

Assignment 2

by Paul Llewellyn Richards

FILE	15889_PAUL_LLEWELLYN_RICHARDS_ASSIGNMENT_2_314650_1368058909.DOCX (395.75K)		
TIME SUBMITTED	12-JAN-2021 08:20PM (UTC+0000)	WORD COUNT	1153
SUBMISSION ID	141831364	CHARACTER COUNT	5357

Name: Paul Richards

Student No: 0251968

Course: Applied Computer Networks and Cybersecurity

Module: Network Programmability
(ACCA5029)

Date:15/01/2021

Contents

Introduction 3

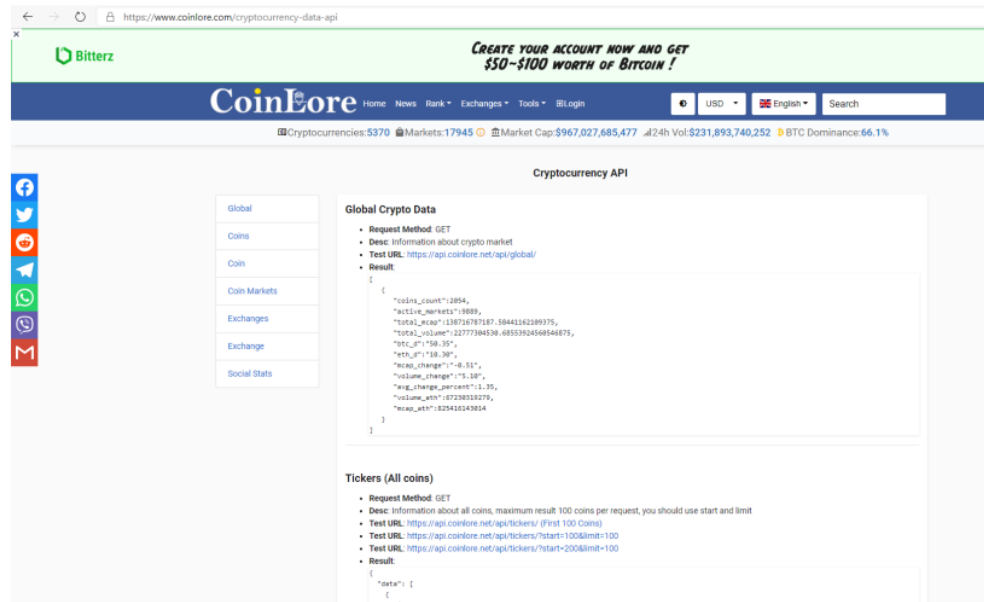
Purpose and scope of the Application 3

Details of Application 3

Bibliography 10

Introduction

My application is based on a Public API that is linked to a crypto currency website called CoinLore. A screenshot of the website is shown below:



I found the website on GitHub at the following URL: [public-apis/public-apis: A collective list of free APIs for use in software and web development. \(github.com\)](https://github.com/public-apis/public-apis)

Purpose and scope of the Application

The purpose of my application is to allow the user to retrieve data from the CoinLore website, then provide the user with the option to filter the data or view all the data. The user will have the option to repeat the process as many times as they like. Details of how the application has been built to cater for these options, is covered below.

Details of Application

CoinLore has many endpoints, but we will be retrieving data from the ticker endpoint. According to Ecrptionary [1], A ticker is an abbreviation of the long name of the crypto currency, e.g. BTC for Bitcoin. This make reading and writing crypto currencies easier.

Below is the endpoint I will be retrieving data from:

```
ticker_url = 'https://api.coinlore.net/api/tickers/?structure=array'
```

I have set an optional parameter at the end of the URL '?structure=array' to bring back the data from the endpoint in the form of an array, to make it easy to read and manipulate the data.

I have created a menu to help the user filter through the data when requesting the ticker URL. This has been achieved by creating four variables:

1. Limit
2. Start
3. Sort
4. Convert

A screenshot of the four variables in the code are shown below:

```
limit = 100  
start = 1  
sort = 'rank'  
convert = 'USD'
```

The limit variable will be used to limit the number of currencies requested, e.g. if the user only wants to retrieve 10 crypto currencies, they can enter 10 when prompted to enter a limit.

The start variable will allow the user to select what cryptocurrency they want to start from, so if they select 2, then they will start from the second currency on the crypto currency list.

The sort variable allows you to sort the data according to what key is available in the dictionary of the CoinLore site, e.g. in the code, I have hard coded it to rank for testing purposes.

The convert variable allows you to convert the crypto currency to a local currency, e.g. USD, United States Dollars. In the code. I have hard coded this to USD for testing purposes.

Then I ask the user if they want to filter the data by presenting them with the option, 'Do you want to enter any custom parameters?'. Below is a screenshot of the question presented to the user when the program is executed.

```
Do you want to enter any custom parameters? (y/n):
```

In the code, I have set up the question by using a simple input statement as shown below:

```
choice = input('Do you want to enter any custom parameters? (y/n):')
```

If the user selects yes, they are presented with four more options which they will need to fill in. A screenshot of the four options are shown below:

```
What is the custom limit?: 10
What is the custom start number?: 2
What do you want to sort by?: rank
What is your local currency?:USD
```

The four questions link to the four variables set up earlier in the code. What is the custom limit is related to the limit variable. What is the custom start number is related to the start variable. What do you want to sort by relates to the sort variable. What is your local currency relates to the convert variable.

In the code, I have used an 'if' statement to give the user the option to use custom parameters by setting it to Y, and created a block of code to ask the user questions by using input statements that store the data in the four variables discussed earlier, which is then added to the URL in the GET request when retrieving the data from CoinLore.

```
if choice == 'y':
    limit = input('What is the custom limit?: ')
    start = input('What is the custom start number?: ')
    sort = input('What do you want to sort by?: ')
    convert = input('What is your local currency?:')

ticker_url += '&limit=' + str(limit) + '&sort=' + sort + '&start=' + str(start) + '&convert=' + convert
request = requests.get(ticker_url)
results = request.json()
```

I have filtered out the data that the user regards as relevant to the crypto currency, to save the user having to search through the data to find what they need. When retrieving the data using the CoinLore ticker URL, it will retrieve a lot of unnecessary data such as name and name ID which repeats the same value, e.g. Bitcoin, as shown in the screenshot below from the CoinLore website.

Tickers (All coins)

- **Request Method:** GET
- **Desc:** Information about all coins, maximum result 100 coins per request, you should use start and limit
- **Test URL:** <https://api.coinlore.net/api/tickers/> (First 100 Coins)
- **Test URL:** <https://api.coinlore.net/api/tickers/?start=100&limit=100>
- **Test URL:** <https://api.coinlore.net/api/tickers/?start=200&limit=100>
- **Result:**

```
{
  "data": [
    {
      "id": "90",
      "symbol": "BTC",
      "name": "Bitcoin",
      "nameid": "bitcoin",
      "rank": 1,
      "price_usd": "6456.52",
      "percent_change_24h": "-1.47",
      "percent_change_1h": "0.05",
      "percent_change_7d": "-1.07",
      "price_btc": "1.00",
      "market_cap_usd": "111566042785.56",
      "volume24": "3997655362.9586277",
      "volume24a": "3657294860.710187",
      "csupply": "17282687.00",
      "tsupply": "17282687",
      "msupply": "21000000"
    },
    {
      "info": {
        "coins_num": 1969,
        "time": 1538560355
      }
    }
  ]
}
```

Whereas by filtering the data, the user is presented with only data that is relevant to them. See screenshot below of the filtered data below:

```
11:ChainLink(LINK)
Market cap:           $5,239,475,964.68
Price:                $14.97
24h Volume:           $888,638,480.7738593
Hour change:          $3.81%
Day change:           $6.81%
Weekly change:        $30.25%
Total supply:         1,000,000,000.0
Circulating Supply:   350,000,000.0
Total supply:         1,000,000,000.0
Percentage of coins in circulation: 35.0
```

In the code, I have used a dictionary and for loop to filter the data the user requires. A screenshot of the code is shown below:

```

for currency in data:
    rank = currency['rank']
    name = currency['name']
    symbol = currency['symbol']
    circulating_supply = currency['csupply']
    total_supply = currency['tsupply']
    price = currency['price_usd']
    market_cap = currency['market_cap_usd']
    hour_change = currency['percent_change_1h']
    day_change = currency['percent_change_24h']
    weekly_change = currency['percent_change_7d']
    volume = currency['volume24']

    volume_string = '{:,}'.format(volume)
    market_cap_string = '{:,}'.format(float(market_cap))
    circulating_supply_string = '{:,}'.format(float(circulating_supply))
    total_supply_string = '{:,}'.format(float(total_supply))

    print(str(rank) + ' ' + name + ' (' + symbol + ')')
    print('Market cap: \t\t\t\t$' + market_cap_string)
    print('Price: \t\t\t\t\t$' + str(price))
    print('24h Volume: \t\t\t\t$' + volume_string)
    print('Hour change: \t\t\t\t$' + str(hour_change) + '%')
    print('Day change: \t\t\t\t$' + str(day_change) + '%')
    print('Weekly change: \t\t\t\t$' + str(weekly_change) + '%')
    print('Circulating Supply: \t\t\t' + circulating_supply_string)
    print('Total supply: \t\t\t\t' + total_supply_string)
    print('Percentage of coins in circulation: \t' + str(float(circulating_supply) / float(total_supply) * 100))
    print()

```

The currency list will store all the data found in the dictionary data keys by storing them in variables, as shown in the screenshot below:

```
price = currency['price_usd']
```

The data that is selected in the data dictionary is then printed out using the print commands, as shown in the screenshot below:

```
print('Price: \t\t\t\t\t$' + str(price))
```

When the user runs the program, the data will be presented as shown in the screenshot below:

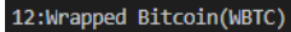
```

11:ChainLink(LINK)
Market cap:                $5,239,475,964.68
Price:                     $14.97
24h Volume:                $888,638,480.7738593
Hour change:               $3.81%
Day change:                $6.81%
Weekly change:             $30.25%
Total supply:              1,000,000,000.0
Circulating Supply:        350,000,000.0
Total supply:              1,000,000,000.0
Percentage of coins in circulation: 35.0

```

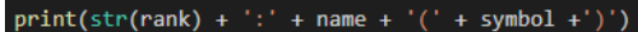
I have also formatted the data to make it easier for the user to read. By using tabs, the format command, and concatenating certain data types.

I have concatenated the rank, name and symbol of the crypto currency to make it easier for the user to read as shown in the screenshot below:



12:Wrapped Bitcoin(WBTC)

This was achieved by using the following line of code:



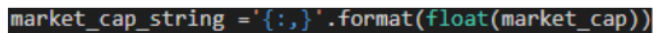
```
print(str(rank) + ':' + name + '(' + symbol + ')')
```

With the large numbers produced by the code, I have used the format and comma options to break up the numbers to make them easier to read as shown in the screenshot below:



Market cap: \$4,180,735,435.13

This was achieved by using the following code:



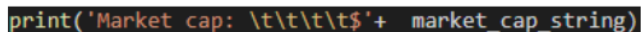
```
market_cap_string = '{:,}'.format(float(market_cap))
```

The name of the variable and the value have been separated by tabs to make it easier for the user to read the data, as shown below:



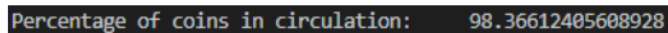
Market cap: \$4,180,735,435.13

This was achieved by using the following code:



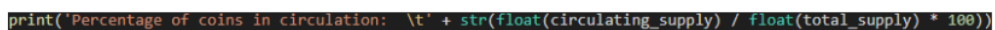
```
print('Market cap: \t\t\t\t$'+ market_cap_string)
```

Also to make it easier to compare circulating supply with the total supply of a crypto currency, I have divided the circulating supply by the total supply, which gives a user a percentage of the circulating supply as part of the total supply, see screenshot below:



Percentage of coins in circulation: 98.36612405608928

This was achieved using the following code:



```
print('Percentage of coins in circulation: \t' + str(float(circulating_supply) / float(total_supply) * 100))
```

If the user wishes to run the program more than once, I have given them the option after running the program to run it again, as shown in the screenshot below:

```
Run again y/n:
```

In the code, I have created a input statement requesting if they want to run the code again as shown below:

```
choice = input('Run again y/n:')
```

At the beginning of the code, I have added a While loop, which will run the program again if the user selects yes. At the end of the code, I have added an if statement set to 'n' and 'break'. If the user selects 'no' and does not want to run the program again, the 'if' and 'break' statement will exit the program. See screenshots below:

```
import json
import requests

while True:

    ticker_url = 'https://api.coinlore.net/api/tickers/?structure=array'
```

```
    if choice == 'n':
        break
```

Bibliography

- [1] ecrptionary, "Ticker Symbol," [Online]. Available:
<https://decryptionary.com/dictionary/ticker-symbol/#:~:text=Get%20Started%20Here!,of%20crypto%20or%20stock%20quicker..>
[Accessed 4 Janary 2020].

Assignment 2

GRADEMARK REPORT

FINAL GRADE

76/100

GENERAL COMMENTS

Instructor

Implementation (70): 55

Report (30): 21

A very nice effort and an interesting application - clearly well thought out, with inclusion of a couple of things you've clearly learned independently. Good stuff!

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10
