
| --- |
| data = element.find(tag, class\_=identifier)  return data.text if data else "" |

→ Menyimpan hasil seluruh scrapping pada halaman tersebut dalam variabel data yang akan di return pada instruksi pemanggil

#### **Fungsi mendapatkan dengan link toko untuk Scrapping**

|  |
| --- |
| def getLink(namaToko, page): |

→ Untuk mencari link dengan parameter namaToko dan page

|  |
| --- |
| linkToko = "https://www.tokopedia.com/" + \  str(namaToko)+"/product/page/"+str(page)+"?perpage=80"  return linkToko |

#### **Fungsi untuk mendapatkan Ulasan**

|  |
| --- |
| def getUlasan(link): |

→ Untuk masuk ke dalam link setiap produk dan melakukan proses scraping ulasan

|  |
| --- |
| driver.get(link)  time.sleep(1)  html = driver.page\_source  soup = BeautifulSoup(html,'lxml')  return findByTestId(soup,"span","lblPDPDetailProductRatingCounter") |

→ Membuka link setiap produk dan Memberikan delay untuk transisi setiap page produk. Setelah itu memberikan return berupa hasil scraping ulasan

#### **Fungsi utama Run (berguna untuk mendapatkan data keseluruhan produk)**

def run(namaToko, page):  
→ Menjalankan webdriver untuk menginisialisasi mulainya proses webscrapping

|  |
| --- |
| linkToko = getLink(namaToko, page)  driver.get(linkToko)  scrollbawah = "window.scrollBy(0,500);" |

→ Membuka alamat URL toko yang diinput dan mendefinisikan variabel untuk melakukan scrolling ke bawah sebelum scrapping

|  |
| --- |
| for y in range(1, 11):  driver.execute\_script(scrollbawah)  time.sleep(1) |

→ Ketika link toko sudah terbuka lakukan scrolling agar semua data pada page tersebut dapat terbaca

|  |
| --- |
| html = driver.page\_source  soup = BeautifulSoup(html, 'lxml')  products = soup.find\_all("div", attrs={'data-testid':  "divProductWrapper"}) |

→ Mengambil isi HTML dari page tersebut untuk mencari seluruh kotak produk pada bagian toko pada tokopedia dengan menspesifikasikan parser yang digunakan

|  |
| --- |
| for product in products:  rating = findByClassId(product, "span", "css-etd83i")  linkProduk = product.find("div", class\_="css-7fmtuv").a["href"]  gambarProd = product.find("div", attrs={'data-testid': "imgProduct"}) |

→ Proses looping untuk mengambil data dari product menggunakan fungsi findbyTestId dan findbyClassId serta pengambilan link produk dengan beautiful soup.

|  |
| --- |
| result = {  "Nama": (findByTestId(product, "div", "linkProductName")),  "Harga": (findByTestId(product, "div", "linkProductPrice")),  "Terjual": (findByClassId(product, "span", "css-1kgbcz0")),  "Rating ": rating,  "Link URL ": linkProduk,  "Harga Sebelum Diskon": (findByTestId(product, "div", 'lblProductSlashPrice')),  "Jumlah ulasan": getUlasan(linkProduk) if rating else "",  } |

→ Memasukkan hasil data scrapping ke dalam sebuah list result yang terdiri dari keseluruhan properti pada toko

|  |
| --- |
| results.append(result) |

→ Menambahkan hasil dari scrapping pada looping tertentu ke list result

return len(products) > 0

→ Memberikan konfirmasi bahwa masih terdapat produk di dalam page yang akan discrap

#### **Fungsi Start (Menginisialisasinya mulai proses scrapping di web driver)**

|  |
| --- |
| def start():  global listTokoFull  for namaToko in listTokoFull :  print(namaToko) |

→ Proses untuk menscrap setiap toko satu persatu dari data input csv nama toko

|  |
| --- |
| global checkerResult  if checkerResult==True:  clear() |

→ Parameter yang mengecek apakah sudah terdapat result sebelum melakukan proses scrapping, bila sudah maka result akan direset kembali menjadi kosong

|  |
| --- |
| Page = 1 global pageCounter  checkerResult = True |

→ Mendefine Page ke untuk mulai proses scrapping dan Membuat variabel global pageCounter

|  |
| --- |
| while True:  print(namaToko)  if run(namaToko, page) == False:  break  jumlahScraped = len(results)  page+= 1  pageCounter += 1  berubah(jumlahScraped) |

→ Metode untuk menjalankan scrapping dengan cara memanggil fungsi run yang akan melakukan proses scrapping per produknya, dan akan berhenti setelah tidak ada produk dalam toko tersebut.

|  |
| --- |
| print(results)  print(pageCounter)  download(namaToko) |

→ Menjalankan fungsi download yang akan menghasilkan file hasil scrapping

|  |
| --- |
| if len(results) == 0:  messageNotice()  else:  messageDone() |

→ Merupakan error handling yang menyatakan apabila tidak terdapat produk maka akan keluar sebuah peringatan atau mengingatkan pesan scrap sudah selesai bekerja.

#### **Fungsi Random Generator (Mengenerate nama file random)**



|  |
| --- |
| def id\_generator(size=6, chars=string.ascii\_uppercase + string.digits):  return ''.join(random.choice(chars) for \_ in range(size)) |

→ Membentuk nama random dengan karakter dan nomor yang akan digabung dengan penamaan nama toko

#### **Fungsi Download()**

|  |
| --- |
| def download(nama\_csv, path):  if len(results) == 0:  messageNotice() |

→ Jika isi list results kosong maka download tidak bisa dilakukan dan akan ditampilkan pesan

|  |
| --- |
| nama\_csv = namaToko + "\_" +id\_generator()  homedir = os.path.expanduser("~")  path = homedir+'\Hasil scrap' |

→ Memberikan penamaan pada hasil scraping serta menyimpan pada default path yaitu dengan membuat folder Hasil Scrap pada user

|  |
| --- |
| if os.path.exists(path) == False:  os.makedirs(path) |

→ Sedangkan, Jika file Hasil Scrap sudah ada, maka tidak membuat folder baru

|  |
| --- |
| df = pd.DataFrame(results)  df.index+=1  df.to\_csv( path + '/' + nama\_csv + '.csv', sep=';', index=1, index\_label='No') |

→ Mengkonversi hasil list pada result menjadi file csv yang akan ditampilkan dengan pesan berhasil

#### **Fungsi semuaToko()**

|  |
| --- |
| def semuaToko(dataCSV): |

→ Merupakan fungsi yang mengolah nama toko input sesuai dengan format parameter pemrograman

|  |
| --- |
| for item in dataCSV:  linkToko = ''.join(str(e) for e in item)  namaTokoScrap = linkToko.strip()  namaTokoScrap = namaTokoScrap.replace("https://www.tokopedia.com/", "")  namaTokoScrap = namaTokoScrap.replace(";", "")  if namaTokoScrap=="":  break  listTokoFull.append(namaTokoScrap)  print(listTokoFull) |

→ Setiap nama toko dalam input csv akan diformat dengan menghilangkan spasi, <https://www.tokopedia.com/>, sehingga sesuai format parameter pemrograman

#### **Fungsi baca()**

|  |
| --- |
| def baca(InputPath):  with open(InputPath, newline='') as f:  reader = csv.reader(f)  dataCSV = list(reader)  semuaToko(dataCSV) |

→ Merupakan fungsi input user yang akan membuka file explorer dan dapat memilih file csv yang akan digunakan untuk proses scrapping

### **GUI Code**

|  |
| --- |
| main\_window = Tk() main\_window.title("Webscrapping")  main\_window.geometry("850x600")  main\_window.configure(bg="#F6F9FE") |

→ Menginisiasi tkinter dengan layout dan background yang didefinisikan

|  |
| --- |
| photo = PhotoImage(file="exer\_kecil.png") photo\_label = Label(main\_window, image=photo) photo\_label.grid(row=0, column=1, sticky = "") |

→Menampilkan logo exercise pada aplikasi

|  |
| --- |
| urlTkinter = StringVar() changeFileName = StringVar() changeFilePath = StringVar() jumlahScrapedTampil=StringVar() pageCounterTampil=StringVar() |

→ Mendefinisikan variabel string yang akan digunakan untuk menerima input atau menampilkan string pada aplikasi

|  |
| --- |
| InputPath = [] |

→ Variabel untuk menampung nama file

|  |
| --- |
| def select\_file\_input():  filetypes = (  ('text files', '\*.csv'),  ('All files', '\*.csv')  )   filename = fd.askopenfilename(  title='Open a file',  initialdir='/',  filetypes=filetypes)    if(filename == ""):  showinfo(  title='Selected File',  message = "Anda Tidak memilih File !"  )  else:  showinfo(  title='Selected File',  message="Anda Telah Memilih File" + filename  )  global InputPath  InputPath = filename  baca(InputPath) |

→ Fungsi yang akan dipanggil ketika tombol input dipencet dan akan mengeluarkan file explorer untuk memudahkan pemilihan file input csv

|  |
| --- |
| # File Input Location  FileInput= LabelFrame(text=" File Input ", font=(  'roboto', 11, 'bold'), bg="#F6F9FE") InpFile = Label(FileInput, text="File Input Name", fg="#C10ABA",  bg="#F6F9FE", font=('roboto', 11, 'bold'), width=80) namaFile\_in = Button(FileInput, padx=19.4, pady=7, text=" Open ", fg="#FAFAFA", borderwidth=2,  bg="#442B71",relief=RAISED, font=('roboto', 12, "bold"), width=10, command=select\_file\_input) InpFile.pack() namaFile\_in.pack() FileInput.place(x=50, y=125) |

→ Sebagai tombol untuk memilih file CSV yang berisi link toko tokopedia untuk di scrap

|  |
| --- |
| Reset = Button(main\_window, text="Reset" ,bg="#442B71",   fg="#FAFAFA",borderwidth=2, padx=19.4, pady=7, relief=RAISED,font=('roboto', 12, 'bold')) StartButton = Button(main\_window, padx=20, pady=8, text="START", relief=RAISED, font=(  'roboto', 12, "bold"), fg="#FAFAFA", bg="#0F2F4F", command=start) StartButton.place(x=380, y=370) Reset.place(x=380, y=430) |

→ Sebagai tombol start untuk memulai proses scrapping dan tombol reset untuk mengosongkan seluruh list

|  |
| --- |
| def clear():  global pageCounter  jumlahScrapedTampil.set("0")  pageCounter=0  pageCounterTampil.set("0")  urlTkinter.set("")  changeFileName.set("")  changeFilePath.set("")  results.clear() |

→ Untuk mereset semua variabel pada program ketika ditekan tombol reset pada aplikasi

|  |
| --- |
| Scraped= LabelFrame(text=" Scraped result ",font=('roboto', 11, 'bold'), bg="#F6F9FE") scrpd = Label(Scraped, text='Scraped Product' , fg="#C10ABA", bg="#F6F9FE", font=('roboto', 11, 'bold'), width=80) Data = Label(Scraped, textvariable=var\_scraped , fg="#C10ABA", bg="#F6F9FE", font=('roboto', 11, 'bold'), width=80) scrpdPage = Label(Scraped, text='Page Scraped' , fg="#C10ABA", bg="#F6F9FE", font=('roboto', 11, 'bold'), width=80) Page = Label(Scraped, textvariable = var\_page, fg="#C10ABA", bg="#F6F9FE",font=('roboto', 11, 'bold'), width=80, height=1) scrpd.pack() Data.pack() scrpdPage.pack() Page.pack() Scraped.place(x=50, y= 245) |

→ Menampilkan progress dari scrapping yang dilakukan berupa banyak produk dan halaman

|  |
| --- |
| def berubah(jumlahScraped):  jumlahScrapedTampil.set(jumlahScraped)  pageCounterTampil.set(pageCounter)  Scraped = LabelFrame(text=" Scraped result ", font=(  'roboto', 11, 'bold'), bg="#F6F9FE")  scrpd = Label(Scraped, text='Scraped Product', fg="#C10ABA",  bg="#F6F9FE", font=('roboto', 11, 'bold'), width=80)  Data = Label(Scraped, textvariable=jumlahScrapedTampil, fg="#C10ABA",  bg="#F6F9FE", font=('roboto', 11, 'bold'), width=80)  scrpdPage = Label(Scraped, text='Page Scraped', fg="#C10ABA",  bg="#F6F9FE", font=('roboto', 11, 'bold'), width=80)  Page = Label(Scraped, textvariable=pageCounterTampil, fg="#C10ABA",  bg="#F6F9FE", font=('roboto', 11, 'bold'), width=80, height=1)  scrpd.pack()  Data.pack()  scrpdPage.pack()  Page.pack()  Scraped.place(x=50, y=245) |

→ Fungsi yang dipanggil ketika dilakukan perubahan pada variabel jumlahScraped dan pageCounter agar terjadi perubahan pada label aplikasi juga

|  |
| --- |
| def messageUrl(): msgShow = messagebox.showinfo("Insert URL", "Masukkan Link Tokopedia untuk di scrap") |

→ Messagebox yang ditampilkan ketika link url belum dimasukkan tetapi menekan tombol start



|  |
| --- |
| def messageNotice(): msgShow = messagebox.showinfo("Info", "Tidak ada data untuk dikonversi ke CSV !") |

→ Messagebox yang ditampilkan ketika belum dilakukan proses scrapping namun telah menekan tombol download



|  |
| --- |
| def mssgUrlError():  msgShow = messagebox.showinfo(  "URL wrong", "Input URL tidak sesuai dengan format") |

→ Message yang ditampilkan jika URL yang dimasukkan salah

|  |
| --- |
| def messageWarning(): msgShow = messagebox.showinfo("Error", "Masukan Path untuk penyimpanan CSV") |

→ Messagebox yang ditampilkan ketika path yang diberikan tidak sesuai



|  |
| --- |
| def messageSccs(): msgShow = messagebox.showinfo("Berhasil", "File CSV berhasil dikonversi !") |

→ Messagebox yang ditampilkan ketika hasil konversi ke bentuk .csv telah berhasil dilakukan

|  |
| --- |
| def messageDone():  msgShow = messagebox.showinfo("Selesai", "Seluruh File CSV telah selesai didownload") |

→ Messagebox yang ditampilkan ketika semua file CSV hasil scrapping sudah selesai dikonversi dan diletakkan pada default file path

|  |
| --- |
| def messageStartClear():  msgShow = messagebox.showinfo("Scrap direset", "Proses baru akan dimulai \n Hasil Scrap sebelum akan dihapus") |

→ Messagebox yang ditampilkan ketika proses scrapping kembali berjalan tanpa ada proses reset terlebih dahulu

## Performance Speed

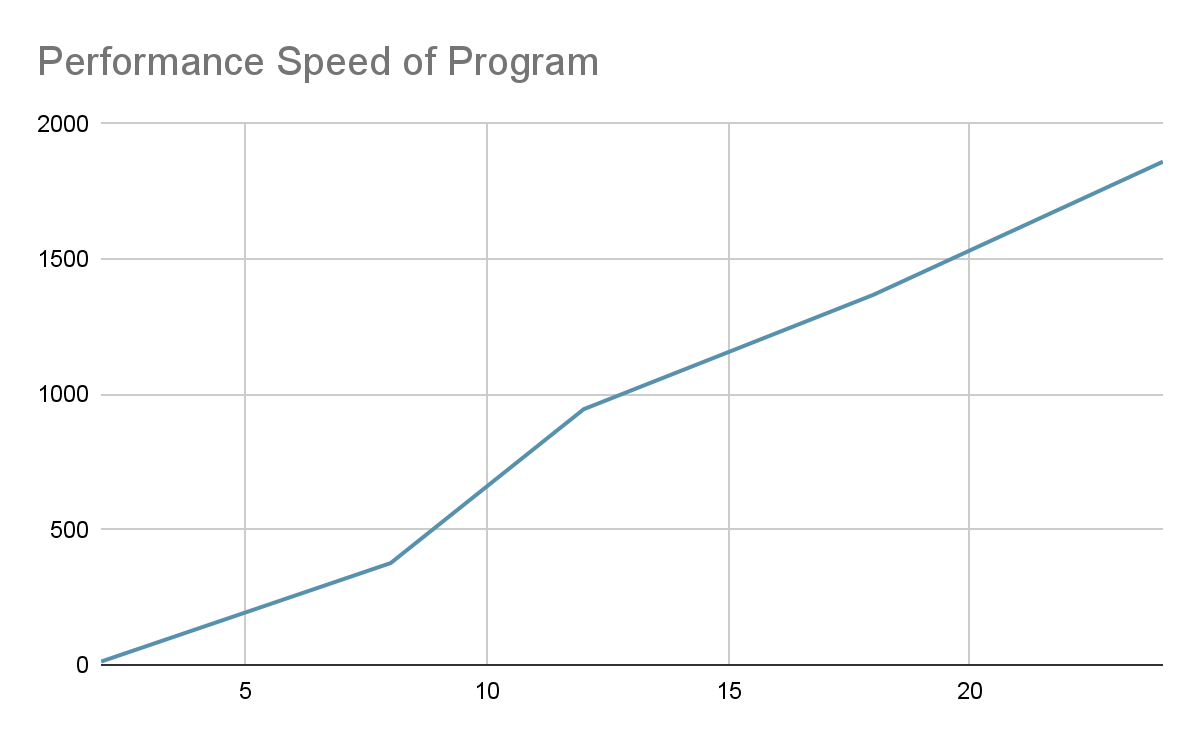
### **Tabel**

Pengujian kecepatan eksekusi program Web Scraping dengan menggunakan library Time dari Python:

|  |  |
| --- | --- |
| Banyak Page | Waktu (s) |
| 2 | 12.34 |
| 8 | 376,43 |
| 12 | 945.174 |
| 18 | 1367.827 |
| 24 | 1860.11 |

Namun, Dalam pengujian tersebut diperlukan adanta peritimbangan faktor lainnya seperti algoritma program yang akan membuka produk untuk melihat banyak ulasannya. Sedangkan untuk produk tertentu tidak diperlukan untuk membuka produk tersebut sehingga terdapat kemungkinan adanya perbedaan waktu pada setiap proses scraping untuk toko yang berbeda juga. Diluar hal itu, kecepatan processor dan jenis memory yang digunakan juga dapat mempengaruhi kecepatan dari program Web Scraping yang dijalankan.

### **Grafik**



Hasil dari proses eksekusi program tidak menghasilkan grafik yang linear karena untuk beberapa toko tidak semua produk memiliki penjualan. Sehingga proses scraping untuk Toko yang memiliki sedikit penjualan akan lebih cepat dijalankan.