ITMGT 25
FINAL PROJECT

SECTION M
GROUP 1

De Vera, Monzon, Ng

Overview

- the main purpose of the code is to help travelers decide what countries would be the best ones to travel to
- our code can help analyze the exchange rates from Philippine peso to the currency/s of the countries they had in mind of travelling

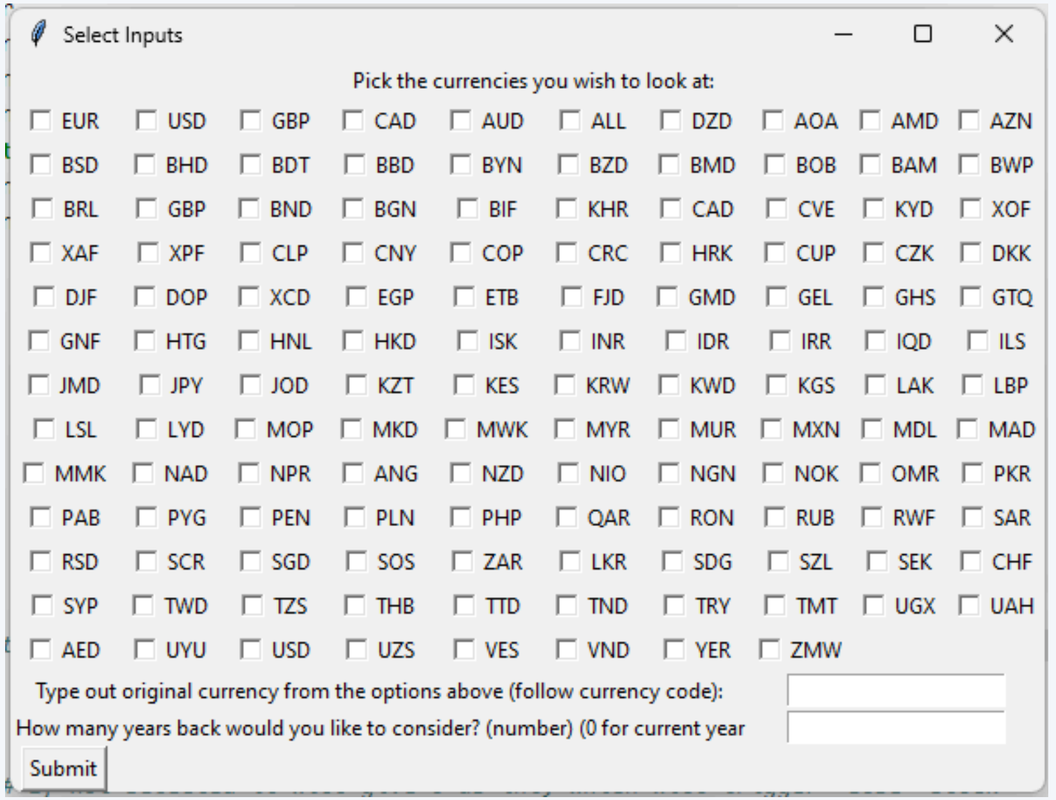
IMPORTING LIBRARIES

```
# IMPORTING ALL NEEDED LIBRARIES
import requests
import time
from selenium import webdriver
from selenium.webdriver.support.ui import Select
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver import ActionChains
from selenium.webdriver.common.actions.action_builder import ActionBuilder
from selenium.webdriver.common.actions.mouse_button import MouseButton
from bs4 import BeautifulSoup
import numpy as np
import pandas as pd
import re
from datetime import date
from tkinter import *
```

CREATING WINDOW FOR CURRENCY AND YEAR SELECTION

```
# CREATING TKINTER WINDOW FOR INPUTS
root = Tk()
root.title('Select Inputs')

# Update List function for getting selected currencies to webscrape
selected = []
def UpdateList(var,text):
    try:
        val = int(var.get()) # If not selected it will give 0 as int, which will trigger `else` block
    except ValueError:
        val = var.get()
    if val: # if val is not empty, ie, if val is any selected value
        selected.append(text)
    else: # if val is 0
        selected.remove(text) # Remove the corresponding text from the list

currencies = ["EUR", "USD", "GBP", "CAD", "AUD", "ALL", "DZD", "AOA", "AMD", "AZN", "BSD", "BHD", "BDT",
    "BBD", "BYN", "BZD", "BMD", "BOB", "BAM", "BWP", "BRL", "BND", "BGN", "BIF", "KHR",
    "CAD", "CVE", "KYD", "XOF", "XAF", "XPF", "CLP", "CNY", "COP", "CRC", "HRK", "CUP", "CZK",
    "DKK", "DJF", "DOP", "XCD", "EGP", "ETB", "FJD", "GMD", "GEL", "GHS", "GTQ", "GNF",
    "HTG", "HNL", "HKD", "ISK", "INR", "IDR", "IRR", "IQD", "ILS", "JMD", "JPY", "JOD", "KZT",
    "KES", "KRW", "KWD", "KGS", "LAK", "LBP", "LSL", "LYD", "MOP", "MKD", "MWK", "MYR", "MUR",
    "MXN", "MDL", "MAD", "MMK", "NAD", "NPR", "ANG", "NZD", "NIO", "NGN", "NOK", "OMR", "PKR",
    "PAB", "PYG", "PEN", "PLN", "PHP", "QAR", "RON", "RUB", "RWF", "SAR", "RSD", "SCR", "SGD",
    "SOS", "ZAR", "LKR", "SDG", "SZL", "SEK", "CHF", "SYP", "TWD", "TZS", "THB", "TTD", "TND",
    "TRY", "TMT", "UGX", "UAH", "AED", "UYU", "USD", "UZS", "VES", "VND", "YER", "ZMW"]

# Labels and Entries and Checboxes for: wanted currencies, original currency, and how many years back theyd want to consider
Label(root, text="Pick the currencies you wish to look at:").grid(row=0, column=1, columnspan = 8)

for idx,i in enumerate(currencies): # a for loop for making checkboxes for all currencies
    var = StringVar(value = " ")
    Checkbutton(root,text=i,variable=var,command=lambda i=i,var=var: UpdateList(var,i),onvalue=i).grid(row=(idx//10)+1,column=idx%10)

# Label(root, text="Type out original currency from the options above (follow currency code):").grid(row=15, column=0, columnspan= 7)
# from_currency_entry = Entry()
# from_currency_entry.grid(row=15, column = 7, columnspan= 3)
Label(root, text= "How many years back would you like to consider? (number) (0 for current year").grid(row=16, column=0, columnspan= 7)
how_many_years_back_entry = Entry()
how_many_years_back_entry.grid(row=16, column = 7, columnspan= 3)

# function getting the data from the text boxes on original currency and how many years back
def get_data():
    global from_currency
    global how_many_years_back
    from_currency = "PHP"
    how_many_years_back = int(how_many_years_back_entry.get())
    root.destroy()

Button(root, text="Submit", command=get_data).grid(row = 17, column=0) # submit button that triggers above function

root.mainloop() # open window
```

Output:



WEB SCRAPE EXCHANGE RATES OF SELECTED CURRENCIES

```
# getting current year
currentdate = date.today()
current_year = int(currentdate.year)

exchange_rates = {} # dictionary containing all webscraped data

# going thru all the exchange rates
for to_currency in selected:
    history = {} # dictionary containing history of exchange rate for one currency
    if to_currency != from_currency:
        for j in range(0, how_many_years_back+1): # for loop going thru each year and getting all the dates and rates from the page
            search_year = int(current_year - j)
            link = f"https://www.exchange-rates.org/exchange-rate-history/{from_currency}-{to_currency}-{search_year}"

            # getting all relevant parts of the html code
            response = requests.get(link)
            soup = BeautifulSoup(response.text, 'html.parser')
            data = soup.find_all('tr')

            # removing rows in table that are not needed
            for sflkdjasfjlk in range(0,5):
                data.pop(0)

            remove_this = ['<tr>', '<td>', '<span class="w">', '<span class="n">', '</span>', '</td>', '</tr>', '<span class="nowrap">', '<tr class="odd">', '\n']
            for i in data: # going thru all rows containing the data
                try:
                    string_i = str(i)
                    # removing tags
                    for eww in remove_this:
                        string_i = string_i.replace(eww,"")

                    # REMOVED EVERTHING EXCEPT THE <A>
                    if "</a>" in string_i:
                        string_i = string_i.replace("</a>", "")
                        string_i = string_i[string_i.index(">")+1:]
                        i_hate_this = string_i[string_i.index("<"):string_i.index(">")+1]
                        string_i = string_i.replace(i_hate_this,"")

                    # FORMAT THE RAW LINE
                    date = string_i[string_i.index(str(search_year))+4:string_i.index(str("1 "+str(from_currency)))]
                    rate = string_i[(string_i.index(date)+len(date)):string_i.index(to_currency)+len(to_currency)]

                    # appending to history dictionary
                    history[str(date)] = rate.replace("1 PHP = ", "").replace(to_currency, "")
                except: # to account for rows that dont have exchange rates in them
                    pass

    if to_currency != from_currency: # to account for when it will accidentally try to add the original currency to the exchange rates dictionary
        exchange_rates[to_currency] = history

# print(exchange_rates)
```

Output:

{'KRW': {'2024-1-1': '23.368 ', '2024-1-2': '23.586 ', '2024-1-3': '23.516 ', '2024-1-4': '23.622 ', '2024-1-5': '23.687 ', '2024-1-8': '23.513 ', '2024-1-9': '23.518 ', '2024-1-10': '23.485 ', '2024-1-11': '23.447 ', '2024-1-12': '23.537 ', '2024-1-15': '23.688 ', '2024-1-16': '23.888 ', '2024-1-17': '23.996 ', '2024-1-18': '23.994 ', '2024-1-19': '23.904 ', '2024-1-22': '23.766 ', '2024-1-23': '23.739 ', '2024-1-24': '23.714 ', '2024-1-25': '23.610 ', '2024-1-26': '23.746 ', '2024-1-29': '23.616 ', '2024-1-30': '23.564 ', '2024-1-31': '23.713 ', '2024-2-1': '23.713 ', '2024-2-2': '23.839 ', '2024-2-5': '23.668 ', '2024-2-6': '23.634 ', '2024-2-7': '23.711 ', '2024-2-8': '23.802 ', '2024-2-9': '23.822 ', '2024-2-12': '23.752 ', '2024-2-13': '23.844 ', '2024-2-14': '23.678 ', '2024-2-15': '23.777 ', '2024-2-16': '23.809 ', '2024-2-19': '23.827 ', '2024-2-20': '23.823 ', '2024-2-21': '23.843 ', '2024-2-22': '23.795 ', '2024-2-23': '23.795 ', '2024-2-26': '23.768 ', '2024-2-27': '23.739 ', '2024-2-28': '23.740 ', '2024-2-29': '23.742 ', '2024-3-1': '23.772 ', '2024-3-4': '23.756 ', '2024-3-5': '23.832 ', '2024-3-6': '23.793 ', '2024-3-7': '23.703 ', '2024-3-8': '23.719 ', '2024-3-11': '23.677 ', '2024-3-12': '23.608 ', '2024-3-13': '23.693 ', '2024-3-14': '23.823 ', '2024-3-15': '23.928 ', '2024-3-18': '24.006 ', '2024-3-19': '23.929 ', '2024-3-20': '23.717 ', '2024-3-21': '23.799 ', '2024-3-22': '23.840 ', '2024-3-25': '23.804 ', '2024-3-26': '23.853 ', '2024-3-27': '23.958 ', '2024-3-28': '23.997 ', '2024-3-29': '23.971 ', '2024-4-1': '24.070 ', '2024-4-2': '23.972 ', '2024-4-3': '23.799 ', '2024-4-4': '23.834 ', '2024-4-5': '23.878 ', '2024-4-8': '23.932 ', '2024-4-9': '23.994 ', '2024-4-10': '24.073 ', '2024-4-11': '24.206 ', '2024-4-12': '24.404 ', '2024-4-15': '24.320 ', '2024-4-16': '24.354 ', '2024-4-17': '24.108 ', '2024-4-18': '24.070 ', '2024-4-19': '23.874 ', '2024-4-22': '23.951 ', '2024-4-23': '23.878 ', '2024-4-24': '23.765 ', '2024-4-25': '23.716 ', '2024-4-26': '23.906 ', '2024-4-29': '23.846 ', '2024-4-30': '23.923 ', '2024-5-1': '23.877 ', '2024-5-2': '23.758 ', '2024-5-3': '23.764 ', '2024-5-6': '23.652 ', '2024-5-7': '23.715 ', '2024-5-8': '23.777 ', '2024-5-9': '23.825 ', '2024-5-10': '23.818 ', '2024-5-13': '23.593 ', '2024-5-14': '23.606 ', '2024-5-15': '23.515 ', '2024-5-16': '23.438 ', '2024-5-17': '23.467 ', '2024-5-20': '23.379 ', '2024-5-21': '23.444 ', '2024-5-22': '23.488 ', '2024-5-23': '23.439 ', '2024-5-24': '23.481 ', '2024-5-27': '23.390 ', '2024-5-28': '23.468 ', '2024-5-29': '23.377 ', '2024-5-30': '23.488 ', '2024-5-31': '23.636 ', '2024-6-3': '23.381 ', '2024-6-4': '23.387 ', '2024-6-5': '23.327 ', '2024-6-6': '23.269 ', '2024-6-7': '23.484 ', '2024-6-10': '23.359 ', '2024-6-11': '23.443 ', '2024-6-12': '23.418 ', '2024-6-13': '23.442 ', '2024-6-14': '23.565 ', '2024-6-17': '23.511 ', '2024-6-18': '23.500 ', '2024-6-19': '23.488 ', '2024-6-20': '23.618 ', '2024-6-21': '23.616 ', '2024-6-24': '23.554 ', '2024-6-25': '23.683 ', '2024-6-26': '23.609 ', '2024-6-27': '23.626 ', '2024-6-28': '23.648 ', '2024-7-1': '23.519 ', '2024-7-2': '23.586 ', '2024-7-3': '23.626 ', '2024-7-4': '23.582 ', '2024-7-5': '23.539 ', '2024-7-8': '23.634 ', '2024-7-9': '23.647 ', '2024-7-10': '23.738 ', '2024-7-11': '23.561 ', '2024-7-12': '23.541 '}, 'JPY': {'2024-1-1': '2.5441 ', '2024-1-2': '2.5568 ', '2024-1-3': '2.5676 ', '2024-1-4': '2.6051 ', '2024-1-5': '2.6058 ', '2024-1-8': '2.5858 ', '2024-1-9': '2.5724 ', '2024-1-10': '2.5936 ', '2024-1-11': '2.5888 ', '2024-1-12': '2.5958 ', '2024-1-15': '2.6163 ', '2024-1-16': '2.6296 ', '2024-1-17': '2.6465 ', '2024-1-18': '2.6536 ', '2024-1-19': '2.6507 ', '2024-1-22': '2.6276 ', '2024-1-23': '2.6294 ', '2024-1-24': '2.6224 ', '2024-1-25': '2.6094 ', '2024-1-26': '2.6311 ', '2024-1-29': '2.6143 ', '2024-1-30': '2.6185 ', '2024-1-31': '2.6095 ', '2024-2-1': '2.6151 ', '2024-2-2': '2.6418 ', '2024-2-5': '2.6369 ', '2024-2-6': '2.6323 ', '2024-2-7': '2.6415 ', '2024-2-8': '2.6666 ', '2024-2-9': '2.6712 ', '2024-2-12': '2.6692 ', '2024-2-13': '2.6823 ', '2024-2-14': '2.6812 ', '2024-2-15': '2.6784 ', '2024-2-16': '2.6840 ', '2024-2-19': '2.6805 ', '2024-2-20': '2.6803 ', '2024-2-21': '2.6883 ', '2024-2-22': '2.6970 ', '2024-2-23': '2.6904 ', '2024-2-26': '2.6883 ', '2024-2-27': '2.6829 ', '2024-2-28': '2.6769 ', '2024-2-29': '2.6720 ', '2024-3-1': '2.6790 ', '2024-3-4': '2.6851 ', '2024-3-5': '2.6773 ', '2024-3-6': '2.6741 ', '2024-3-7': '2.6504 ', '2024-3-8': '2.6487 ', '2024-3-11': '2.6546 ', '2024-3-12': '2.6581 ', '2024-3-13': '2.6670 ', '2024-3-14': '2.6705 ', '2024-3-15': '2.6801 ', '2024-3-18': '2.6784 ', '2024-3-19': '2.7014 ', '2024-3-20': '2.6933 ', '2024-3-21': '2.7060 ', '2024-3-22': '2.6852 ', '2024-3-25': '2.6905 ', '2024-3-26': '2.6926 ', '2024-3-27': '2.6882 ', '2024-3-28': '2.6948 ', '2024-3-29': '2.6947 ', '2024-4-1': '2.6959 ', '2024-4-2': '2.6898 ', '2024-4-3': '2.6856 ', '2024-4-4': '2.6702 ', '2024-4-5': '2.6790 ', '2024-4-8': '2.6858 ', '2024-4-9': '2.6956 ', '2024-4-10': '2.7034 ', '2024-4-11': '2.7124 ', '2024-4-12': '2.7093 ', '2024-4-15': '2.7058 ', '2024-4-16': '2.7116 ', '2024-4-17': '2.6960 ', '2024-4-18': '2.6992 ', '2024-4-19': '2.6857 ', '2024-4-22': '2.6917 ', '2024-4-23': '2.6947 ', '2024-4-24': '2.6814 ', '2024-4-25': '2.6899 ', '2024-4-26': '2.7469 ', '2024-4-29': '2.7093 ', '2024-4-30': '2.7265 ', '2024-5-1': '2.6904 ', '2024-5-2': '2.6692 ', '2024-5-3': '2.6822 ', '2024-5-6': '2.6860 ', '2024-5-7': '2.7024 ', '2024-5-8': '2.7095 ', '2024-5-9': '2.7103 ', '2024-5-10': '2.7062 ', '2024-5-13': '2.6997 ', '2024-5-14': '2.7081 ', '2024-5-15': '2.6870 ', '2024-5-16': '2.7029 ', '2024-5-17': '2.6968 ', '2024-5-20': '2.6883 ', '2024-5-21': '2.6844 ', '2024-5-22': '2.6984 ', '2024-5-23': '2.6927 ', '2024-5-24': '2.6968 ', '2024-5-27': '2.6997 ', '2024-5-28': '2.7086 ', '2024-5-29': '2.6912 ', '2024-5-30': '2.6805 ', '2024-5-31': '2.6873 ', '2024-6-3': '2.6617 ', '2024-6-4': '2.6346 ', '2024-6-5': '2.6580 ', '2024-6-6': '2.6535 ', '2024-6-7': '2.6664 ', '2024-6-10': '2.6673 ', '2024-6-11': '2.6719 ', '2024-6-12': '2.6799 ', '2024-6-13': '2.6749 ', '2024-6-14': '2.6819 ', '2024-6-17': '2.6879 ', '2024-6-18': '2.6884 ', '2024-6-19': '2.6883 ', '2024-6-20': '2.6997 ', '2024-6-21': '2.7141 ', '2024-6-24': '2.7127 ', '2024-6-25': '2.7169 ', '2024-6-26': '2.7260 ', '2024-6-27': '2.7406 ', '2024-6-28': '2.7547 ', '2024-7-1': '2.7471 ', '2024-7-2': '2.7460 ', '2024-7-3': '2.7561 ', '2024-7-4': '2.7535 ', '2024-7-5': '2.7467 ', '2024-7-8': '2.7470 ', '2024-7-9': '2.7564 ', '2024-7-10': '2.7720 ', '2024-7-11': '2.7339 ', '2024-7-12': '2.7022 '}}

CLEANING AND INTERPOLATING THE DATA

```
pd.options.display.max_rows = 9999 # display all rows
df = pd.DataFrame(exchange_rates) # create dataframe

# ARRANGING TABLE BY DATE AND MAKING THE DATE THE INDEX AS WELL
df.reset_index(inplace=True, names="date") # making date a normal column to allow for converting to standardized format
df['date'] = pd.to_datetime(df['date']) # converting to consistent format
df.sort_values(by=['date'], inplace=True, ascending=True) # arranging table by date
df = df.set_index('date') # making the date the index

# filling in missing dates
start = df.index[0].date()
end = df.index[len(df)-1].date()
new_dates = pd.date_range(start=start, end=end, freq='D') # creating range of dates from the earliest date to latest date
df = df.reindex(new_dates) # reindexing using all dates including those that were missing
df = df.rename_axis('date') # naming the index "date"

# interpolating for missing data values
for to_currency in selected:
    df[to_currency] = pd.to_numeric(df[to_currency], errors='coerce') # converting each column to a numeric insetad of object type column
df = df.interpolate()
```

Output:

date	KRW	JPY
2024-01-01	23.368000	2.544100
2024-01-02	23.586000	2.556800
2024-01-03	23.516000	2.567600
2024-01-04	23.622000	2.605100
2024-01-05	23.687000	2.605800
2024-01-06	23.629000	2.599133
2024-01-07	23.571000	2.592467
2024-01-08	23.513000	2.585800
2024-01-09	23.518000	2.572400
2024-01-10	23.485000	2.593600
2024-01-11	23.447000	2.588800
2024-01-12	23.537000	2.595800
2024-01-13	23.587333	2.602633
2024-01-14	23.637667	2.609467
2024-01-15	23.688000	2.616300
2024-01-16	23.888000	2.629600
2024-01-17	23.996000	2.646500

GETTING THE MEANS

```
from datetime import date
currentdate = date.today()
current_year = currentdate.year
current_month = currentdate.month
current_day = currentdate.day

data_dictionary_means = {} # dictionary for all important mean points

for monthshift in [1, 3, 6, 9]: # past 1, 3, 6, 9 years
    try:
        if (monthshift >= current_month) & (how_many_years_back > 0): # check if needs to go back one year when going back x months
            # adds past x months : mean of all data since date x months ago to dictionary
            data_dictionary_means['Past '+str(monthshift)+' Month(s)'] = dict(df[df.index.date > date(current_year-1, current_month-(monthshift-12), current_day-1)].mean()
            # print('Past '+str(monthshift)+' Month(s)', date(current_year-1, current_month-(monthshift-12), current_day-1))
        elif (monthshift < current_month): # same thing here, just no need to subtract 1 from the current year
            data_dictionary_means['Past '+str(monthshift)+' Month(s)'] = dict(df[df.index.date > date(current_year, current_month-monthshift, current_day-1)].mean())
            # print('Past '+str(monthshift)+' Month(s)', date(current_year, current_month-monthshift, current_day-1))
    except:
        continue

if (how_many_years_back > 0): # check if its possible to output past 365 days data
    yearshift = 1
    while yearshift <= how_many_years_back: # keep outputting past x year(s) until can no longer access older data
        data_dictionary_means['Past '+str(yearshift)+' Year(s)'] = dict(df[df.index.date > date(current_year-yearshift, current_month, current_day-1)].mean())
        # print('Past '+str(yearshift)+' Year(s)', date(current_year-yearshift, current_month, current_day-1))
        yearshift += 1

data_dictionary_means["Since "+str(start)] = dict(df.mean()) # getting the mean of all the data in the dataframe
```

Output:

```
{'Past 1 Month(s)': {'JPY': 2.7246366666666675},
 'Past 3 Month(s)': {'JPY': 2.702602197802198},
 'Past 6 Month(s)': {'JPY': 2.684642307692308},
 'Past 9 Month(s)': {'JPY': 2.6664978102189782},
 'Past 1 Year(s)': {'JPY': 2.646735245901639},
 'Past 2 Year(s)': {'JPY': 2.550864979480164},
 'Past 3 Year(s)': {'JPY': 2.466596989051095},
 'Past 4 Year(s)': {'JPY': 2.4005356605065025},
 'Past 5 Year(s)': {'JPY': 2.3447473453749317},
 'Since 2019-01-01': {'JPY': 2.322244504950495}}
```

GETTING THE MINIMUM VALUES

```
data_dictionary_mins = {} # dictionary for all important mean points

for monthshift in [1, 3, 6, 9]: # past 1, 3, 6, 9 years
    try:
        if (monthshift >= current_month) & (how_many_years_back > 0): # check if needs to go back one year when going back x months
            # adds past x months : mean of all data since date x months ago to dictionary
            data_dictionary_mins['Past '+str(monthshift)+' Month(s)'] = dict(df[df.index.date > date(current_year-1, current_month-(monthshift-12), current_day-1)].min())
            # print('Past '+str(monthshift)+' Month(s)', date(current_year-1, current_month-(monthshift-12), current_day-1))
        elif (monthshift < current_month): # same thing here, just no need to subtract 1 from the current year
            data_dictionary_mins['Past '+str(monthshift)+' Month(s)'] = dict(df[df.index.date > date(current_year, current_month-monthshift, current_day-1)].min())
            # print('Past '+str(monthshift)+' Month(s)', date(current_year, current_month-monthshift, current_day-1))
    except:
        continue

if (how_many_years_back > 0): # check if its possible to output past 365 days data
    yearshift = 1
    while yearshift <= how_many_years_back: # keep outputting past x year(s) until can no longer access older data
        data_dictionary_mins['Past '+str(yearshift)+' Year(s)'] = dict(df[df.index.date > date(current_year-yearshift, current_month, current_day-1)].min())
        # print('Past '+str(yearshift)+' Year(s)', date(current_year-yearshift, current_month, current_day-1))
        yearshift += 1

data_dictionary_mins["Since "+str(start)] = dict(df.min()) # getting the mean of all the data in the dataframe
```

Output:

```
{'Past 1 Month(s)': {'JPY': 2.6749},
 'Past 3 Month(s)': {'JPY': 2.6346},
 'Past 6 Month(s)': {'JPY': 2.6026333333333334},
 'Past 9 Month(s)': {'JPY': 2.5441},
 'Past 1 Year(s)': {'JPY': 2.5356},
 'Past 2 Year(s)': {'JPY': 2.3311},
 'Past 3 Year(s)': {'JPY': 2.1503},
 'Past 4 Year(s)': {'JPY': 2.13},
 'Past 5 Year(s)': {'JPY': 2.0103},
 'Since 2019-01-01': {'JPY': 2.0103}}
```

GETTING THE MAXIMUM VALUES

```
data_dictionary_maxs = {} # dictionary for all important mean points

for monthshift in [1, 3, 6, 9]: # past 1, 3, 6, 9 months
    try:
        if (monthshift >= current_month) & (how_many_years_back > 0): # check if needs to go back one year when going back x months
            # adds past x months : mean of all data since date x months ago to dictionary
            data_dictionary_maxs['Past '+str(monthshift)+' Month(s)'] = dict(df[df.index.date > date(current_year-1, current_month-(monthshift-12), current_day-1)].max())
            # print('Past '+str(monthshift)+' Month(s)', date(current_year-1, current_month-(monthshift-12), current_day-1))
        elif (monthshift < current_month): # same thing here, just no need to subtract 1 from the current year
            data_dictionary_maxs['Past '+str(monthshift)+' Month(s)'] = dict(df[df.index.date > date(current_year, current_month-monthshift, current_day-1)].max())
            # print('Past '+str(monthshift)+' Month(s)', date(current_year, current_month-monthshift, current_day-1))
    except:
        continue

if (how_many_years_back > 0): # check if its possible to output past 365 days data
    yearshift = 1
    while yearshift <= how_many_years_back: # keep outputting past x year(s) until can no longer access older data
        data_dictionary_maxs['Past '+str(yearshift)+' Year(s)'] = dict(df[df.index.date > date(current_year-yearshift, current_month, current_day-1)].max())
        # print('Past '+str(yearshift)+' Year(s)', date(current_year-yearshift, current_month, current_day-1))
        yearshift += 1

data_dictionary_maxs["Since "+str(start)] = dict(df.max()) # getting the mean of all the data in the dataframe
```

Output:

```
{'Past 1 Month(s)': {'JPY': 2.772},
 'Past 3 Month(s)': {'JPY': 2.772},
 'Past 6 Month(s)': {'JPY': 2.772},
 'Past 9 Month(s)': {'JPY': 2.772},
 'Past 1 Year(s)': {'JPY': 2.772},
 'Past 2 Year(s)': {'JPY': 2.772},
 'Past 3 Year(s)': {'JPY': 2.772},
 'Past 4 Year(s)': {'JPY': 2.772},
 'Past 5 Year(s)': {'JPY': 2.772},
 'Since 2019-01-01': {'JPY': 2.772}}
```

CREATING FLAGS/LABELS FOR HIGH RATES

```
for currency in selected:
    print('Most Recent Rate : 1 PHP = '+str(df.loc[df.index[-1], currency]) + ' '+currency)

# Adding a star for when the maximum last month is higher than the max rate in specified timeframe
    max_recent_month = data_dictionary_maxs['Past 1 Month(s)'][currency]
    max_recent_3_month = data_dictionary_maxs.get('Past 3 Month(s)', {}).get(currency, None)
    max_recent_6_month = data_dictionary_maxs.get('Past 6 Month(s)', {}).get(currency, None)
    max_recent_9_month = data_dictionary_maxs.get('Past 9 Month(s)', {}).get(currency, None)
    current_rate = float(df.loc[df.index[-1], currency])
    if how_many_years_back == 0:
        recent_rates = [max_recent_3_month, max_recent_6_month, max_recent_9_month]
        recent_rates = [float(rate) for rate in recent_rates if rate is not None]

        if recent_rates:
            percentage_difference = ((float(current_rate) - float(max_recent_month)) / float(max_recent_month)) * 100
            if float(max_recent_month) >= float(max(recent_rates)):
                if percentage_difference > 0:
                    print(f"⭐ | The highest rate recorded this year was seen in the last month, current rate is {percentage_difference:.2f}% higher than that.")
                else:
                    print(f"⭐ | The highest rate recorded this year was seen in the last month, current rate is {abs(percentage_difference):.2f}% lower than that.")

    if how_many_years_back == 1:
        max_last_year = df[df.index.year == (df.index[-1].year - 1)][currency].max()
        if max_last_year is not None:
            percentage_difference = ((float(current_rate) - float(max_recent_month)) / float(max_recent_month)) * 100
            if float(max_recent_month) >= float(max_last_year):
                if percentage_difference > 0:
                    print(f"⭐ | The highest rate recorded in the last month is higher than the max rate recorded last year, current is {percentage_difference:.2f}% higher than that.")
                else:
                    print(f"⭐ | The highest rate recorded in the last month is higher than the max rate recorded last year, current is {abs(percentage_difference):.2f}% lower than that.")

    if how_many_years_back >= 2:
        max_previous_years = []
        for year in range(1, how_many_years_back + 1):
            value = data_dictionary_maxs.get(f'Past {year} Year(s)', {}).get(currency, None)
            if value is not None:
                max_previous_years.append(value)

        max_previous_years = df[df.index.year < (df.index[-1].year)][currency].max()  # Get max for previous years

        if max_previous_years is not None:
            percentage_difference = ((float(current_rate) - float(max_recent_month)) / float(max_recent_month)) * 100
            if float(max_recent_month) >= float(max_previous_years):
                if percentage_difference > 0:
                    print(f"⭐ | The highest rate recorded in the last month is higher than the max rate recorded in the previous {how_many_years_back} years, excluding this year, current is {percentage_difference:.2f}% higher than this!")
                else:
                    print(f"⭐ | The highest rate recorded in the last month is higher than the max rate recorded in the previous {how_many_years_back} years, excluding this year, current is {abs(percentage_difference):.2f}% lower than this!")

# Adding a star for when the exact current rate is higher than the max rate in specified timeframe
    if how_many_years_back == 0:
        recent_rates_ver_2 = [max_recent_month, max_recent_3_month, max_recent_6_month, max_recent_9_month]
        recent_rates_ver_2 = [float(rate) for rate in recent_rates_ver_2 if rate is not None]
        if float(current_rate) >= float(max(recent_rates_ver_2)):
            print(f"⭐ | The current rate is the highest rate this year!")

    elif how_many_years_back == 1:
        if float(current_rate) >= float(max_last_year):
            print(f"⭐ | The current rate is higher the max rate last year!")

    elif how_many_years_back >= 2:
        if float(current_rate) >= float(max_previous_years):
            print(f"⭐ | The current rate is higher than the max in the last {how_many_years_back} years! (excluding current year)")
```

COMPARING CURRENT EXCHANGE RATE WITH MEANS

Code:

```
print(from_currency+' to '+currency+' mean in :')
for monthshift in [1, 3, 6, 9]:
    index = 'Past '+str(monthshift)+' Month(s)'
    try:
        percent_change = round(((df.loc[df.index[-1], currency]-data_dictionary_means[index][currency])/data_dictionary_means[index][currency])*100,2)
        if percent_change < 0:
            percent_change_str = str(percent_change*(-1)) + "% Decrease"
        else:
            percent_change_str = str(percent_change) + "% Increase"
        print(index + str(' :'), data_dictionary_means[index][currency], " ,  Current has", percent_change_str)
    except:
        continue
if (how_many_years_back > 0):
    yearshift = 1
    while yearshift <= how_many_years_back:
        index = 'Past '+str(yearshift)+' Year(s)'
        percent_change = round(((df.loc[df.index[-1], currency]-data_dictionary_means[index][currency])/data_dictionary_means[index][currency])*100,2)
        if percent_change < 0:
            percent_change_str = str(percent_change*(-1)) + "% Decrease"
        else:
            percent_change_str = str(percent_change) + "% Increase"
        print(index + str(' :'), data_dictionary_means[index][currency], " ,  Current has", percent_change_str)
        yearshift += 1
percent_change = round(((df.loc[df.index[-1], currency]-data_dictionary_means["Since "+str(start)][currency])/data_dictionary_means["Since "+str(start)][currency])*100,2)
if percent_change < 0:
    percent_change_str = str(percent_change*(-1)) + "% Decrease"
else:
    percent_change_str = str(percent_change) + "% Increase"
print("Since "+str(start) + str(' :'), data_dictionary_means["Since "+str(start)][currency], " ,  Current has", percent_change_str)
print('')
```

Output:

```
Most Recent Rate : 1 PHP = 2.6818 JPY
⭐ | The highest rate recorded in the last month is higher than the max rate recorded in the previous 2 years, excluding this year, current is 3.25% low
er than this!
PHP to JPY mean in :
Past 1 Month(s) : 2.72777  ,  Current has 1.69% Decrease
Past 3 Month(s) : 2.702382417582418  ,  Current has 0.76% Decrease
Past 6 Month(s) : 2.686862087912088  ,  Current has 0.19% Decrease
Past 9 Month(s) : 2.6676496350364967  ,  Current has 0.53% Increase
Past 1 Year(s) : 2.6488461748633876  ,  Current has 1.24% Increase
Past 2 Year(s) : 2.552532284541723 8  ,  Current has 5.06% Increase
Since 2022-01-03 : 2.5134167206040994  ,  Current has 6.7% Increase
```

COMPARING CURRENT EXCHANGE RATE WITH MAXIMUMS

Code:

```
print(from_currency+' to '+currency+' highs in :')
for monthshift in [1, 3, 6, 9]:
    try:
        index = 'Past '+str(monthshift)+' Month(s)'
        percent_change = round(((df.loc[df.index[-1], currency]-data_dictionary_maxs[index][currency])/data_dictionary_maxs[index][currency])*100,2)
        if percent_change < 0:
            percent_change_str = str(percent_change*(-1)) + "% Decrease"
        else:
            percent_change_str = str(percent_change) + "% Increase"
        print(index + str(' :'), data_dictionary_maxs[index][currency], " ,  Current has", percent_change_str)
    except:
        continue
if (how_many_years_back > 0):
    yearshift = 1
    while yearshift <= how_many_years_back:
        index = 'Past '+str(yearshift)+' Year(s)'
        percent_change = round(((df.loc[df.index[-1], currency]-data_dictionary_maxs[index][currency])/data_dictionary_maxs[index][currency])*100,2)
        if percent_change < 0:
            percent_change_str = str(percent_change*(-1)) + "% Decrease"
        else:
            percent_change_str = str(percent_change) + "% Increase"
        print(index + str(' :'), data_dictionary_maxs[index][currency], " ,  Current has", percent_change_str)
        yearshift += 1
percent_change = round(((df.loc[df.index[-1], currency]-data_dictionary_maxs["Since "+str(start)][currency])/data_dictionary_maxs["Since "+str(start)][currency])*100,2)
if percent_change < 0:
    percent_change_str = str(percent_change*(-1)) + "% Decrease"
else:
    percent_change_str = str(percent_change) + "% Increase"
print("Since "+str(start) + str(' :'), data_dictionary_maxs["Since "+str(start)][currency], " ,  Current has", percent_change_str)
print('')
```

Output:

```
PHP to JPY highs in :
Past 1 Month(s) : 2.772  ,  Current has 3.25% Decrease
Past 3 Month(s) : 2.772  ,  Current has 3.25% Decrease
Past 6 Month(s) : 2.772  ,  Current has 3.25% Decrease
Past 9 Month(s) : 2.772  ,  Current has 3.25% Decrease
Past 1 Year(s) : 2.772  ,  Current has 3.25% Decrease
Past 2 Year(s) : 2.772  ,  Current has 3.25% Decrease
Since 2022-01-03 : 2.772  ,  Current has 3.25% Decrease
```

COMPARING CURRENT EXCHANGE RATE WITH MINIMUMS

```
print(from_currency+' to '+currency+' lows in :')
for monthshift in [1, 3, 6, 9]:
    try:
        index = 'Past '+str(monthshift)+' Month(s)'
        percent_change = round(((df.loc[df.index[-1], currency]-data_dictionary_mins[index][currency])/data_dictionary_mins[index][currency])*100,2)
        if percent_change < 0:
            percent_change_str = str(percent_change*(-1)) + "% Decrease"
        else:
            percent_change_str = str(percent_change) + "% Increase"
        print(index + str(' :'), data_dictionary_mins[index][currency], " ,  Current has", percent_change_str)
    except:
        continue
if (how_many_years_back > 0):
    yearshift = 1
    while yearshift <= how_many_years_back:
        index = 'Past '+str(yearshift)+' Year(s)'
        percent_change = round(((df.loc[df.index[-1], currency]-data_dictionary_mins[index][currency])/data_dictionary_mins[index][currency])*100,2)
        if percent_change < 0:
            percent_change_str = str(percent_change*(-1)) + "% Decrease"
        else:
            percent_change_str = str(percent_change) + "% Increase"
        print(index + str(' :'), data_dictionary_mins[index][currency], " ,  Current has", percent_change_str)
        yearshift += 1
percent_change = round(((df.loc[df.index[-1], currency]-data_dictionary_mins["Since "+str(start)][currency])/data_dictionary_mins["Since "+str(start)][currency])*100,2)
if percent_change < 0:
    percent_change_str = str(percent_change*(-1)) + "% Decrease"
else:
    percent_change_str = str(percent_change) + "% Increase"
print("Since "+str(start) + str(' :'), data_dictionary_mins["Since "+str(start)][currency], " ,  Current has", percent_change_str)
print('')

--------------------------------------------------------------------
print('====================================================================================')
print('')
```

Code:

Output:

```
PHP to JPY lows in :
Past 1 Month(s) : 2.6818  ,  Current has 0.0% Increase
Past 3 Month(s) : 2.6346  ,  Current has 1.79% Increase
Past 6 Month(s) : 2.6094  ,  Current has 2.77% Increase
Past 9 Month(s) : 2.5441  ,  Current has 5.41% Increase
Past 1 Year(s) : 2.5356  ,  Current has 5.77% Increase
Past 2 Year(s) : 2.3311  ,  Current has 15.04% Increase
Since 2022-01-03 : 2.2064  ,  Current has 21.55% Increase
```

LOOKING AT MULTIPLE CURRENCIES

```
Most Recent Rate : 1 PHP = 2.6818 JPY
⭐ | The highest rate recorded in the last month is higher than the max rate recorded in the previous 2 years, excluding this year, current is 3.25% l
ower than this!
PHP to JPY mean in :
Past 1 Month(s) : 2.72777  ,  Current has 1.69% Decrease
Past 3 Month(s) : 2.702382417582418  ,  Current has 0.76% Decrease
Past 6 Month(s) : 2.686862087912088  ,  Current has 0.19% Decrease
Past 9 Month(s) : 2.667649635036496  ,  Current has 0.53% Increase
Past 1 Year(s) : 2.6488461748633876  ,  Current has 1.24% Increase
Past 2 Year(s) : 2.5525322845417238  ,  Current has 5.06% Increase
Since 2022-01-03 : 2.5134167206040994  ,  Current has 6.7% Increase

PHP to JPY highs in :
Past 1 Month(s) : 2.772  ,  Current has 3.25% Decrease
Past 3 Month(s) : 2.772  ,  Current has 3.25% Decrease
Past 6 Month(s) : 2.772  ,  Current has 3.25% Decrease
Past 9 Month(s) : 2.772  ,  Current has 3.25% Decrease
Past 1 Year(s) : 2.772  ,  Current has 3.25% Decrease
Past 2 Year(s) : 2.772  ,  Current has 3.25% Decrease
Since 2022-01-03 : 2.772  ,  Current has 3.25% Decrease

PHP to JPY lows in :
Past 1 Month(s) : 2.6818  ,  Current has 0.0% Increase
Past 3 Month(s) : 2.6346  ,  Current has 1.79% Increase
Past 6 Month(s) : 2.6094  ,  Current has 2.77% Increase
Past 9 Month(s) : 2.5441  ,  Current has 5.41% Increase
Past 1 Year(s) : 2.5356  ,  Current has 5.77% Increase
Past 2 Year(s) : 2.3311  ,  Current has 15.04% Increase
Since 2022-01-03 : 2.2064  ,  Current has 21.55% Increase

================================================================================

Most Recent Rate : 1 PHP = 23.712 KRW
PHP to KRW mean in :
Past 1 Month(s) : 23.6028  ,  Current has 0.46% Increase
Past 3 Month(s) : 23.604956043956044  ,  Current has 0.45% Increase
Past 6 Month(s) : 23.725543956043957  ,  Current has 0.06% Decrease
Past 9 Month(s) : 23.6584598540146  ,  Current has 0.23% Increase
Past 1 Year(s) : 23.62170491803279  ,  Current has 0.38% Increase
Past 2 Year(s) : 23.607413132694937  ,  Current has 0.44% Increase
Since 2022-01-03 : 23.611696871628908  ,  Current has 0.42% Increase

PHP to KRW highs in :
Past 1 Month(s) : 23.738  ,  Current has 0.11% Decrease
Past 3 Month(s) : 24.07  ,  Current has 1.49% Decrease
Past 6 Month(s) : 24.404  ,  Current has 2.84% Decrease
Past 9 Month(s) : 24.404  ,  Current has 2.84% Decrease
Past 1 Year(s) : 24.404  ,  Current has 2.84% Decrease
Past 2 Year(s) : 24.554  ,  Current has 3.43% Decrease
Since 2022-01-03 : 24.639  ,  Current has 3.76% Decrease

PHP to KRW lows in :
Past 1 Month(s) : 23.488  ,  Current has 0.95% Increase
Past 3 Month(s) : 23.269  ,  Current has 1.9% Increase
Past 6 Month(s) : 23.269  ,  Current has 1.9% Increase
Past 9 Month(s) : 23.124  ,  Current has 2.54% Increase
Past 1 Year(s) : 23.124  ,  Current has 2.54% Increase
Past 2 Year(s) : 22.511  ,  Current has 5.34% Increase
Since 2022-01-03 : 22.511  ,  Current has 5.34% Increase

================================================================================
```

THANK YOU!