Converter Bot:

A Real Time Currency Converter Discord Bot

Marco Austriaco, Kris Lastrella, Margaret Nethercott, Santi Puno, Trisha Tang

CONTEXT AND RATIONALE

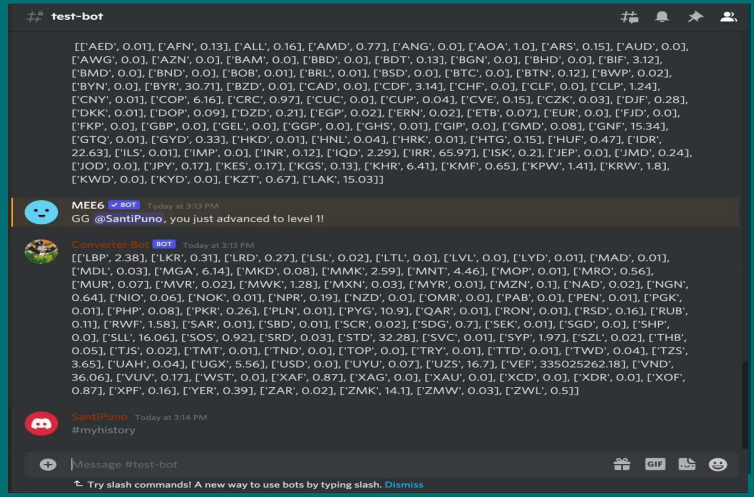
Currency rates constantly fluctuate depending on the market. Factors such as a country's global sentiments towards a specific country and its currency can cause this fluctuation. As a result, **currency rates can change at any time.** An easily accessible and universal online currency converter will be of help to travelers to inform them how their own currencies will fare when exchanged with international currencies. It will be a convenient way to quickly convert their local currency to another. Furthermore, **not all travellers know the currency of their destination**, a problem resolved by the features of this bot.

Discord is an online group-chatting platform designed for connecting different communities around the world. Over the course of quarantine, Discord grew internet traction by 86 million monthly active users from 56 million in 2019. The popular online platform features **bots that make community engagement easier** by enhancing human experience and providing different services.

With that, the group aims to create a Currency Bot that provides a fully functional currency converter for companies, investors, and governments in real time.

THE CONVERTER BOT

THE END PRODUCT: USER DEMO



Access youtube link here: https://youtu.be/xdIwARGfgwg
***If the video has issues loading, please use the link above.**

To access our replit (cloud based code):
https://replit.com/join/tcskopjsfl-margaret-sophia

DISCORD BOT COMMANDS OVERVIEW

CURRENCY CONVERSION - #CONV/#RATES

- Currency conversion based on real-time conversion rates through the #CONV function.
  - User may opt to get all conversion rates from their desired currency to all the available currencies.
  - User may specify an amount of one currency to be converted to another.

ERROR HANDLING - #HELP

- In the event that the user's input is invalid, the converter bot resolves these errors by prompting the user to call the #HELP function.
- When called, the #help function displays all the possible errors for users to troubleshoot their commands and try again.

CURRENCY IDENTIFICATION - #CURR

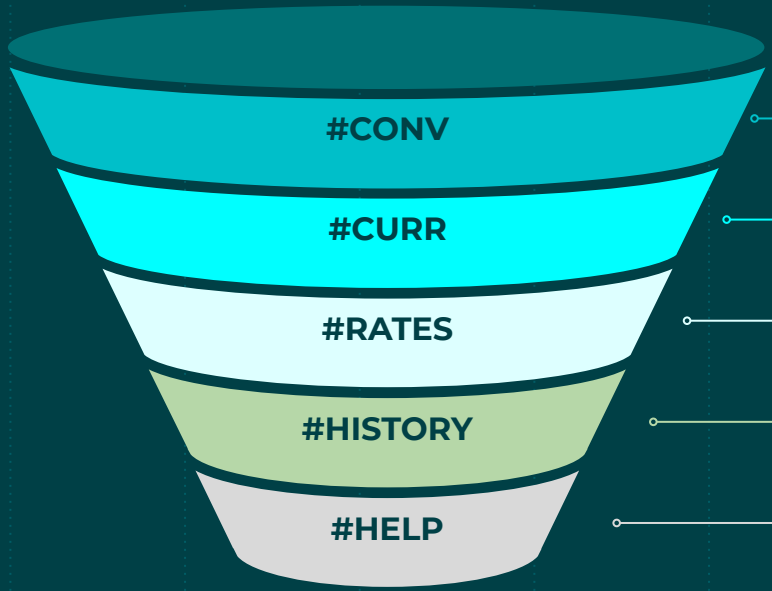
- Currency identification based on the specified country through the #CURR function.
  - The converter bot will return the currency of the user's specified country. This is useful for those who know the country they are travelling to, but do not know it's currency.

DATA COLLECTION - #HISTORY

- Converter also tracks the history of transactions of users through the #History function.
  - Users may opt to view their transaction history with the bot.
  - Administrators may access ALL transactions of ALL users in the server. Data collected here can be extracted for further analysis if needed.

THE CODE

PARTS OF THE CODE



- #CONV: Converts any amount from one currency to another
- #CURR: Obtains currency of chosen country
- #RATES: Obtains all rates of that currency
- #HISTORY: Data Collection
- #HELP: Error Handling

#CONV

CURRENCY CONVERSION

REQUESTS

```
#EXCHANGE RATES
r = requests.get
("http://api.exchangeratesapi.io/v1/latest?
access_key=c24bf5efd631f4053640225f5e2a8483")
dictionary = r.json()
exchange_rates = dictionary['rates']
valid_keys = list(exchange_rates.keys())
```

#CONV COMMAND

REQUESTS

- Through the requests module, the bot accesses a conversion rate API, which updates rates in real-time.
- This was converted to a json dictionary for the user defined functions to access data.
- The keys of the dictionary are the currencies. Ex: [PHP, USD]
- The values of the dictionary are the conversion rates from EURO (base currency of the API) to the desired currency. EX: [0.019]
- The dictionary keys were converted into a list to prevent users from requesting unavailable currency conversions.

THE CONVERSION

```
def convert_message():
    orig_money = round(float(msg[6:msg.find(' ',6)]),2)
    currency1 = msg[msg.find(' ',6)+1:msg.find("TO")-1]
    currency2 = msg[-3:]
    if currency1 in valid_keys and currency2 in valid_keys:
        euro = exchange_rates[currency1]
        conversion = round(exchange_rates[currency2]/euro,2)
        final_money = orig_money*conversion
        response = (f'\nThe current rate of {currency1} to
        {currency2} is {conversion}.\n {orig_money} {currency1} is
        {round(final_money,2)} {currency2}. Thank you!
        :money_mouth:')
        return response
    else:
        reponse = (f"I'm sorry {name}, we may not have the currency
        you are trying to convert :cry: For more information please
        call the **#HELP** function")
```

#CONV COMMAND

CONVERSION FUNCTION

[DEFINING VARIABLES]

- Variables were defined, splicing user input to access the following:
  - Amount (Ex: 500)
  - Original currency (Ex: PHP)
  - Desired currency (Ex: USD)

[CONDITIONAL]

- The function scans the list of available currencies for the original currency obtained in the user's message.
- If it is available, this is used as the key to access the conversion rate in the JSON dictionary (for both original and desired currency).

[CONVERSION]

- The API's base currency is in euros. Thus, to obtain the appropriate conversion rate, divide (EUR/CURR1) by (EUR/CURR2).
- These numbers are obtained through accessing the JSON dictionary exchange_rates.
- The resulting conversion factor is then multiplied to the money identified in the user's original message.
- This returned as "response"

USER INPUT

```
if msg[0:5].upper() == '#CONV':
  try:
    await message.channel.send(convert_message())
    if name in history:
      history[name].append(msg[6:])
    else:
      history[name] = [msg[6:]]
  except:
    await message.channel.send("Converter Bot responses using the **#CONV** function
    must be formatted as **#CONV [AMOUNT] [CUURENT CURRENCY] TO [DESIRED CURRENCY]**.
    *EX: #CONV 500 PHP TO USD*. For more information please call the function #HELP
    :cry:")
```

#CONV COMMAND

CONVERSION FUNCTION USER INPUT

- The function will not run unless #CONV is called on the discord server. A try-except is used to swiftly handle errors, and guide the user in resolving them.
- If convert_message() returns a valid response, the bot will reply with the final conversion and append this to the overall bot history:

The current rate of USD to UZS is 10656.0.
100.0 USD is 1065600.0 UZS. Thank you!

- If an error is detected (ex: unavailable currency), the user is prompted to use the #HELP command.

#CURR

CURRENCY IDENTIFICATION

CURRENCY DICTIONARY

```
#DICTIONARIES
loc = {'AFGHANISTAN': 'AFN', 'AKROTIRI AND DHEKELIA': 'EUR', 'ALAND ISLANDS': 'EUR',
'ALBANIA': 'ALL', 'ALGERIA': 'DZD', 'AMERICAN SAMOA': 'USD', 'ANDORRA': 'EUR',
'ANGOLA': 'AOA', 'ANGUILLA': 'XCD', 'ANTIGUA AND BARBUDA': 'XCD', 'ARGENTINA': 'ARS',
'ARMENIA': 'AMD', 'ARUBA': 'AWG', 'ASCENSION ISLAND': 'SHP', 'AUSTRALIA': 'AUD',
'AUSTRIA': 'EUR', 'AZERBAIJAN': 'AZN', 'BAHAMAS': 'BSD', 'BAHRAIN': 'BHD',
'BANGLADESH': 'BDT', 'BARBADOS': 'BBD', 'BELARUS': 'BYN', 'BELGIUM': 'EUR', 'BELIZE':
'BZD', 'BENIN': 'XOF', 'BERMUDA': 'BMD', 'BHUTAN': 'BTN', 'BOLIVIA': 'BOB', 'BONAIRE':
'USD', 'BOSNIA AND HERZEGOVINA': 'BAM', 'BOTSWANA': 'BWP', 'BRAZIL': 'BRL', 'BRITISH
INDIAN OCEAN TERRITORY': 'USD', 'BRITISH VIRGIN ISLANDS': 'USD', 'BRUNEI': 'BND',
'BULGARIA': 'BGN', 'BURKINA FASO': 'XOF', 'BURUNDI': 'BIF', 'CABO VERDE': 'CVE',
'CAMBODIA': 'KHR', 'CAMEROON': 'XAF', 'CANADA': 'CAD', 'CARIBBEAN NETHERLANDS': 'USD',
'CAYMAN ISLANDS': 'KYD', 'CENTRAL AFRICAN REPUBLIC': 'XAF', 'CHAD': 'XAF', 'CHATHAM
ISLANDS': 'NZD', 'CHILE': 'CLP', 'CHINA': 'CNY', 'CHRISTMAS ISLAND': 'AUD', 'COCOS
ISLANDS': 'AUD', 'COLOMBIA': 'COP', 'COMOROS': 'KMF', 'CONGO, DEMOCRATIC REPUBLIC OF
THE': 'CDF', 'CONGO, REPUBLIC OF THE': 'XAF', 'COOK ISLANDS': 'NZD', 'COSTA RICA':
'CRC', "COTE D'IVOIRE": 'XOF', 'CROATIA': 'HRK', 'CUBA': 'CUP', 'CURACAO': 'ANG',
'CYPRUS': 'EUR', 'CZECHIA': 'CZK', 'DENMARK': 'DKK', 'DJIBOUTI': 'DJF', 'DOMINICA':
'XCD', 'DOMINICAN REPUBLIC': 'DOP', 'ECUADOR': 'USD', 'EGYPT': 'EGP', 'EL SALVADOR':
'USD', 'EQUATORIAL GUINEA': 'XAF', 'ERITREA': 'ERN', 'ESTONIA': 'EUR', 'ESWATINI':
'SZL', 'ETHIOPIA': 'ETB', 'FALKLAND ISLANDS': 'FKP', 'FAROE ISLANDS': 'DKK', 'FIJI':
```

Since there's no existing API on the internet for a list of countries and their equivalent currencies, the group created a dictionary of all the countries that the converter bot has with keys as the country and their currency as the values/pair.

get_currency() function

```
def get_currency():
    country = msg[6:].upper()
    if country in list(loc.keys()):
        currentcurrency = loc[country]
        response = (f'\nThe currency of {country} is {currentcurrency}. Thank You!
        :money_mouth:')
        return response
```

- We get the country from the content of the message by getting the 6th letter and onwards since the command function of the bot is #CURR.
- Now that we have the country, the program now checks if the country that has been inputted exists in the dictionary “loc”.
- If the country exists in the dictionary, it gets the pair/value of it which is the current currency of the said country.
    - It then creates the response in which the program calls the country and its current currency formatted to a string of text to return to the user.
    - If the country does not exist in the list of dictionary keys in loc, it does nothing.

get_currency() function

```
def get_currency():
    country = msg[6:].upper()
    if country in list(loc.keys()):
        currentcurrency = loc[country]
        response = (f'\nThe currency of {country} is {currentcurrency}. Thank You!
        :money_mouth:')
        return response
```

- The command to call this function is called “#CURR” which means the user has to type “#curr” followed by the country in which the user wants to get the currency. Given that, if the code sees the keyword at the start of the message it will try to execute the function called get_currency.

- If the get_currency function fails (i.e the country does not exist in the dictionary, or entered an invalid format), it will then send a message that informs the user that the bot doesn’t carry the country’s currency or that the user has inputted a wrong format. It then shows the right format of the command.

#RATES

COLLECTS ALL CONVERSION RATES FROM CURRENT CURRENCY TO ALL AVAILABLE CURRENCIES

RATES FUNCTION

```
if msg.upper()[0:6] == "#RATES":
  rate_list = []
  currency = msg[7:10]
  if currency in valid_keys:
    for j in exchange_rates:
      euro = exchange_rates[currency]
      conversion = round(exchange_rates[j]/euro,2)
      rate_list.append([j,conversion])
    await message.channel.send(f'Hi {name} here are the current
    conversion rates for {currency} :\n\n {rate_list[0:84]}')
    await message.channel.send(rate_list[84:])
  else:
    await message.channel.send(f"Sorry {name} but we do not
    carry the currency: '{msg[7:]}' :cry: ")
```

Rates logic (Function)

- The #rates command gets all the existing rates of the desired currency.
- If the program sees “#rates” in the start of the message, the program creates an empty list called “rates_list” which would be used later.
- The user would have to input a currency – for example: PHP .
- The program would now create a new variable called currency which gets the 7-9th letter of the message and would check if the currency inputted by the user exists in the valid_keys.
- If the currency exists in the valid_keys dictionary, it would then get the currency’s equivalent rates to all countries in exchange_rates. This is done using a for loop.

Rates logic (Function)

- Since the API's base currency is in euros. Thus, to obtain the appropriate conversion rate, it must follow this formula:
  - (country currency/euro)
- These numbers are obtained through accessing the JSON dictionary exchange_rates.
- The resulting conversion factor is now the rate of a single unit of the inputted currency to a different country's currency.
- This is then created into a tuple where the format is (country, rate) and is added to a list using the append function
- Since there's a 2000 character limit for a single message on discord, the function would return 2 messages. The first message would consist of the first half of the list and the second would return the remaining half of the rates_list.
- If the inputted currency could not found on the list, the program would send a message that would inform the user that we do not carry the currency that the user gave.

#HELP

ERROR HANDLING

#HELP FUNCTION

```
if msg[0:].upper() == "#HELP":
  await message.channel.send('\nRemember the following when using Converter Bot:\n\t**#CONV**
  function :hugging: : \n\t\tThe **#CONV** function converts an amount of a currency to a specified
  currency. \n\t\t\t-Converter Bot responses must be formatted as:\n\t\t\t**#CONV [AMOUNT] [CURRENT
  CURRENCY] TO [DESIRED CURRENCY]**\n\t\t\t-*EX: #CONV 500 PHP TO USD*\n\t\t\t-You must have only
  one # at the start\n\t\t\t(Please note we do not accept the following currencies and country:
  **VES (Venezuela)**,\n\t\t\t**SSP (Sao Tome And Principe)**, **STN (South Sudan)**, and **MRU
  (Mauritania)**\n\n\t**#CURR** function:\n\t\tThe **#CURR** function identifies the currency of a
  specified country. \n\t\t\t-Converter Bot responses must be formatted as: **#CURR [COUNTRY]
  **\n\t\t\t-*EX: #CURR PHILIPPINES*\n\n\t**#MYHISTORY** function:\n\t\tThe **#MYHISTORY** function
  gives your transaction history with us.\n\t\t\t-Converter Bot responses mus be formatted as:
  **#MYHISTORY**\n\t\t\t-*EX: #MYHISTORY*\n\n\t**#RATES** function:\n\t\tThe **#RATES** function
  will give you the rates of all available countries from your currency.\n\t\t\t-Converter Bot
  responses must be formatted as: **#RATES [currency]**\n\t\t\t-*EX: #RATES PHP*')
```

The “#help” command could be seen in most commands if the program fails. This is done to show the user that the bot has a help command wherein it would send all the functions that the bot has which consists of the command keyword, what the command does, the format of the message, and an example.

Further Application

#HISTORY/MYHISTORY

CASE 1: DATA COLLECTION FOR FURTHER ANALYSIS

#MYHISTORY

```
if msg.upper() == "#MYHISTORY":
    try:
        await message.channel.send(f'Hi {name}
        here is a list of your previous
        transactions:\n{history[name]}')
    except:
        await message.channel.send(f"Sorry {name},
         you have not made any transactions with
        me yet! :cry:")
```

- The “#MYHISTORY” command checks if the user has made any previous transactions with Currency Bot.
- If the user has, the bot returns all of the past transactions of the user who inputted the command.
- If the user is a first time user of the bot, Currency Bot will declare that he/she has not made any transaction with it.

#HISTORY

```
#ADMINS LIST
admins_list = []
```

```
#PASSWORD
password = "joeisthebest"
```

DEFINING VARIABLES

- The password variable is defined by a specific string - “joeisthebest”
- The “#PASSWORD” command provides the user authorization to use the “#HISTORY” command.
- The “admin_list” variable stores a list of all the users who are admin in the server.

#HISTORY

```
if message.content.upper()[0:9] ==
"#PASSWORD":
  if message.content[10:] == password:
    admins_list.append(user_code)
    await message.channel.purge(limit=1)
    await message.channel.send("You have
    inputted the correct password. You are
    now an admin.")
  else:
    await message.channel.send('You have
    inputted an incorrect password')
```

- If the user inputs the correct password (i.e. *#PASSWORD joeisthebest*), it will add the user to the list of admins who have access to the “#HISTORY” command.
- Currency Bot will then “purge” or delete the user’s message containing the correct password to maintain the access of the “#HISTORY” command confidential.
- Afterwards, Currency Bot will give authorization to the user and declare that he/she is now an admin.
- If the user types the wrong password (e.g. *#PASSWORD hi)*, Currency Bot declares that he/she has inputted the wrong password and gives the user no access to the #HISTORY command.

#HISTORY

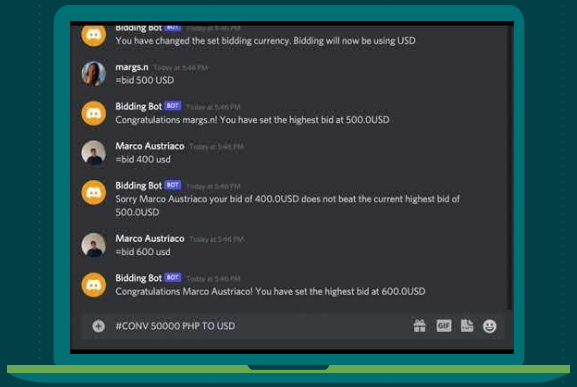
```
if msg.upper() == '#HISTORY':
  if user_code in admins_list:
    await message.channel.send(history)
  else:
    await message.channel.send(f"I'm sorry
    {name} but you do not have authorization
    to use this command. If you are an admin
    please input the password after
    **#PASSWORD**")
```

- The “#HISTORY” command records all the previous transactions made by all the users who have used Currency Bot.
- Before the bot gives the user the history, it checks if the user is included in the admin list.
- If he/she is part of the admin list, the user is able to see the history. Otherwise, Currency Bot will declare that he/she has no authorization to use the “#HISTORY” command.

INTERNATIONAL LIVE BIDDING

CASE 2: E-COMMERCE INVOLVING LIVE BIDDING

CURRENCY CONVERSION IN A LIVE BIDDING SETTING



Access youtube link here:
https://youtu.be/HP9SSweCKT8
***If the video has issues loading, please use the link above.**

- ➔ Due to COVID, transactions and bidding systems must be held online.
- ➔ If an international seller standardizes currency, consumers from different countries can use the converter bot to check if the money in their current currency can beat the previous bid.
- ➔ See the demo to the left.