



Assignment Cover Letter

(Individual Work)

Student Information:

1.

Surname

 Sukardi
Teja

Given Names

Andreas

Student ID Number

2301900416

Course Code : COMP6056

Course Name : Introduction to Programming

Class : L1AC

Name of Lecturer(s) : Ida Bagus Kerthyayana Manuaba

Major : CS

Title of Assignment : Python Search Engine
(if any)

Type of Assignment : Final Project

Submission Pattern
Due Date : 17-1-2020

Submission Date : 12-1-2020

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

(Name of Student)

Andreas Sukardi Teja

“Python Search Engine”

Name : Andreas Sukardi Teja

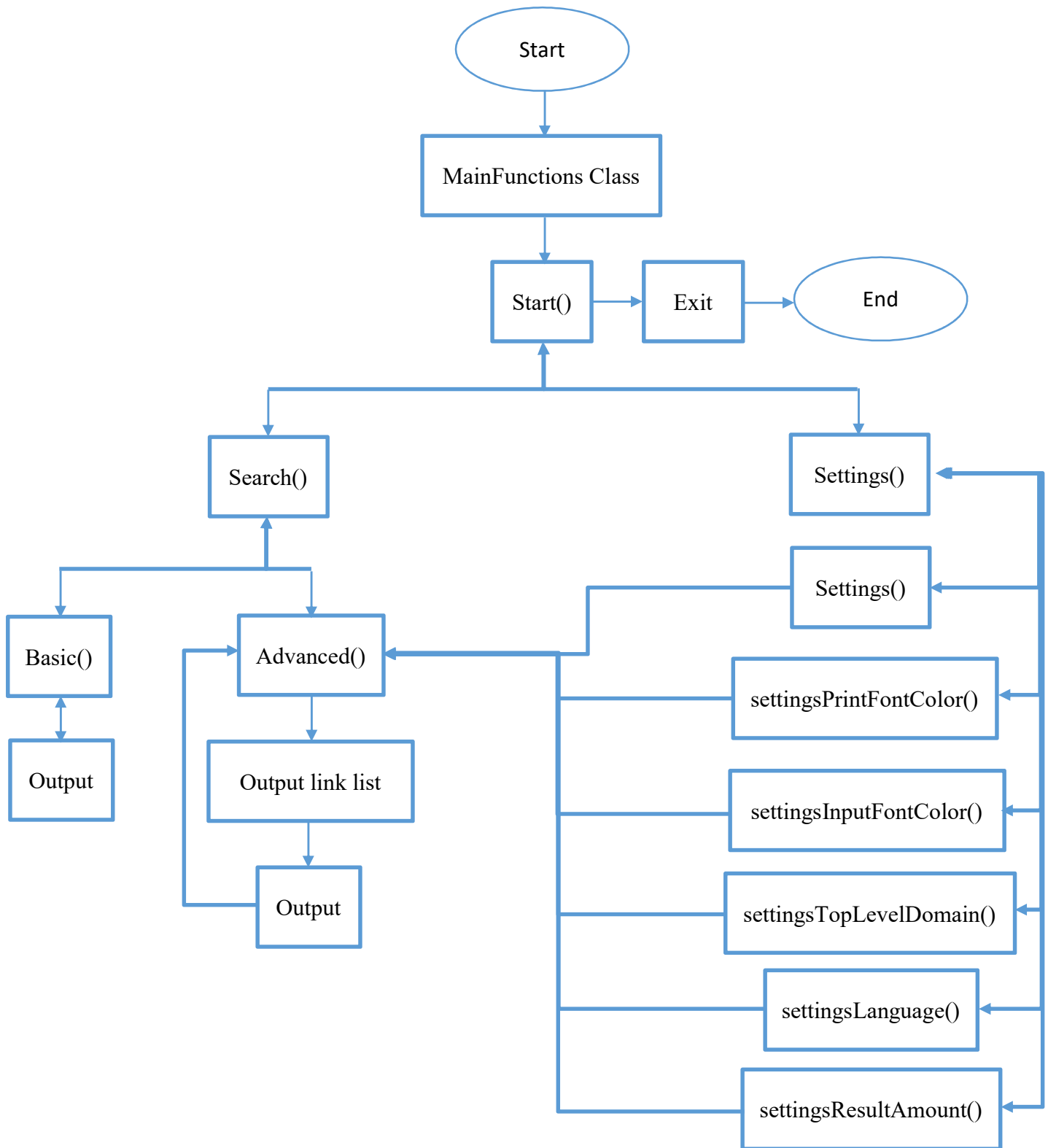
ID : 2301900416

I. Description**The function of this program:**

The purpose of this program is take a query from the user in the form of text before using the module **googlesearch** to send a HTTPS request to Google which Google would then return a list of links related to the query and then using the module **webbrowser** to open said link from the default browser of the user's computer. The program also allows the user to change aspects of the HTTPS request such as **language**, **top level domain** and **result amount**. It also has two modes, **basic** and **advanced**. **Basic** simply opens the first link that Google returns while **Advanced** would print a numbered list of links that Google returns, the results shown would be affected by the selected **language** and **top level domain** while how many is shown would be affected by the **result amount** set, the user would then be able to select which link to open by inputting the number assigned to the link. It also has two auxiliary settings that can be changed namely **print font color** and **input font color**, **print font color** only changed the font color of the numbers in the numbered list when using **advanced** mode and the description of a link (if provided by Google) while **input font color** changes the font color of the questions that the program is asking for an input from the user, both functions use the modules **Colorama** and **Termcolor** to allow this. The program also allows the user to exit the program from inside it using the **Sys** module.

II.a. Design/Plan

Project's Hierarchy Chart



II.b.

Explanation of Each Function Inside the Class

Final_Project.py

- **__init__(self, mode=1, printFontColor='white', inputFontColor='cyan', top-LevelDomain='com', language='en', resultAmount=10):**
 - Used to set the default settings.
- **Start(self):**
 - Accepts input from the user to detect:
 - Search or s - The program will initialize the search engine.
 - Settings or se - The program will initialize the setting options.
 - help or h - The program will display the possible commands.
 - Exit or e - The program will exit.
 - Abnormal data - The program will print an error message telling the user to use 'help' to view the available commands.
- **search(self):**
 - Detects which mode it is on.
- **basic(self):**
 - Accepts input from the user as the query.
 - 'SearchExit' to go back to **Start(self)**.
 - Automatically opens the first link Google returns.
- **advanced(self):**
 - Accepts input from the user as the query.
 - 'SearchExit' to go back to **Start(self)**.
 - Generates a numbered list of links.

- Accepts input from the user as the choice.
- Opens the link the user chose.

- **settings(self):**
 - Accepts input from the user to detect:
 - Mode or m - Starts settingsMode(self):
 - Print font color or p - Starts settingsPrintFontColor(self):
 - Input font color or i - Starts settingsInputFontColor(self):
 - Top Level Domain or t - Starts settingsTopLevelDomain(self):
 - Language or l - Starts settingsLanguage(self):
 - Result Amount or r - Starts settingsResultAmount(self):
 - help or h - The program will the possible commands.
 - back or b - The program will display the previous section.
 - Abnormal data - The program will print an error message telling the user to use 'help' to view the available commands.

- **settingsMode(self):**
 - Accepts input from the user to detect:
 - basic or b- The program will automatically open the first link that Google provides.
 - advanced or a- The program will display links equal in amount with the value specified in Result Amount before letting the user choose which link to open.
 - help or h - The program will display the possible commands.
 - back or b - The program will display the previous section.

- Abnormal data - The program will print an error message telling the user to use 'help' to view the available commands.

- **settingsPrintFontColor(self):**

- Accepts input from the user to detect:
 - green - The program will ask the user to pick another option due to it already being used for helpFontColor.
 - red - The program will ask the user to pick another option due to it already being used for errorFontColor.
 - yellow, blue, magenta, cyan and white - Changes the color of prints according to the chosen color.
 - help or h - The program will display the possible commands.
 - back or b - The program will display the previous section.
 - Abnormal data - The program will print an error message telling the user to use 'help' to view the available commands.

- **settingsInputFontColor(self):**

- Accepts input from the user to detect:
 - green - The program will ask the user to pick another option due to it already being used for helpFontColor.
 - red - The program will ask the user to pick another option due to it already being used for errorFontColor.
 - yellow, blue, magenta, cyan and white - Changes the color of prints according to the chosen color.
 - help or h - The program will display the possible commands.
 - back or b - The program will display the previous section.
 - Abnormal data - The program will print an error message telling the user to use 'help' to view the available commands.

- **settingsTopLevelDomain(self):**

- Accepts input from the user to detect:
 - com, net, co.id and com.sg - Changes the TLD into the chosen TLD.
 - org, edu, gov and etc. - The program will ask the user to pick another option due to the TLD being blocked by Indonesia.
 - help or h - The program will display the possible commands.
 - back or b - The program will display the previous section.
 - Abnormal data - The program will print an error message telling the user to use 'help' to view the available commands.

- **settingsLanguage(self):**

- Accepts input from the user to detect:
 - en - English (US)
 - en-GB - English (UK)
 - zh-CN - Chinese (PRC)
 - zh-TW - Chinese (Taiwan)
 - es - Spanish
 - hi - Hindi
 - ar - Arabic
 - ms - Malay
 - ru - Russian
 - bn - Bengali
 - pt-BR - Portuguese (Brazil)
 - pt-PT - Portuguese (Portugal)
 - fr - French

- id – Indonesian
 - help or h - The program will display the possible commands.
 - back or b - The program will display the previous section.
 - Abnormal data - The program will print an error message telling the user to use 'help' to view the available commands.
- **settingsResultAmount(self):**
 - Accepts input from the user to detect:
 - Integers above 0 - Changes the result amount to the input.
 - Integers below or 0 – The program will print an error message telling the user to only integer values above 0.
 - Non-Integers - The program will print an error message telling the user to only input integers.
 - help or h - The program will display the possible commands.
 - back or b - The program will display the previous section.

Class Diagram



III.a. Lessons that Have Been Learned

1. *The use of search() in Googlesearch:*

```
from googlesearch import search      # The main module needed to allow the
google search.
```

After a little bit of research, I found a module named **Googlesearch** which gave me access to the search() function.

Basic algorithm:

```
for i in search(query, stop=1):      # Searches the item, it will first
output a generator which 'i' will convert into a link which we can actually
use.
```

Advanced algorithm:

```
for i in search(query, tld=self.topLevelDomain, lang=self.language,
num=100, stop=self.resultAmount):
```

2. *The use of Colorama and Termcolor:*

```
from colorama import init            # Optional module to add color func-
tionality.
from termcolor import colored        # Optional module to assign the colors
to text.
```

After a bit of research on how to add font colors, I found about various methods to do it but decided on using **Colorama** and **Termcolor**.

The colors available from **Colorama**:



taken from pypi.org

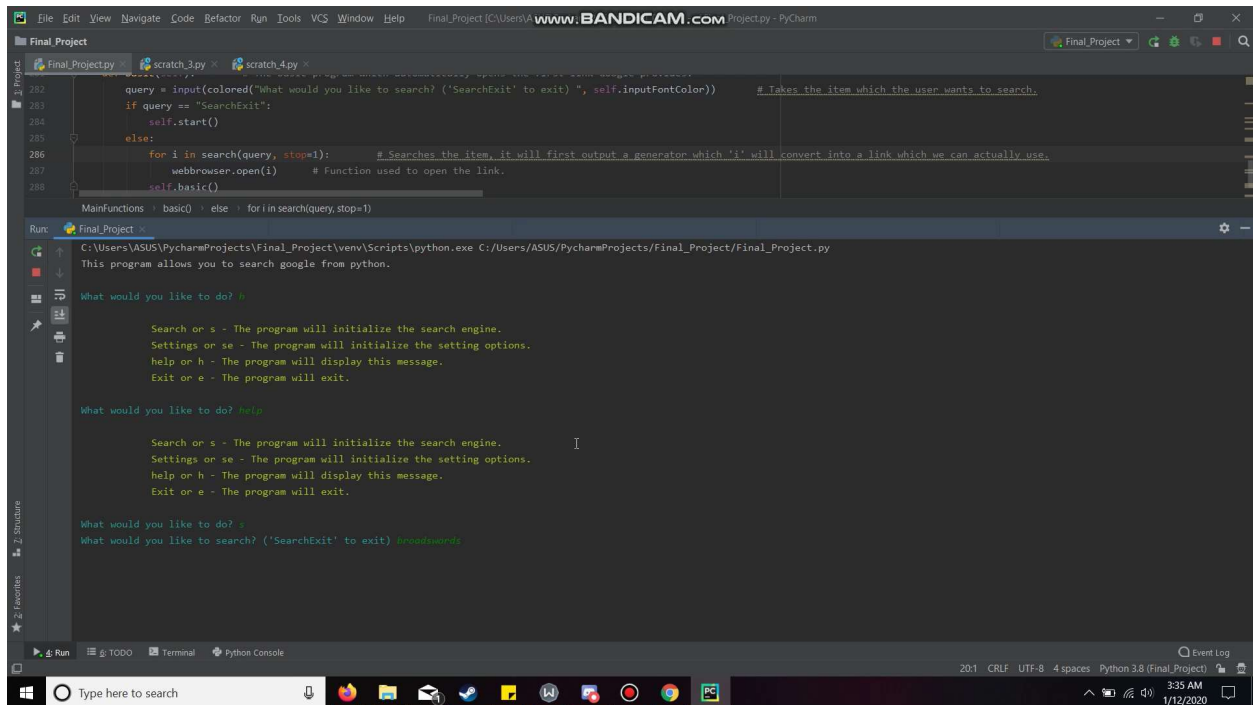
The use of `colored()` from **Termcolor**:

```
settingsOption = input(colored("Which setting would you like to change? ",  
self.inputFontColor))
```

III.b. Problem that Have Been Overcome

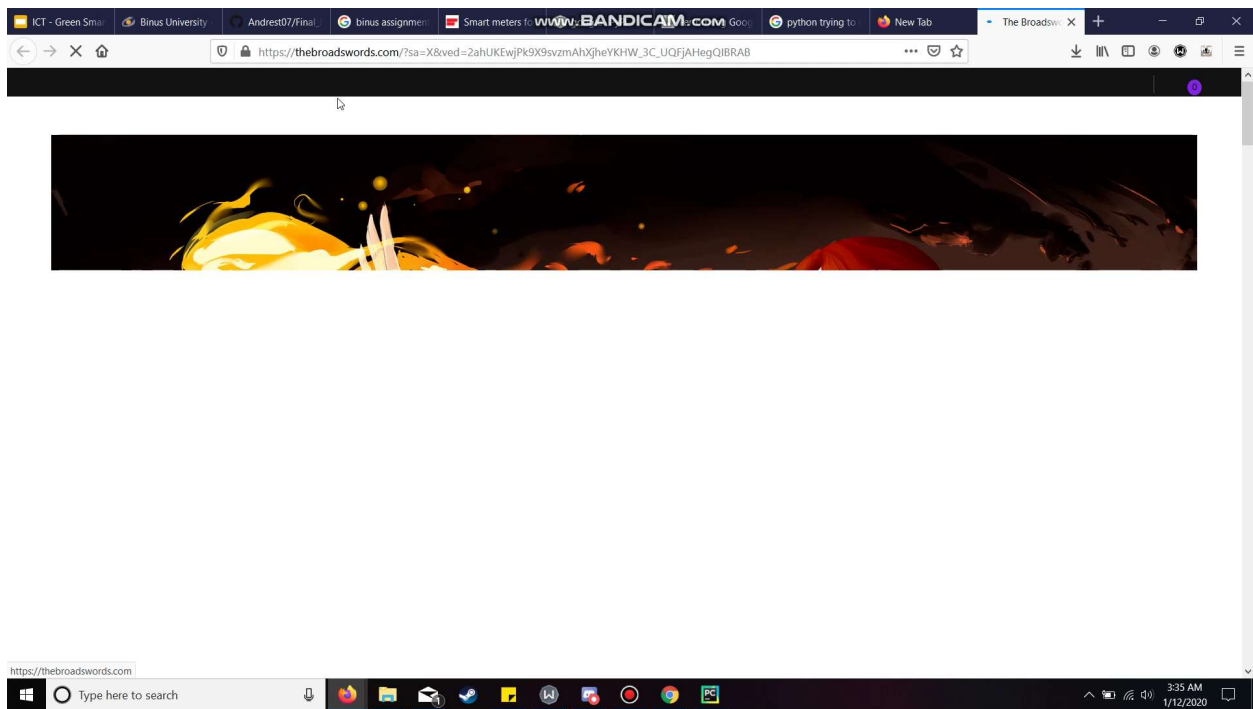
Like making any type of larger program in a programming language you aren't fully familiar with yet, it's just plain annoying and honestly not that hard when something doesn't work due to the way the programming language handles things and you have to restructure or, in the worst case scenario, remake everything. In my case I initially had two separate Classes, one to handle the main functions such as `search()`, `settings()` and `exit` and another to handle the settings where the default settings are initialized and also where the user can change said settings individually. During testing I found an issue with trying to do things this way and although I believe there is a method to fix it, I decided that it's simpler to dump the two-class plan and just merge it into one. Furthermore during testing, I found out that some TLDs are in fact rendered unusable by presumably Indonesia as general and Indonesian TLDs work but not TLDs for other nations. I also found some HTTPS errors caused by abnormal data inputs in `settingsResultAmount()`, due to some funky mechanics of how `try`, `except` and `else` is handled by Python, I had to go through about 3 iterations of the original algorithm to make it work consistently during stress tests. I attempted using `except` to block the HTTPS error but apparently that particular error isn't even listed in the `except` error list, research into the internet also did not turn up any similar issues so I had to solve it myself. Bug-fixing and maybe the help texts were the worst.

IV. Evidence

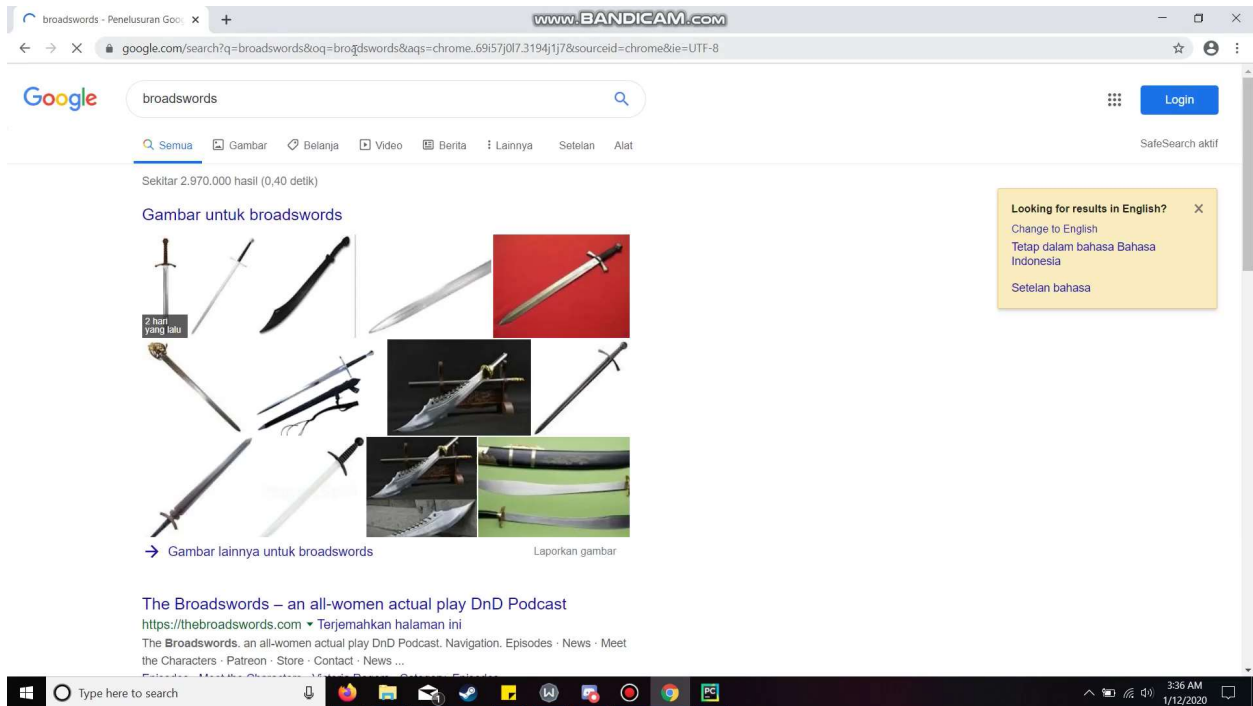


```
File Edit View Navigate Code Refactor Run Tools VCS Window Help Final_Project [C:\Users\A\www.BANDICAM.com] Project.py - PyCharm
Final_Project
Final_Project.py scratch_3.py scratch_4.py
282 query = input(colored("What would you like to search? ('SearchExit' to exit) ", self.inputFontColor)) # Takes the item which the user wants to search.
283 if query == "SearchExit":
284     self.start()
285 else:
286     for i in search(query, stop=1): # Searches the item, it will first output a generator which 'i' will convert into a link which we can actually use.
287         webbrowser.open(i) # Function used to open the link.
288     self.basic()
MainFunctions basic() else for i in search(query, stop=1)
Run: Final_Project x
C:\Users\ASUS\PycharmProjects\Final_Project\venv\Scripts\python.exe C:\Users\ASUS\PycharmProjects\Final_Project\Final_Project.py
This program allows you to search google from python.
What would you like to do? h
Search or s - The program will initialize the search engine.
Settings or se - The program will initialize the setting options.
help or h - The program will display this message.
Exit or e - The program will exit.
What would you like to do? help
Search or s - The program will initialize the search engine.
Settings or se - The program will initialize the setting options.
help or h - The program will display this message.
Exit or e - The program will exit.
What would you like to do? s
What would you like to search? ('SearchExit' to exit) broadsword
20:1 CRLF UTF-8 4 spaces Python 3.8 (Final_Project) 3:35 AM 1/12/2020
```

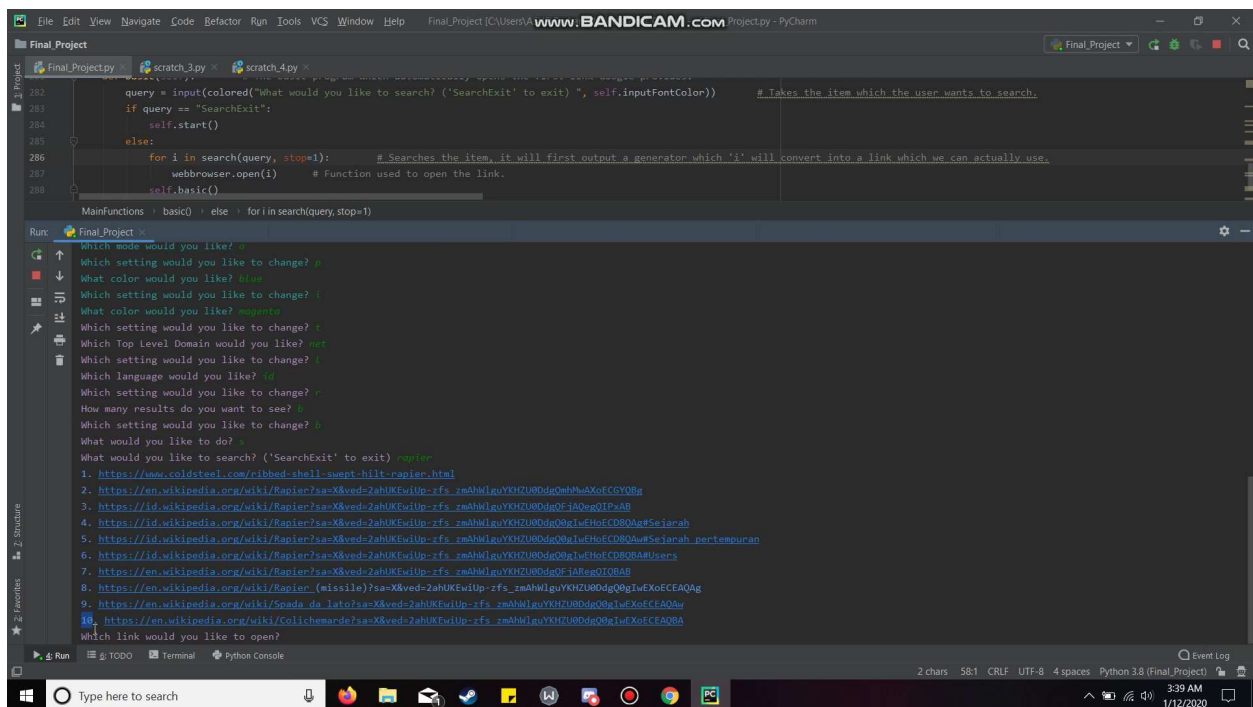
Basic Search program interface



Website opened by the program



First result returned by Google in a clean browser is the same link



Changing the settings and doing an advanced search

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help Final_Project (C:\Users\A\www.BANDICAM.com\Project) - PyCharm
Final_Project
Final_Project.py scratch_3.py scratch_4.py
282 query = input(colored("What would you like to search? ('SearchExit' to exit) ", self.inputFontColor)) # Takes the item which the user wants to search.
283 if query == "SearchExit":
284     self.start()
285 else:
286     for i in search(query, stop=1): # Searches the item, it will first output a generator which 'i' will convert into a link which we can actually use.
287         webbrowser.open(i) # Function used to open the link.
288     self.basic()
MainFunctions basic() else for i in search(query, stop=1)
Run: Final_Project
Which word would you like? m
Which setting would you like to change? p
What color would you like? blue
Which setting would you like to change? i
What color would you like? magenta
Which setting would you like to change? c
Which Top Level Domain would you like? net
Which setting would you like to change? l
Which language would you like? id
Which setting would you like to change? n
How many results do you want to see? 5
Which setting would you like to change? b
What would you like to do? s
What would you like to search? ('SearchExit' to exit) rapier
1. https://www.coldsteel.com/ribbed-shell-swept-hilt-rapier.html
2. https://en.wikipedia.org/wiki/Rapier?sa=X&ved=2ahUKEwIUp-zfs_zmAHWlguYKHZU0DdgQmhmMwAXoECGYQBg
3. https://id.wikipedia.org/wiki/Rapier?sa=X&ved=2ahUKEwIUp-zfs_zmAHWlguYKHZU0DdgQmhmMwAXoECGYQBg
4. https://id.wikipedia.org/wiki/Rapier?sa=X&ved=2ahUKEwIUp-zfs_zmAHWlguYKHZU0DdgQmhmMwAXoECGYQBg
5. https://id.wikipedia.org/wiki/Rapier?sa=X&ved=2ahUKEwIUp-zfs_zmAHWlguYKHZU0DdgQmhmMwAXoECGYQBg
6. https://id.wikipedia.org/wiki/Rapier?sa=X&ved=2ahUKEwIUp-zfs_zmAHWlguYKHZU0DdgQmhmMwAXoECGYQBg
7. https://en.wikipedia.org/wiki/Rapier?sa=X&ved=2ahUKEwIUp-zfs_zmAHWlguYKHZU0DdgQmhmMwAXoECGYQBg
8. https://en.wikipedia.org/wiki/Rapier_(missile)?sa=X&ved=2ahUKEwIUp-zfs_zmAHWlguYKHZU0DdgQmhmMwAXoECGYQBg
9. https://en.wikipedia.org/wiki/Spada_da_lato?sa=X&ved=2ahUKEwIUp-zfs_zmAHWlguYKHZU0DdgQmhmMwAXoECGYQBg
10. https://en.wikipedia.org/wiki/Colichemarde?sa=X&ved=2ahUKEwIUp-zfs_zmAHWlguYKHZU0DdgQmhmMwAXoECGYQBg
Which link would you like to open?
```

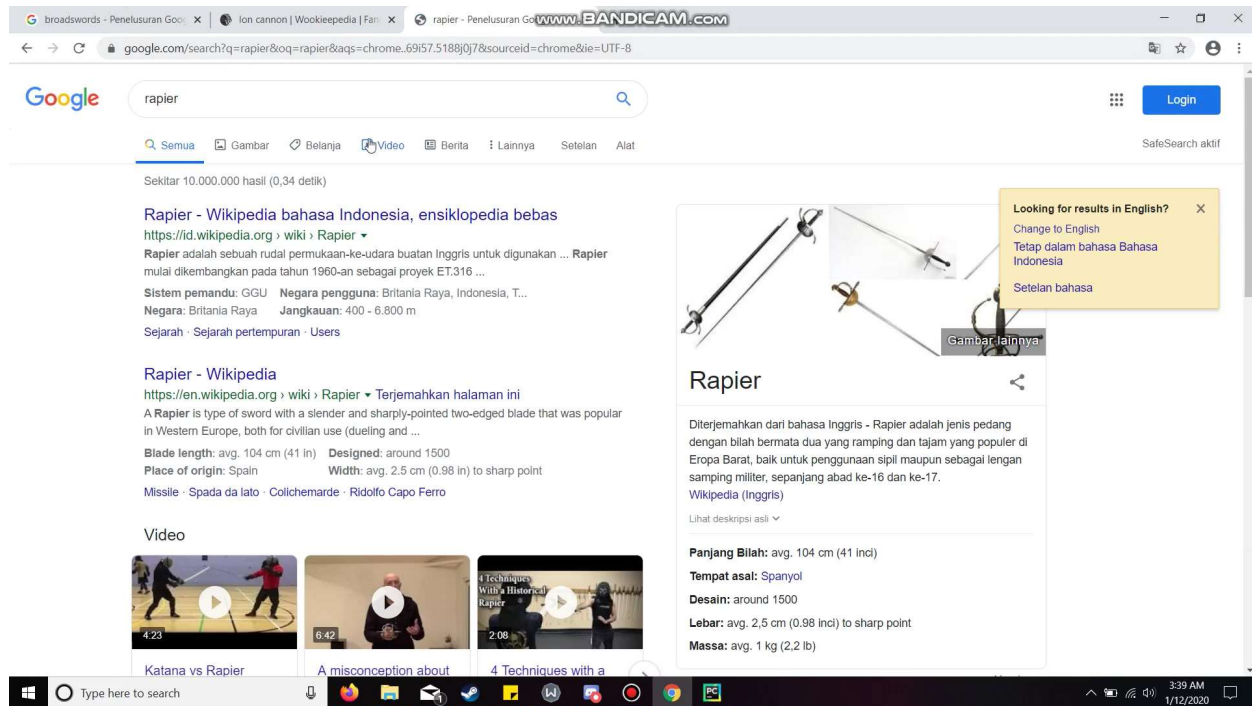
Selecting link 2

Wikipedia article for **Rapier**. The article describes the rapier as a type of sword with a slender and sharply-pointed two-edged blade, popular in Western Europe from the 16th to 17th centuries. It mentions that rapier fencing was popularized by writers like Salvator Fabris and Ridolfo Capo Ferro, and that the French smallsword was a direct continuation of this tradition.

The article also includes a table of specifications for the rapier, such as its mass, blade length, width, and hilt type.

Rapier / espada ropera	
Type	Sword
Place of origin	Spain ^[1]
Production history	
Designed	around 1500
Specifications	
Mass	avg. 1 kg (2.2 lb)
Blade length	avg. 104 cm (41 in)
Width	avg. 2.5 cm (0.98 in) to sharp point
Blade type	single or double edged, straight blade
Hilt type	complex, protective hilt

Website opened by the program



Second result returned by Google in a clean browser is the same link

Important Note: Getting the same result in a separate browser and the program is very difficult due to the browser and the program each having their own different cookie from Google and each having different settings which may result in different links being provided by Google.

V. Resources

- [geeksforgeeks.org](https://www.geeksforgeeks.org) – Learning how to use search()
- pypi.org – To get the list of colors.
- stackoverflow.com – Mostly for bug-fixing but also module finding.

V. Source Code

Final_Project.py

```
# First, pip installing 'google' is necessary for this code to work.
# The module 'google' also has a dependency on 'beautifulsoup' but
# pip install should install 'beautifulsoup' automatically.
# Importing colorama and termcolor is optional, I'm using it only for text font
color.

from googlesearch import search      # The main module needed to allow the google
search.
from colorama import init            # Optional module to add color functionality.
from termcolor import colored        # Optional module to assign the colors to text.

# Using 'from' to reduce risks of name collisions

import sys                          # To allow exiting the program from inside the program.
import webbrowser                    # The main module needed to allow opening links.

init()                              # Initializing colorama.
errorFontColor = 'red'
helpFontColor = 'green'

# Putting these variables outside the classes to ensure they are not edited acci-
dentally.

class MainFunctions:
    def __init__(self, mode=1, printFontColor='white', inputFontColor='cyan', top-
LevelDomain='com', language='en',
                resultAmount=10):    # Used to assign default settings.
        self.mode = mode
        self.printFontColor = printFontColor
        self.inputFontColor = inputFontColor
        self.topLevelDomain = topLevelDomain
        self.language = language
        self.resultAmount = resultAmount

    def settings(self):              # Used to start up the correct class function according
to the setting the user chose.
        settingsOption = input(colored("Which setting would you like to change? ",
self.inputFontColor))
        if settingsOption == "Mode" or settingsOption == "m":
            self.settingsmode()      # Starts up the class function to change the
mode.
        elif settingsOption == "Print font color" or settingsOption == "p":
            self.settingsPrintFontColor()    # Starts up the class function to
change the print font color.
        elif settingsOption == "Input font color" or settingsOption == "i":
            self.settingsInputFontColor()    # Starts up the class function to
change the input font color.
        elif settingsOption == "Top Level Domain" or settingsOption == "t":
            self.settingsTopLevelDomain()    # Starts up the class function to
change the top level domain.
        elif settingsOption == "Language" or settingsOption == "l":
            self.settingsLanguage()          # Starts up the class to change function the
```

```

language of the results.
    elif settingsOption == "Result Amount" or settingsOption == "r":
        self.settingsResultAmount()    # Starts up the class function to change
the amount of results which will appear.
    elif settingsOption == "help" or settingsOption == "h":    # A help command
for the user to use if they want to see the possible commands for this class.
        print(colored("""
Mode or m- The option to switch from basic and advanced.
Print font color or p - The option to change the print font color.
Input font color or i - The option to change the input font color.
Top Level Domain or t - The option to change the top level domain.
Language or l - The option to change the result language.
Result Amount or r - The option to change how many results you see.
help or h - The program will display this message.
back or b - The program will display the previous section.
""", helpFontColor))
        self.settings()    # Returns the user to the settings class instead of
ending the program.
    elif settingsOption == "back" or settingsOption == "b":
        self.start()    # A command to return the user to the previous class
function.
    else:
        print(colored("Error. Please input 'help' for a list of available com-
mands.", errorFontColor))
        self.settings()    # Tells the user to use 'help' and returns them to
the current class function.

    def settingsmode(self):    # Used to change the mode settings.
        modeInput = input(colored("Which mode would you like? ", self.in-
putFontColor))
        if modeInput == "basic" or modeInput == "b":    # If function to detect
the possible correct inputs and
            self.mode = 1    # execute the correct
code in response.
            self.settings()
        elif modeInput == "advanced" or modeInput == "a":
            self.mode = 2
            self.settings()
        elif modeInput == "help" or modeInput == "h":
            print(colored("""
basic or b- The program will automatically open the first link that
Google provides.
advanced or a- The program will display links equal in amount with the
value specified in Result Amount
before letting the user choose which link to open.
help or h - The program will display this message.
back or b - The program will display the previous section.
""", helpFontColor))
            self.settingsmode()
        elif modeInput == "back" or modeInput == "b":
            self.settings()
        else:
            print(colored("Error. Please input 'help' for a list of available com-
mands.", errorFontColor))
            self.settingsmode()

    def settingsPrintFontColor(self):    # Used to change the print font color.

```

```

        printFontColorInput = input(colored("What color would you like? ", self.in-
putFontColor))
        if printFontColorInput == "yellow" or printFontColorInput == "blue" or
printFontColorInput == "magenta" or printFontColorInput == "cyan" or printFontColor-
Input == "white":      # If function to detect the possible correct inputs and execute
the correct code in response.
            self.printFontColor = printFontColorInput
            self.settings()
        elif printFontColorInput == "green":
            print(colored("Green is used for help texts. Please pick another.", er-
rorFontColor))
            self.settingsPrintFontColor()
        elif printFontColorInput == "red":
            print(colored("Red is used for error texts. Please pick another.", error-
FontColor))
            self.settingsPrintFontColor()
        elif printFontColorInput == "help" or printFontColorInput == "h":
            print(colored("The available colors are:", helpFontColor))
            print(colored("yellow", "yellow"))      #Separate prints to showcase the
different possible colors.
            print(colored("blue", "blue"))
            print(colored("magenta", "magenta"))
            print(colored("cyan", "cyan"))
            print(colored("white", "white"))
            print(colored("""
Green and Red have been used for help and error texts respectively.
help or h - The program will display this message.
back or b - The program will display the previous section.
""", helpFontColor))
            self.settingsPrintFontColor()
        elif printFontColorInput == "back" or printFontColorInput == "b":
            self.settings()
        else:
            print(colored("Error. Please input 'help' for a list of available com-
mands.", errorFontColor))
            self.settingsPrintFontColor()

    def settingsInputFontColor(self):      # Essentially the same as above.
        inputFontColorInput = input(colored("What color would you like? ", self.in-
putFontColor))
        if inputFontColorInput == "yellow" or inputFontColorInput == "blue" or in-
putFontColorInput == "magenta" or inputFontColorInput == "cyan" or inputFontColorIn-
put == "white":
            self.inputFontColor = inputFontColorInput
            self.settings()
        elif inputFontColorInput == "green":
            print(colored("Green is used for help texts. Please pick another.", er-
rorFontColor))
            self.settingsPrintFontColor()
        elif inputFontColorInput == "red":
            print(colored("Red is used for error texts. Please pick another.", error-
FontColor))
            self.settingsPrintFontColor()
        elif inputFontColorInput == "help" or inputFontColorInput == "h":
            print(colored("The available colors are:", helpFontColor))
            print(colored("yellow", "yellow"))
            print(colored("blue", "blue"))

```

```

        print(colored("magenta", "magenta"))
        print(colored("cyan", "cyan"))
        print(colored("white", "white"))
        print(colored("""
Green and Red have been used for help and error texts respectively.
help or h - The program will display this message.
back or b - The program will display the previous section.
""", helpFontColor))
        self.settingsInputFontColor()
    elif inputFontColorInput == "back" or inputFontColorInput == "b":
        self.settings()
    else:
        print(colored("Error. Please input 'help' for a list of available com-
mands.", errorFontColor))
        self.settingsInputFontColor()

    def settingsTopLevelDomain(self):          # Used to change the Top Level Domain.
Same concept as above.
        topLevelDomainInput = input(colored("Which Top Level Domain would you like?
", self.inputFontColor))
        if topLevelDomainInput == "com" or topLevelDomainInput == "net" or top-
LevelDomainInput == "co.id" or topLevelDomainInput == "com.sg":
            self.topLevelDomain = topLevelDomainInput
            self.settings()
        elif topLevelDomainInput == "org" or topLevelDomainInput == "edu" or top-
LevelDomainInput == "gov" or topLevelDomainInput == "uk" or topLevelDomainInput ==
"ca" or topLevelDomainInput == "de" or topLevelDomainInput == "jp" or topLevelDomain-
Input == "fr" or topLevelDomainInput == "au" or topLevelDomainInput == "us" or top-
LevelDomainInput == "ru" or topLevelDomainInput == "ch" or topLevelDomainInput ==
"it" or topLevelDomainInput == "nl" or topLevelDomainInput == "se" or topLevelDomain-
Input == "no" or topLevelDomainInput == "es" or topLevelDomainInput == "mil":
            print(colored("Please only use the 'com', 'net', 'co.id' and 'com.sg'
TLDs. Indonesia seems to be blocking other TLDs. Once you connect to the 'co.id' TLD,
Indonesia will force your requests back into 'co.id' no matter what you input and
force the language into Indonesian.))
            self.settingsTopLevelDomain()
        elif topLevelDomainInput == "help" or topLevelDomainInput == "h":          # The
code below are Google's TLD identifiers.
            print(colored("""
The available TLDs are:
com - Commercial
org - Non-commercial
edu - US accredited post-secondary institutions
gov - United States Government
uk - United Kingdom
net - Network services
ca - Canada
de - Germany
jp - Japan
fr - France
au - Australia
us - United States
ru - Russian Federation
ch - Switzerland
it - Italy
nl - Netherlands
se - Sweden

```

```
no - Norway
es - Spain
mil - United States Military
co.id - Indonesia
com.sg - Singapore
```

Note: This is not a complete list of TLDs, there are 273 TLDs currently offered by Google so this program will only be accepting the top 20 most popular TLDs, Indonesian and Singaporean TLDs.

IMPORTANT NOTE: Indonesia also seem to be intercepting the requests causing most TLDs to fail except for 'com', 'net', 'co.id' and 'com.sg'. It will first display results in english but after using 'co.id' once, Indonesia will force the requests back to the Indonesian TLD and Indonesian language.

```
help or h - The program will display this message.
back or b - The program will display the previous section.
""" , helpFontColor))
self.settingsTopLevelDomain()
elif topLevelDomainInput == "back" or topLevelDomainInput == "b":
    self.settings()
else:
    print(colored("Error. Please input 'help' for a list of available com-
mands.", errorFontColor))
    self.settingsTopLevelDomain()

def settingsLanguage(self):    # Used to change the language of the results.
    languageInput = input(colored("Which language would you like? ", self.in-
putFontColor))
    if languageInput == "en" or languageInput == "en-GB" or languageInput == "zh-
CN" or languageInput == "zh-TW" or languageInput == "es" or languageInput == "hi" or
languageInput == "ar" or languageInput == "ms" or languageInput == "ru" or language-
Input == "bn" or languageInput == "pt-BR" or languageInput == "pt-PT" or languageIn-
put == "fr" or languageInput == "id":
        self.language = languageInput
        self.settings()
    elif languageInput == "help" or languageInput == "h":    # Similar to the
above, these are Google's language identifiers.
        print(colored("""
The available languages are:
en - English (US)
en-GB - English (UK)
zh-CN - Chinese (PRC)
zh-TW - Chinese (Taiwan)
es - Spanish
hi - Hindi
ar - Arabic
ms - Malay
ru - Russian
bn - Bengali
pt-BR - Portuguese (Brazil)
pt-PT - Portuguese (Portugal)
fr - French
```

id - Indonesian

Note: This is not a complete list of languages, there are 56 languages currently offered by Google so this program will only be accepting the top 10 most spoken languages and Indonesian language.

```
        help or h - The program will display this message.
        back or b - The program will display the previous section.
        """ , helpFontColor))
        self.settingsLanguage()
    elif languageInput == "back" or languageInput == "b":
        self.settings()
    else:
        print(colored("Error. Please input 'help' for a list of available com-
mands.", errorFontColor))
        self.settingsLanguage()

    def settingsResultAmount(self):      # Used to change the amount of results the
user gets.
        resultAmountInput = input(colored("How many results do you want to see? ",
self.inputFontColor))
        if resultAmountInput == "back" or resultAmountInput == "b":
            self.settings()
        try:
            resultAmountInput = int(resultAmountInput)
        except ValueError:      # Detects string inputs which are not 'back' or 'b'
which therefore are abnormal data inputs.
            print(colored("Input is invalid. Please only input integer values.", er-
rorFontColor))
            self.settingsResultAmount()
        except NameError:      # To prevent errors related to the googlesearch mod-
ule.
            print(colored("Input is invalid. Please only input integer values.", er-
rorFontColor))
            self.settingsResultAmount()
        else:
            if resultAmountInput < 1:      # To prevent abnormal integer inputs
which would cause an error during the search.
                print(colored("Input is invalid. Please only input integer values
above 0.", errorFontColor))
                self.settingsResultAmount()
            self.resultAmount = resultAmountInput
            self.settings()

    def start(self):      # The starting 'menu' of sorts for the program.
        startInput = input(colored("What would you like to do? ", self.in-
putFontColor))
        if startInput == "Search" or startInput == "s":
            self.search()
        elif startInput == "Settings" or startInput == "se":
            self.settings()
        elif startInput == "help" or startInput == "h":
            print(colored("""
Search or s - The program will initialize the search engine.
Settings or se - The program will initialize the setting options.
help or h - The program will display this message.
```

```

        Exit or e - The program will exit.
        """, helpFontColor))
    self.start()
    elif startInput == "Exit" or startInput == "e":
        print(colored("Thank you for using my program.", self.printFontColor))
        sys.exit()
    else:
        print(colored("Error. Please input 'help' for a list of available com-
mands.", errorFontColor))
        self.start()

    def search(self):          # Used to trigger either the basic or advanced versions
of the code.
        if self.mode == 1:
            self.basic()
        else:
            self.advanced()

    def basic(self):           # The basic program which automatically opens the first
link Google provides.
        query = input(colored("What would you like to search? ('SearchExit' to exit)
", self.inputFontColor))    # Takes the item which the user wants to search.
        if query == "SearchExit":
            self.start()
        else:
            for i in search(query, stop=1):          # Searches the item, it will first
output a generator which 'i' will convert into a link which we can actually use.
                webbrowser.open(i)                # Function used to open the link.
            self.basic()

    def advanced(self):       # The advanced program which will give the user a list of
links for them to choose from.
        query = input(colored("What would you like to search? ('SearchExit' to exit)
", self.inputFontColor))    # The same as above.
        num = 1              # Just for allowing the use of numbered list.
        lst = []             # Initializing a list to allow link selection.
        if query == "SearchExit":
            self.start()
        else:
            for i in search(query, tld=self.topLevelDomain, lang=self.language,
num=100, stop=self.resultAmount):
                print(colored(str(num) + ". " + i, self.printFontColor))          #
Printing the numbered list.
                lst.append(i)
                num = num + 1
            try:
                choice = eval(input(colored("Which link would you like to open? ",
self.inputFontColor)))
            except TypeError:
                print(colored("Input is invalid. Please only input integer values.",
errorFontColor))
                self.advanced()
            else:
                try:
                    webbrowser.open(lst[choice - 1])
                except KeyError:

```



```
        print(colored("Input is invalid. Please only input valid integers.", errorFontColor))
        self.advanced()
    else:
        self.advanced()

print("This program allows you to search google from python.\n")
m = MainFunctions()
m.start()      # Initiates the main program.
```