Problem Definition

Our project is titled as “API based Currency Tracker”. The project uses daily data of exchange rates of all the currencies by using an API provided by a public site. The live data is then parsed and stored in SQL database. A web front end allows the user to search and present trends of currency values in graphical form on a local page.

Problem Analysis

As soon as the program is executed, there are first API calls made to the website, fixer.io, which contains all the data regarding current and past exchange rates.

An API, Application Program Interface, is basically an interface through which applications talk to each other using structured data - in this case the call is made from the python code to the website. The API defines the kind of call that is made and what exactly is being requested, how to request it, the specific data formats to be used (in this case, the JSON format, JavaScript Object Notation) and the rules to be followed.

The function "callAPI()" makes the actual API call from the code to the website. It requires a URL as an argument and uses the requests module in Python to make the actual call. It makes a request to get the currency values of all the countries in JSON format, which is then converted into Python dictionary format, returning the same nested dictionary.

However, the website offers rates with respect to only EUR and not with respect to USD (the most traded currency) and so, the convertToUSD() function takes a dictionary as an argument and returns the converted dictionary with respect to USD.

There is also another API call made for a converter between currency codes and countries as mapping them to each other is essential in order to properly receive user input in terms of the country name, as the users are not expected to remember the currency codes of the required currencies.

After the APIs, the first things that are done are calling the updateCurr31 function and the currcountries function.

The updateCurr31 function is responsible for creating, maintaining and updating a table, which contains all the currency data of the last 31 days. It checks whether the data of all 31 days is present. If not, it deletes the unwanted rows and obtains the new ones. If a required date is not present, it will obtain the relevant data by performing several API calls and entering the data into the table.

The currcountries function is responsible for the correlation between countries and their respective currency codes. It maintains a table that stores some data about each country, involving the country name, region, currency name, and currency code. It also checks whether the relevant table is already a part of the SQL database. If it already is, then it does nothing. If it is not, then it creates the required table and fills in the records.

After this, certain popular countries such as India, UK, France, etc. are chosen. Their currency data for the past 31 days will be graphically displayed as sample images for any user. However, only 4 random countries from this list will be picked, each with different colours, to be graphed and shown as samples.

Graphing of these sample images is being done with the saveGraph function, a standard graphing function which takes the name of the country and the colour and graphs the currency trend for the past 31 days. The graphing is done by the pyplot package of the matplotlib library.

Subsequently, the landing page is being prepared. A HTML file is being opened, to store the HTML script that houses the landing page. The landing page has 3 boxes, for the name, start date and end date, and a drop down menu, to select the required countries. The boxes and the drop-down menu are created using HTML code. There are also sample images, as mentioned above, of popular currencies, which are all tabulated with respect to USD.

Once the values are entered by the user and the submit button is pressed, the data is taken and the tab reloads to a new tab. Meanwhile, there are so many tasks done by the program.

Once the data is entered, it is obtained by the HTML form and sent directly to the Flask Server. These values can now be accessed and used to provide the desired outputs. Now that the values are present, potential errors are checked for, including problems with the dates, future dates, start date greater than end date, no countries selected, etc.

After all the check-steps, a success URL is created, which will be the site of the output page. The URL is one of the parts required for the log table, which keeps track of all the users of this program. The function updateLog inserts new records into the Log Table, and the log table is instrumental for showing the user his previous search as well.

Now, the main part of the program kicks off. A HTML script is started, with the last search option. It allows you to see the number of searches you have searched for, and also allow you to see your last search.

The function responsible for the majority of the program - Collecting the required data using the APIs; creating a user specific table for every unique user and inserting all the values into the required table - is the ultimateFunction. There is a specialty to the ultimate function, being that it doesn't actually use up API calls(limited to 1000 per month) unless absolutely necessary. It searches for similar data in the other tables, searching for identical columns and copying that data into the same table which is very efficient.

After inserting all the data, the next job is to get it graphed. And the function saveDateGraph does the job of taking data from tables and graphing the currency values with respect to time.

Finally, the output page is loaded with the requested currency trends in graphs and displayed to the user.

Hardware and Software requirements

- Our program requires the installation of a Python 3.7 distribution from Anaconda and the use of a MySQL database to store the data. Also requires storage space in order to store the required graphs and HTML files.
- Inside Python, both the Flask module and the pyplot package of the Matplotlib library are essential to the project. In order to call APIs, the requests module is also required. The datetime module in Python is heavily used. Other common modules such as random are also used.
- In order to connect to SQL, the mysql.connector package is used. Also, the use of MySQL workbench is suggested in order to manage the tables easily.
- On the hardware side, the recommended configuration below is ideal for the Anaconda distribution used.
    - CPU: 2 x 64-bit 2.8 GHz 8.00 GT/s CPUs.
    - RAM: 32 GB (or 16 GB of 1600 MHz DDR3 RAM)
    - Storage: 50 GB
    - Internet access to download the files from Anaconda.org and for API access

Future Enhancements

In the future, we will further customize the user experience with the use of basic AI recognition. We will analyze the patterns created by the currency values of the past 3 to 4 years and suggest expected trends based on AI techniques.

Using Python Libraries like Pandas would help us incorporate the techniques of machine learning into our project. Usage of libraries such as Numpy would also increase the efficiency of our code, to better handle bigger values and use their arrays better.

Learning JavaScript and more HTML-CSS would help us create better webpages with a lot more customization options. We could also replace the Flask application with a Django application to further open up new avenues, as Django is the leading Web Framework in Python.

For the SQL part, we will learn how to use stored procedures in MySQL to better optimize our code, reducing network traffic between Python and MySQL (which is the most time consuming part of our project).

SOURCE CODE

```
    # -*- coding: utf-8 -*-
    """
    Created on Thu Jul 23 22:51:52 2020

    @author: Anand
    """

    import requests, random, webbrowser
    from datetime import date, timedelta,datetime
    import mysql.connector as sqltor
    import matplotlib.pyplot as plt
    import os

    def convertToUSD(lst):
        lst['base'] = 'USD'
        n = lst['rates']['USD']
        for keys in lst['rates']:
            lst['rates'][keys]/=n
        lst['rates']['EUR']=1/n
        return lst

    def callAPI(url):
        r = requests.get(url).json()
        return r

    def convertDateToUnderscore(num):
        str1 = num
        str2 = ''
        for i in range(len(str1)):
            if str1[i] != '-':
                str2+=str1[i]
            else:
                str2+='_'
        return str2

    mycon = sqltor.connect(host = 'localhost',user = "root",passwd =
'Pressing.123',database = 'dbtest')
    if mycon.is_connected() == False:
        print("Error connecting to database")
    else:
        print("Successfully connected")
    cursor = mycon.cursor()

    def currcountries():
        cursor.execute('''''SELECT TABLE_NAME FROM information_schema.columns WHERE
TABLE_NAME = "currcountries"''')
        r = cursor.fetchall()
        if len(r) != 0:
```

```
            return
        counturl = "http://countryapi.gear.host/v1/Country/getCountries"
        resujson = callAPI(counturl)
        cursor.execute("create table IF NOT EXISTS currcountries(Name varchar(100) PRIMARY
KEY,CurrencyCode varchar(10), CurrencyName varchar(100),Region varchar(20))")
        mycon.commit()
        for dct in resujson['Response']:
            cursor.execute('''''insert ignore into currcountries
values("{0}","{1}","{2}","{3}")'''.format(dct["Name"],dct["CurrencyCode"],dct["CurrencyName"
],dct["Region"]))
            mycon.commit()
        cursor.execute('''''UPDATE currcountries
                        SET CurrencyCode = 'SGD',CurrencyName = 'Singapore Dollar'
                        WHERE Name = 'Singapore' ''')
        cursor.execute('''''UPDATE currcountries
                        SET CurrencyCode = 'IMP',CurrencyName = 'Isle of Man Pound'
                        WHERE Name = 'Isle of Man' ''')
        cursor.execute('''''UPDATE currcountries
                        SET CurrencyCode = 'JEP',CurrencyName = 'Jersey Pound'
                        WHERE Name = 'Jersey' ''')
        cursor.execute("insert ignore into currcountries
values('Bitcoin','BTC','Bitcoin','Cryptocurrency')")
        cursor.execute("insert ignore into currcountries values('Chilean Unit of
Account','CLF','Chilean Unit of Account','Americas')")
        cursor.execute("insert ignore into currcountries values('Cuban Peso','CUP','Cuban
Peso','Americas')")
        cursor.execute("insert ignore into currcountries values('Silver','XAG','Silver
Ounce','Metal')")
        cursor.execute("insert ignore into currcountries values('Gold','XAU','Gold
Ounce','Metal')")
        cursor.execute("DELETE FROM currcountries WHERE Name = 'South Sudan'")
        mycon.commit()

    def getDictionaryLatest():
        d = {}
        cursor.execute('''''SELECT * from currvalues''')
        data = cursor.fetchall()
        for i in range(len(data)):
            d[data[i][0]] = data[i][1]
        mycon.commit()
        return d
    d = getDictionaryLatest()

    def getCurrCode(country):
        country = country.capitalize()
        cursor.execute("select CurrencyCode from currcountries where Name like
'{0}'".format(country))
        code = cursor.fetchone()[0]
        return code

    def getCurrCodeLatest(country):
        country = country.capitalize()
        cursor.execute("select CurrencyCode from currcountries where Name like
'{0}'".format(country))
        code = cursor.fetchone()[0]
        cursor.execute("select CurrFactorLatest from currvalues where CurrCode =
'{0}'".format(code))
        value = cursor.fetchone()[0]
        return value

    def listOfDays(startdate,enddate = date.today(),gap = 1,limit = 500): #Returns a list
of days from specified start dates to end dates, both inclusive
```

```
        delta1 = enddate - startdate
        list1 = []
        if delta1.days//gap >limit:
            print("No of days is too large, not supported by this function")
            return []
        start_date = startdate
        end_date = enddate
        delta = timedelta(days=gap)
        while start_date <= end_date:
            list1.append(start_date.strftime("%Y-%m-%d"))
            start_date += delta
        return list1

    def daysListFromDate(limit, enddate = date.today()): #Returns the list of days upto
'limit' no of days from specified end date, total limit + 1 elements
        if type(limit) != int:
            print("The argument should be of int datatype")
            return []
        if limit > 50:
            print("The no of values is too high")
            return []
        list1 = []
        for i in range(limit + 1):
            list1.append((enddate - timedelta(days=i)).strftime("%Y-%m-%d"))
        list1.reverse()
        return list1

    def datetimeToString(date):
        datestr = date.strftime("%Y-%m-%d")
        return datestr

    def stringToDatetime(date_str):
        format_str = '%d-%m-%Y'
        datetime_obj = datetime.strptime(date_str, format_str)
        return datetime_obj.date()

    def saveGraph(country1,color):
        uname = 'Curr31'
        cursor.execute("select CurrencyCode from currcountries where Name like
'{0}'".format(country1))
        currcode1 = cursor.fetchone()[0]
        PostgreSQL_select_Query = '''''select column_name from information_schema.columns
where table_name = '{0}' '''.format(uname)
        cursor.execute(PostgreSQL_select_Query)
        records = cursor.fetchall()
        tup = ('CurrCode',)
        for i in records:
            if i == tup:
                records.remove(i)
        d1 = {}
        for i in range(len(records)):
            cursor.execute('''''SELECT {2} from {0} WHERE CurrCode in
('{1}')'''.format(uname,currcode1,records[i][0]))
            data = cursor.fetchall()
            rec = records[i][0]
            d1[rec] = data[0][0]

        x1 = list(d1.keys())
        y1 = list(d1.values())
        plt.style.use(['dark_background'])
        plt.rcParams["figure.figsize"] = (10,5)
        plt.plot(x1,y1,color = color)
```

```
        plt.xlabel('Date(past month)')
        plt.xticks(x1,rotation = 90)
        plt.ylabel('{0} wrt USD'.format(currcode1))
        plt.title(country1 + " (for the past 31 days)")
        loc = "C:\\Users\\Anand\\Coding\\Images\\{0}-{1}-2-
sample.png".format(country1,date.today())
        if os.path.isfile(loc):
            os.remove(loc)
        plt.savefig(loc,bbox_inches = 'tight', dpi = 300)
        plt.clf()
        plt.close()
        return "file:///C:/Users/Anand/Coding/Images/{0}-{1}-2-
sample.png" .format(country1,date.today())


    def createTable(uname,startdate,enddate):
        startdate = stringToDatetime(startdate)
        enddate = stringToDatetime(enddate)
        dayslst2 = listOfDays(startdate,enddate)
        cursor.execute('''''create table IF NOT EXISTS {0}(CurrCode varchar(10) PRIMARY
KEY)'''.format(uname))
        mycon.commit()
        startdate1 = startdate.strftime("%Y-%m-%d")
        starturl = 'http://data.fixer.io/api/'+ startdate1
+'?access_key=104d12bf481223711dd4e2e0cf6eaac0'
        startvalues = convertToUSD(callAPI(starturl))
        for key in startvalues['rates']:
            cursor.execute('''''insert ignore into {1}(CurrCode)
values("{0}")'''.format(key,uname))
            mycon.commit()
        for i in range(len(dayslst2)):
            currhisturl = 'http://data.fixer.io/api/' + dayslst2[i]
+'?access_key=104d12bf481223711dd4e2e0cf6eaac0'
            histvalues = callAPI(currhisturl)
            histvaluesrates = convertToUSD(histvalues)
            unddate = convertDateToUnderscore(dayslst2[i])
            try:
                query = '''''ALTER TABLE {1}
                        ADD ({0} float)'''
                cursor.execute(query.format(unddate,uname))
                mycon.commit()
                for key in histvaluesrates['rates']:
                    query1 = '''''UPDATE {3}
                           SET {2} = "{1}"
                           WHERE CurrCode = "{0}"'''

cursor.execute(query1.format(key,histvaluesrates['rates'][key],unddate,uname))
                    mycon.commit()
            except:
                for key in histvaluesrates['rates']:
                    query1 = '''''UPDATE {3}
                           SET {2} = "{1}"
                           WHERE CurrCode = "{0}"'''

cursor.execute(query1.format(key,histvaluesrates['rates'][key],unddate,uname))
                    mycon.commit()

    def ultimateFunction(uname,startdate,enddate):
        dayslst2 = listOfDays(stringToDatetime(startdate),stringToDatetime(enddate))
        cursor.execute('''''create table IF NOT EXISTS {0}(CurrCode varchar(10) PRIMARY
KEY)'''.format(uname))
        mycon.commit()
```

```
        cursor.execute("SELECT Currcode from currvalues")
        rc1 = cursor.fetchall()
        for i in range(len(rc1)):
            cursor.execute('''''insert ignore into {1}(CurrCode)
values("{0}")'''.format(rc1[i][0],uname))
        cursor.execute("select column_name from information_schema.columns where table_name
= '{0}'".format(uname))
        data = cursor.fetchall()
        for i in range(len(dayslst2)):
            unddate = convertDateToUnderscore(dayslst2[i])
            tup = (unddate,)
            cursor.execute("select table_name,column_name from information_schema.columns
where column_name = '{0}'".format(unddate))
            data1 = cursor.fetchall()
            dict1 = {}
            for i in range(len(data1)):
                dict1[data1[i][0]] = data1[i][1]
            if unddate in dict1.values():
                if tup not in data:
                    tabname = list(dict1.keys())[0]
                    query = '''''ALTER TABLE {0}
                    ADD ({1} float DEFAULT 0.0)'''.format(uname,unddate)
                    cursor.execute(query)
                    mycon.commit()
                    query1 = '''''UPDATE {0}
                    INNER JOIN {1} ON {0}.CurrCode = {1}.CurrCode
                    SET {0}.{2} = {1}.{2}'''.format(uname,tabname,unddate)
                    cursor.execute(query1)
                    mycon.commit()
            if unddate not in dict1.values():
                j = 0
                currhisturl = 'http://data.fixer.io/api/' + dayslst2[i]
+'?access_key=104d12bf481223711dd4e2e0cf6eaac0'
                histvaluesrates = convertToUSD(callAPI(currhisturl))
                if j == 0:
                    for key in histvaluesrates['rates']:
                        cursor.execute('''''insert ignore into {1}(CurrCode)
values("{0}")'''.format(key,uname))
                        mycon.commit()
                j+=1
                query = '''''ALTER TABLE {1}
                ADD ({0} float DEFAULT 0.00)'''
                cursor.execute(query.format(unddate,uname))
                mycon.commit()
                for key in histvaluesrates['rates']:
                    query1 = '''''UPDATE {3}
                    SET {2} = "{1}"
                    WHERE CurrCode = "{0}"'''

cursor.execute(query1.format(key,histvaluesrates['rates'][key],unddate,uname))
                    mycon.commit()

    def saveDateGraph(uname,startdate,enddate,country1,color):
        dayslst2 = listOfDays(stringToDatetime(startdate),stringToDatetime(enddate))
        cursor.execute("select CurrencyCode from currcountries where Name like
'{0}'".format(country1))
        currcode1 = cursor.fetchone()[0]
        d1 = {}
        for i in range(len(dayslst2)):
            unddate = convertDateToUnderscore(dayslst2[i])
            cursor.execute('''''SELECT {2} from {0} WHERE CurrCode in
('{1}')'''.format(uname,currcode1,unddate))
            data = cursor.fetchall()
```

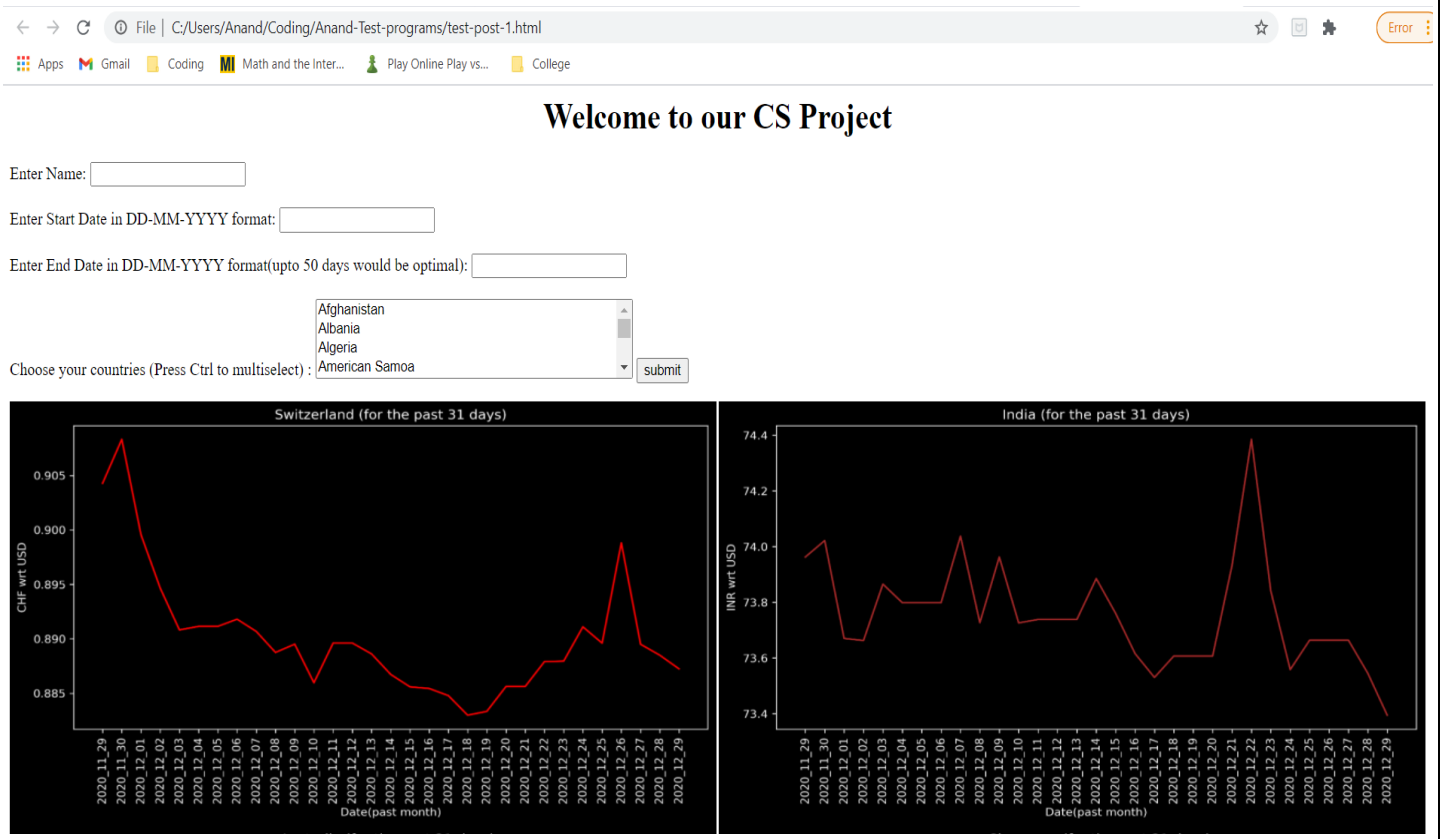
```
            d1[unddate] = data[0][0]

        x1 = list(d1.keys())
        y1 = list(d1.values())
        plt.style.use(['dark_background'])
        plt.rcParams["figure.figsize"] = (10,5)
        plt.plot(x1,y1,color = color)
        plt.xlabel('Date')
        plt.xticks(x1,rotation = 90)
        plt.ylabel('{0} wrt USD'.format(currcode1))
        plt.title(country1 + " from {0} to {1} ".format(startdate,enddate))
        nowdate = date.today()
        nowtime = datetime.now().strftime("%H-%M-%S")
        loc = "C:\\Users\\Anand\\Coding\\Anand-Test-
programs\\static\\{1}_{0}_{2}_{3}.png".format(country1,uname,nowdate,nowtime)
        if os.path.isfile(loc):
            os.remove(loc)
        plt.savefig(loc,bbox_inches = 'tight', dpi = 300)
        plt.clf()
        return nowtime
#"/static/{1}_{0}_{2}_{3}.png".format(country1,uname,nowdate,nowtime)


    def updateCurr31():
        lst3 = daysListFromDate(30)
        lst4 = []
        for i in range(len(lst3)):
            undname30 = convertDateToUnderscore(lst3[i])
            lst4.append(undname30)
        cursor.execute("select column_name from information_schema.columns where table_name
= 'Curr31'")
        d = cursor.fetchall()
        for i in range(len(d)-1):
            if d[i][0] not in lst4 and d[i][0] != 'CurrCode':
                query = '''''ALTER TABLE Curr31
                    DROP {0}'''
                cursor.execute(query.format(d[i][0]))
                mycon.commit()
        for i in range(len(lst3)):
            undname31 = convertDateToUnderscore(lst3[i])
            tup = (undname31,)
            if tup not in d:
                url31 = 'http://data.fixer.io/api/'+ lst3[i]
+'?access_key=104d12bf481223711dd4e2e0cf6eaac0'
                values31 = convertToUSD(callAPI(url31))
                query = '''''ALTER TABLE Curr31
                    ADD ({0} float)'''
                cursor.execute(query.format(undname31))
                mycon.commit()
                for key in values31['rates']:
                    query1 = '''''UPDATE Curr31
                       SET {2} = "{1}"
                       WHERE CurrCode = "{0}"'''
                    cursor.execute(query1.format(key,values31['rates'][key],undname31))
                    mycon.commit()
    def updateLog(name1,stdate1,endate1,csv1,url1):
        cursor.execute('''''create table IF NOT EXISTS
                       Log(sno integer PRIMARY KEY,name varchar(100) NOT NULL,count integer
NOT NULL,
                           logindatetime datetime NOT NULL,startdate date NOT NULL,enddate
date NOT NULL,
                           countries varchar(1000) NOT NULL,url varchar(1000) NOT NULL)''')
```

```
        mycon.commit()
        cursor.execute("select count(sno) from Log")
        snocount = cursor.fetchone()
        sno = snocount[0] + 1
        stdate1 = datetime.strptime(stdate1, "%d-%m-%Y").strftime("%Y-%m-%d")
        endate1 = datetime.strptime(endate1, "%d-%m-%Y").strftime("%Y-%m-%d")
        cursor.execute("SELECT COUNT(sno) FROM Log WHERE name = '{0}'".format(name1))
        namecount = cursor.fetchone()
        count1 = namecount[0] + 1
        logdate1 = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        cursor.execute('''''insert into Log
values("{0}","{1}","{2}","{3}","{4}","{5}","{6}","{7}")'''.format(sno,name1,count1,logdate1,
stdate1,endate1,csv1,url1))


    updateCurr31()
    currcountries()

    uk = "United Kingdom of Great Britain and Northern Ireland"
    commonlist =
["India",uk,"Japan","China","France","Switzerland","Singapore","Australia","Canada","Bitcoin
","Gold","Silver"]
    colorlist =
['blue','red','green','yellow','orange','cyan','purple','magenta','brown','white']
    rlist = random.sample(commonlist,4)
    slist = random.sample(colorlist,4)
    srcode1 = saveGraph(rlist[0],slist[0])
    srcode2 = saveGraph(rlist[1],slist[1])
    srcode3 = saveGraph(rlist[2],slist[2])
    srcode4 = saveGraph(rlist[3],slist[3])


    f = open('test-post-1.html','w+')

    cursor.execute("select Name from currcountries ")
    code = cursor.fetchall()
    message = """<html>
    <head>
        <title> CS Project </title>
    </head>
    <body>
        <form action = "http://localhost:5000/login" method = "POST">
            <center><h1>Welcome to our CS Project</h1></center>
            <label for="nm">Enter Name:</label>
            <input type = "text" name = "nm" id = "nm" /><br><br>
            <label for="date1">Enter Start Date in DD-MM-YYYY format:</label>
            <input type = "text" name = "date1" id = "date1" /><br><br>
            <label for="date2">Enter End Date in DD-MM-YYYY format(upto 50 days would be
optimal):</label>
            <input type = "text" name = "date2" id = "date2" /><br><br>
            <label for="currency1">Choose your countries (Press Ctrl to multiselect) :
</label>
            <select name="currency1" id="currency1" multiple>"""
    for i in range(len(code)):
        if i !=1:
             message = message + """<option value =
"{0}" >{0}</option>""".format(code[i][0])

    message = message + """</select>
            <input type = "submit" value = "submit" />
        </form>
```

```
        <img align = "left" src = "file:///C:/Users/Anand/Coding/Images/{0}-{4}-2-
sample.png" alt = "Sample image" width = "750" height = "375">
        <img src = "file:///C:/Users/Anand/Coding/Images/{1}-{4}-2-sample.png" alt =
"Sample image" width = "750" height = "375">
        <img align = "left" src = "file:///C:/Users/Anand/Coding/Images/{2}-{4}-2-
sample.png" alt = "Sample image" width = "750" height = "375">
        <img src = "file:///C:/Users/Anand/Coding/Images/{3}-{4}-2-sample.png" alt =
"Sample image" width = "750" height = "375">

    </body>
    </html>""".format(rlist[0],rlist[1],rlist[2],rlist[3],date.today())

    f.write(message)
    f.close()

    webbrowser.open_new_tab('test-post-1.html')









    from flask import Flask, redirect, url_for, request
    app = Flask(__name__)

    @app.route('/success/<name>/<sdate>/<edate>/<curr1>/<count>')
    def success(name,sdate,edate,curr1,count):
        count = int(count)
        requrl =
"http://localhost:5000/success/{0}/{1}/{2}/{3}/{4}".format(name,sdate,edate,curr1,count)
        cursor.execute("select url from log where name = '{0}'".format(name))
        reqtup = (requrl,)
        histurl1 = cursor.fetchall()
        index1 = count - 1
        curr1 = curr1.split(",")
        colorlst =
['blue','red','green','yellow','cyan','orange','purple','brown','magenta','white','pink','ol
ive','gold']
        n1 = len(colorlst)
        n = len(curr1)
        if n1>=n:
            randcolorlst = random.sample(colorlst,n)
        else:
            randcolorlst = random.sample(colorlst,n1)*(n//n1 + 1)
        ultimateFunction(name, sdate, edate)
        message1 = '''''<html>
                        <head>
                            <title> Success!!! </title>
                        </head>
                        <body>
                            <h1><center>Hello {0}</center></h1>'''.format(name)
        if index1>0:
            histurl = histurl1[index1-1][0]
            message1+= '''''<h3><center>You have made {1} no of valid searches so
far<br><br> <a href="{0}" target = "_blank">Your last search
is</a></center></h3>'''.format(histurl,count)
        elif index1 == 0:
            message1+='''''<h3><center>This is your first search </center></h3>'''
        #for i in range(n):
            #message1+='''<h3>The value of {1} is {0}
<br></h3>'''.format(getCurrCodeLatest(curr1[i]),getCurrCode(curr1[i]))
```

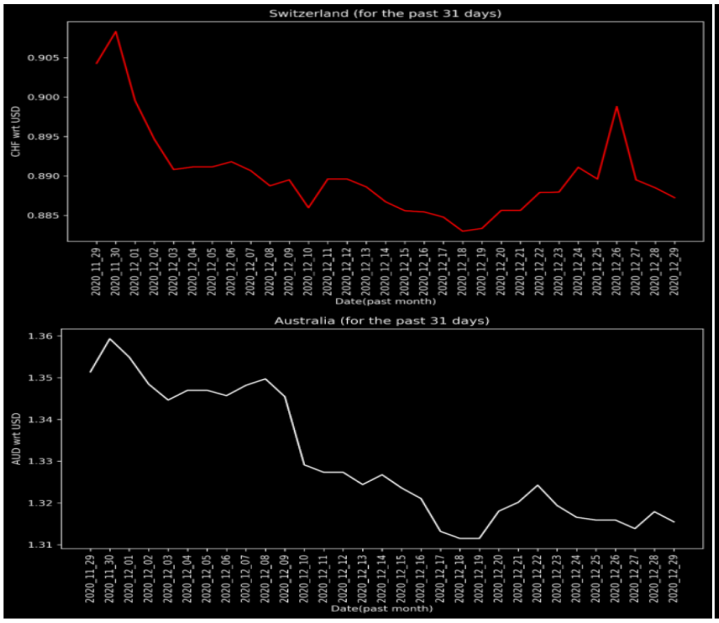
```
        for i in range(n):
            rn = randcolorlst[i]
            src1 = saveDateGraph(name, sdate, edate, curr1[i], rn)
            #message1+='''<img src = {0} alt = "Sample image" width = "750" height =
"375"/>'''.format(src1)
            message1+='''''<img src = "/static/{1}_{0}_{2}_{3}.png" alt = {0} width = "750"
height = "375"/>'''.format(curr1[i],name,date.today(),src1)
            #message1+='''<img src = "file:///C:/Users/Anand/Coding/Images/{1}_{0}_{2}.png"
alt = {0} '''.format(curr1[i],name,date.today())
        message1+='''''</body>
                  </html>'''
        return message1

    @app.route('/error/<name>/<details>')
    def error(name,details):
        message2 = '''''<html>
        <head>
        <title> Error!!! </title>
        </head>
        <body>
        <h1><center>Hello {0}<br></center></h1>
        <h2>Sorry {0}, you have an error with the values inputted<br><br></h2>
        <h3>It seems to be related to "{1}"'''.format(name,details)
        return message2


    @app.route('/login',methods = ['POST', 'GET'])
    def login():
        if request.method == 'POST':
            user = request.form.get('nm')
            date1 = request.form.get('date1')
            date2 = request.form.get('date2')
            try:
                dt1 = stringToDatetime(date1)
                dt2 = stringToDatetime(date2)
            except:
                return redirect(url_for('error',name = user, details = 'Invalid dates,
either format or date itself'))
            if dt1>date.today() or dt2>date.today():
                return redirect(url_for('error',name = user, details = 'Future dates'))
            if dt1>dt2:
                return redirect(url_for('error',name = user, details = 'Start date greater
than End date'))
            cur1 = request.form.getlist('currency1')
            if len(cur1) == 0:
                return redirect(url_for('error',name = user,details = 'No countries
selected'))
            cur1_csv = ",".join(cur1)
            cursor.execute("SELECT COUNT(sno) FROM Log WHERE name = '{0}'".format(user))
            namecount = cursor.fetchone()
            count1 = namecount[0] + 1
            requrl =
"http://localhost:5000/success/{0}/{1}/{2}/{3}/{4}".format(user,date1,date2,cur1_csv,count1)
            updateLog(user,date1,date2,cur1_csv,requrl)
            return redirect(url_for('success',name = user,sdate = date1,edate = date2,curr1
= cur1_csv, count = count1))

      if __name__ == '__main__':
        app.run(debug = True)

    mycon.close()
```

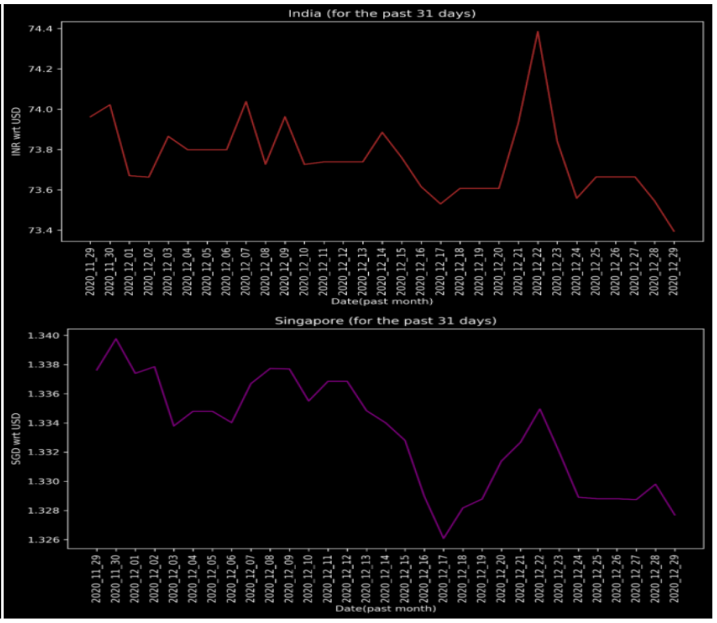
OUTPUTS


File | C:/Users/Anand/Coding/Anand-Test-programs/test-post-1.html
Apps
Gmail
Coding
Math and the Inter...
Play Online Play vs...
College
Error
Welcome to our CS Project
Enter Name:
Enter Start Date in DD-MM-YYYY format:
Enter End Date in DD-MM-YYYY format(upto 50 days would be optimal):
Choose your countries (Press Ctrl to multiselect) :
Afghanistan
Albania
Algeria
American Samoa
submit

Switzerland (for the past 31 days)
CHF wrt USD
Date(past month)

India (for the past 31 days)
INR wrt USD
Date(past month)


Switzerland (for the past 31 days)
CHF wrt USD
Date(past month)


India (for the past 31 days)
INR wrt USD
Date(past month)

Australia (for the past 31 days)
AUD wrt USD
Date(past month)

Singapore (for the past 31 days)
SGD wrt USD
Date(past month)

File | C:/Users/Anand/Coding/Anand-Test-programs/test-post-1.html

Apps Gmail Coding Math and the Inter... Play Online Play vs... College

Welcome to our CS Project

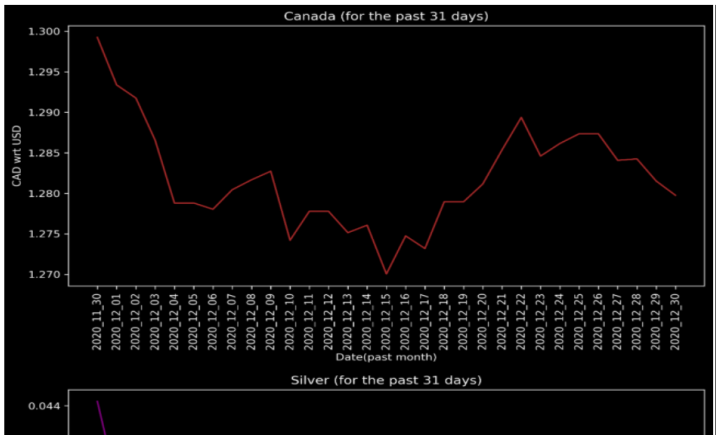
Enter Name: test

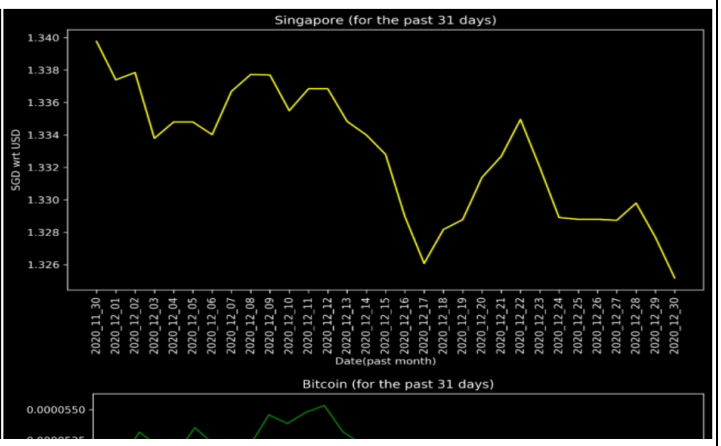
Enter Start Date in DD-MM-YYYY format: 15-11-2020

Enter End Date in DD-MM-YYYY format(upto 50 days would be optimal): 29-12-2020

Choose your countries (Press Ctrl to multiselect) : Afghanistan, Albania, Algeria, American Samoa

submit



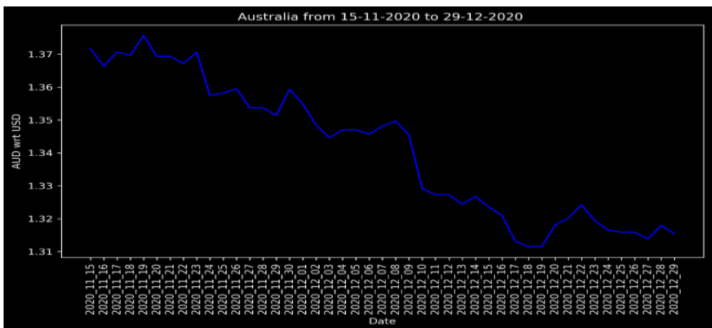


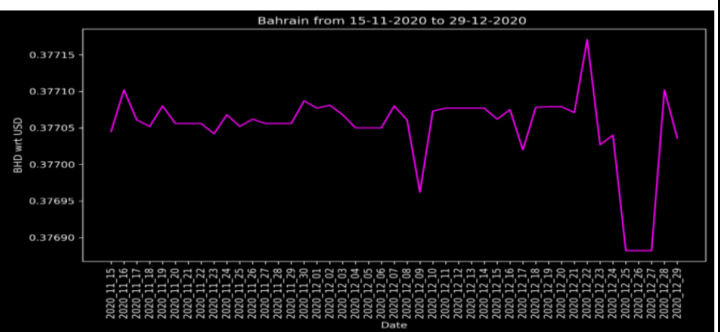
localhost:5000/success/test/15-11-2020/29-12-2020/Australia%2CBahrain%2CBitcoin%2CBrazil%2CCanada%2CChina%2CFrance%2CGold%2CIndia/1

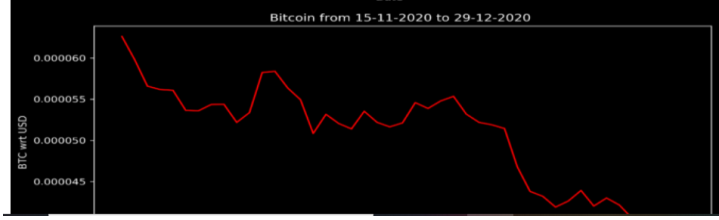
Apps Gmail Coding Math and the Inter... Play Online Play vs... College

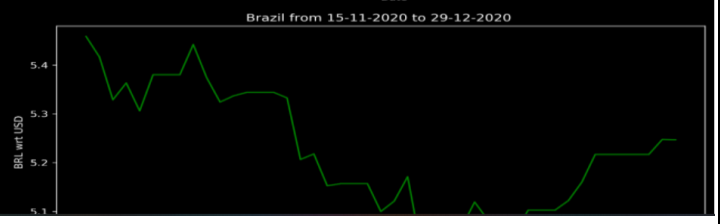
Hello test

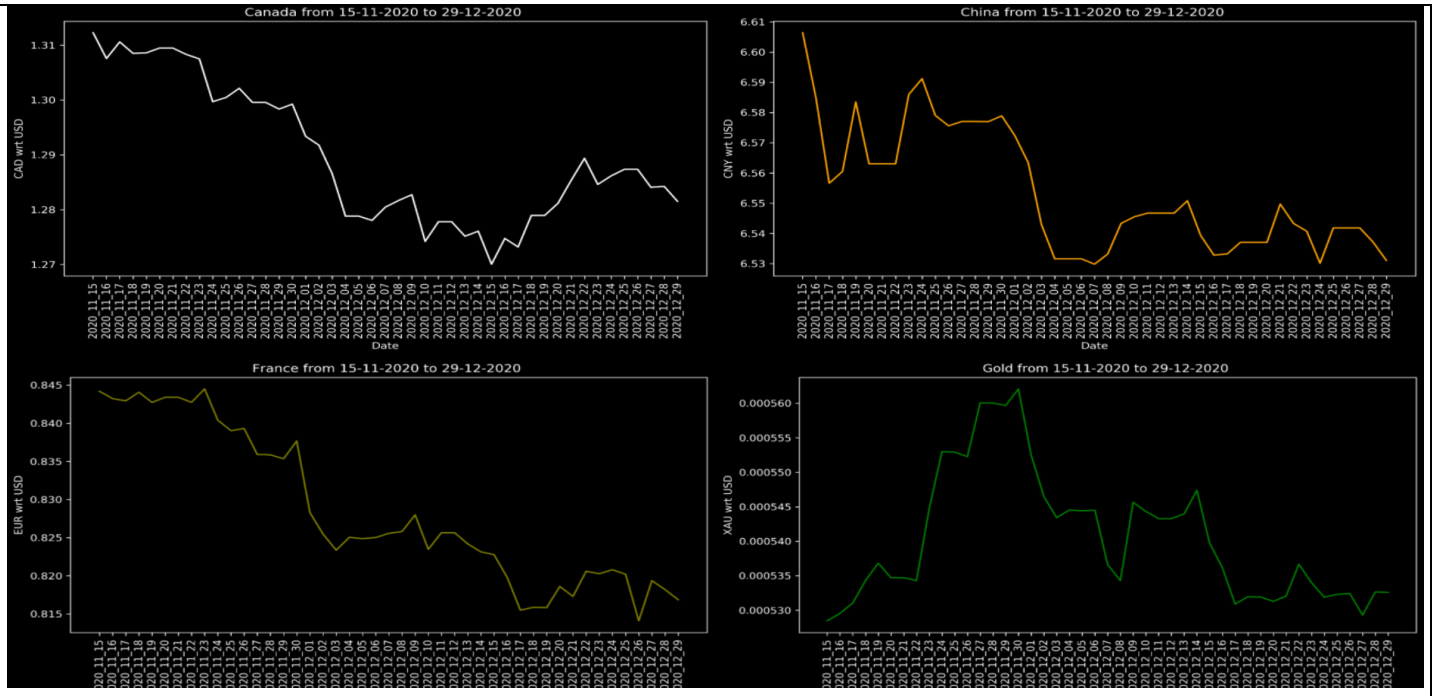
This is your first search

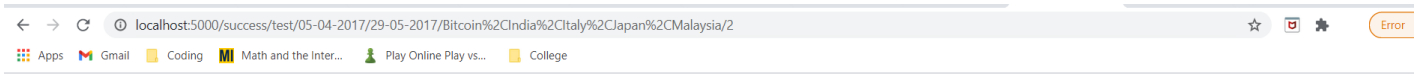








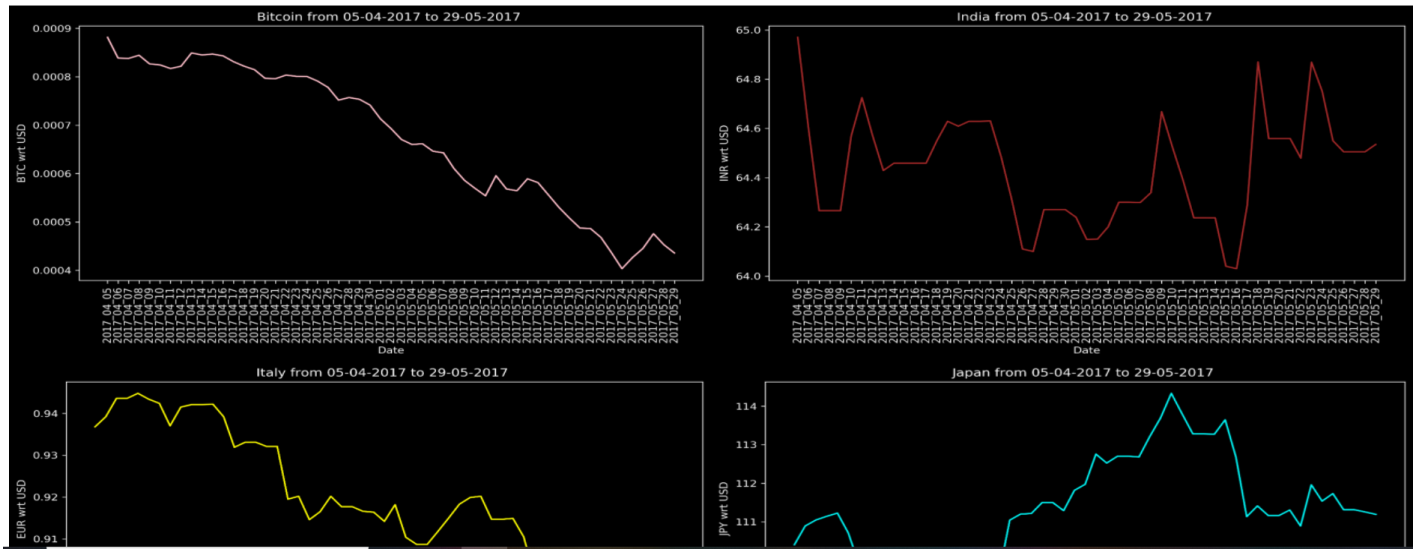




Hello test

You have made 2 no of valid searches so far

Your last search is



File | C:/Users/Anand/Coding/Anand-Test-programs/test-post-1.html

Apps Gmail Coding Math and the Inter... Play Online Play vs... College

Welcome to our CS Project

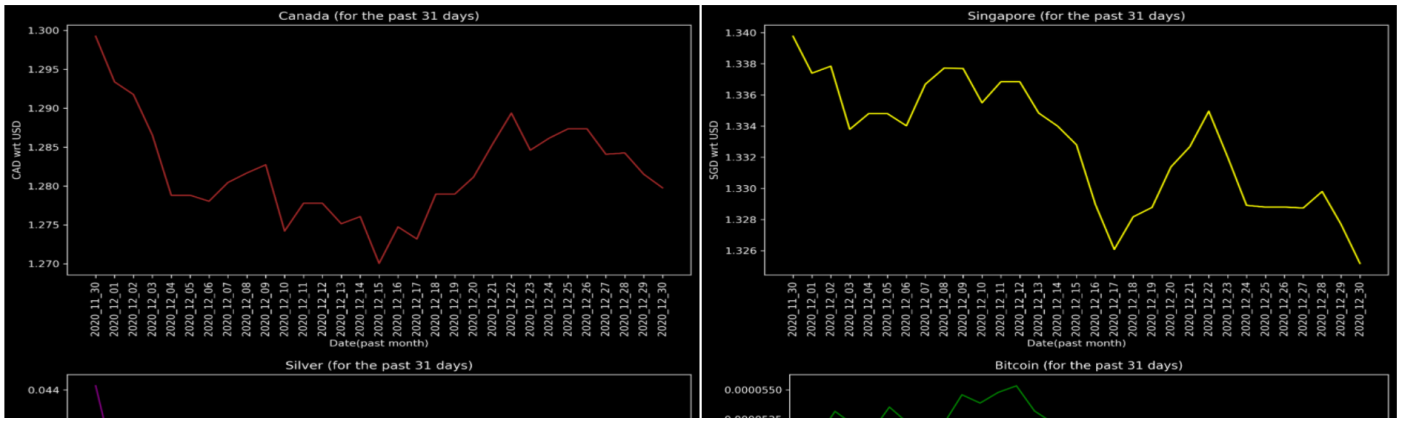
Enter Name: test

Enter Start Date in DD-MM-YYYY format: 748397

Enter End Date in DD-MM-YYYY format(upto 50 days would be optimal): 829304

Choose your countries (Press Ctrl to multiselect) : Afghanistan, Albania, Algeria, American Samoa

submit



localhost:5000/error/test/Invalid%20dates%2C%20either%20format%20or%20date%20itself

Apps Gmail Coding Math and the Inter... Play Online Play vs... College

Hello test

Sorry test, you have an error with the values inputted

It seems to be related to "Invalid dates, either format or date itself"

Bibliography

- Computer Science with python Textbook for Class 12 Examination 2020-2021 – Sumita Arora
- https://www.tutorialspoint.com/flask/index.htm - Flask Tutorial
- https://flask.palletsprojects.com/en/1.1.x/tutorial/ - Flask Tutorial
- fixer.io - Currency APIs - Documentation
- https://matplotlib.org/3.3.3/contents.html - Matplotlib documentation
- https://www.w3schools.com/html/ - HTML Tutorial
- https://www.tutorialspoint.com/html/index.htm - HTML Tutorial