Programming task

In these scenarios, we would like you to create a command line tool which can obtain the latest price of Bitcoin from one of the many available APIs.

Using the Bitstamp JSON API, please implement the following:

1. Script to get the latest price

The API provides a number of functions. The `ticker` function returns the latest cryptocurrency prices (and other information). The endpoint `https://www.bitstamp.net/api/v2/ticker/{currency_pair}/`, takes as an argument `currency_pair`, which can be one of the following:

`btcusd`, `btceur`,`eurusd`, `xrpusd`, `xrpeur`, `xrpbtc`, `ltcusd`, `ltceur`, `ltcbtc`, `ethusd`, `etheur`, `ethbtc`, `bchusd`, `bcheur`, `bchbtc`.

An example call to obtain information for `btcusd` from `https://www.bitstamp.net/api/v2/ticker/btcusd` would yield the following JSON data:

```
{
   "vwap" : "4425.54",
   "timestamp" : "1542804560",
   "open" : 4349.41,
   "low" : "4048.58000000",
   "volume" : "31398.86586924",
   "bid" : "4535.60",
   "high" : "4761.00000000",
   "ask" : "4540.83",
   "last" : "4538.34"
}
```

Your task is to write a script, which when run will simply obtain this information from the API, format the result and display the information to the user. You may use a programming or scripting language of your choice (E.g., Python, Bash).

The program/script should accept as an argument one of the `btcusd`, `...`, `bchbtc` options and obtain the relevant data from the API. You should also account for erroneous user input.

Example

```
./my_script --pair btcusd
```

Output:

```
Current price as of Wed 21 Nov 2018 13:55:33 GMT
================================================
bid 4467.79
ask 4470.57
open    4349.41
high    4761.00
low 4048.00
last    4467.79
vwap    4427.84
volume  30908.13133993
```

Note: You are free to format the output in any way you please - For example, you may wish to introduce terminal colours.

2. Script to get a real time price feed.

Building on the above, the next task is to create a script/program to show a continuous price feed to the user. A simple way to achieve this would be to wrap your first script with the Unix watch command.

However, for this task you should extend your script to accept an additional command-line argument "`<refresh time>`" which will be used to specify the frequency (in milliseconds) at which to update the price information. For example, if a refresh time of 500 is specified by the user, the script should fetch the latest information from the Bitstamp price API (as above) and display to the user every 0.5 seconds.

- Before each update, the screen should be cleared.
- If the user specifies 0, this should be interpreted as "update the price information whenever a change occurs". You might implement this by polling the remote API once per second to check for a change. If the data has changed, then update the output, else do nothing.
- Your script/program should run forever until the user issues `control-c`, in which case the script/program should exit gracefully.

3. Service wrapper.

For this task, the requirement is to implement a very basic web app to display the same information as above, but this time in a web-browser.

Upon visiting a page, the user should be presented with an up-to-date view of the current market price along with the other variables: `bid` / `ask` etc.

By default, the price information displayed should be from the `btcusd` feed, E.g, from `https://www.bitstamp.net/api/v2/ticker/btcusd/`.

You may choose to implement the page entirely on the client side (in javascript), in which case you will be making calls directly to the Bitstamp API, or you may choose to implement a "price server" (using Flask for example if using Python).

You may use any language(s) you wish and there is no right or wrong or right solution, as long as the solution works.

Some guidance

- We are looking for clean, concise code for this exercise.
- You should deal with potential issues such as if the remote price API is unavailable.
- The task is open ended: You are free to do as little or as much as you like.
- We appreciate that this task may require some time (depending on how you approach it), we would therefore accept *as a minimum* a description of how you would deliver this service. If you choose to do this, please be as specific as possible in terms of your choice of software stack and proposed architecture.

Extras

Although not at all expected, we would appreciate bonus features such as:

- Ability to switch the feed to any of the following: `btcusd`, `btceur`,`eurusd`, `xrpusd`, `xrpeur`, `xrpbtc`, `ltcusd`, `ltceur`, `ltcbtc`, `ethusd`, `etheur`, `ethbtc`, `bchusd`, `bcheur`, `bchbtc` (E.g., by a drop-down in the UI, a set of tabs or different URL for each)
- A view of *all* of the above feeds on the same page.
- Real time updates in the front-end: the price information may update without refreshing the page. (You might make use of websockets to achieve this.)
- A visualisation of the time series generated by the price feed.

Deliverables

Please submit your work **at least** 2 days before your interview date.

If possible, please commit your work to Github/Bitbucket and email us a link to your repository: datasciencecampus@ons.gov.uk. We can also create a private Github repository for you to work in if you prefer. Please contact us in the first instance if you wish us to set this up for you.

While we would prefer your work to be shared with us via Github/Bitbucket, we are happy to accept a tar/zip file via email. If you choose to do this and

wish to send your code as an attachment, please keep the zipped size as small as possible (<= 2 MB).

- Email: datasciencecampus@ons.gov.uk
- Subject: Your Name - Interview code

Questions

Please feel free to email the campus with any questions you may have with respect to this task.

We look forward to meeting you!