

1903054 - Timothy Ferfers

Assignment 2

by Tim Ferfers

FILE	17246_TIM_FERFERS_1903054_- _TIMOTHY_FERFERS_ASSIGNMENT_2_314650_1763416718.DOCX (1.56M)		
TIME SUBMITTED	13-JAN-2021 03:58PM (UTC+0000)	WORD COUNT	1061
SUBMISSION ID	141950917	CHARACTER COUNT	5199

Network Programmability

ACCA5029

Timothy Ferfers

1903054

BSc: Computing (Computer Networks and Cybersecurity)

Implementation of Currency Exchange Rates (& Records) Checker: Report

When considering what type of application to design through Python, my earliest intention was to utilise a database of sorts which users could interact with to grab specific results. With my knowledge of Python and REST services, I knew that at minimum I wanted to incorporate a GET function to display lists of chosen data which users can decide through input.

Originally, my application was going to be based on locating bikes and tourism services in different countries and cities worldwide, using an API found via GitHub, however I found integrating this API in the way I desired very temperamental. After some testing, I also discovered that the creator of this API also had included very few countries and cities within those countries, therefore I was not happy with the scope of its potential output and did not progress to development with this API.

CityBikes API Documentation

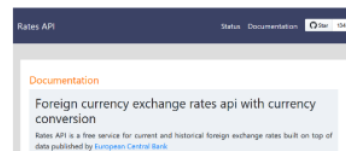
Welcome to the CityBikes API documentation page. This is a brief explanation on how to use CityBikes data.

API End points

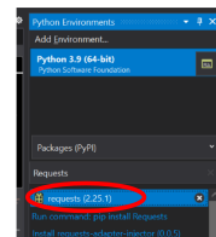
<https://api.citybikes.es/v2/networks>

```
{
  "networks": [
    {
      "name": "02000000",
      "code": "02000000",
      "url": "https://api.citybikes.es/v2/networks/02000000",
      "status": "active",
      "type": "bicycle",
      "country": "ES",
      "city": "Madrid",
      "operator": "02000000"
    },
    {
      "name": "02000001",
      "code": "02000001",
      "url": "https://api.citybikes.es/v2/networks/02000001",
      "status": "active",
      "type": "bicycle",
      "country": "ES",
      "city": "Madrid",
      "operator": "02000001"
    },
    ...
  ]
}
```

This led me back to GitHub to search for a new API to base my application around. I finally decided on currency exchange rates. The 'Rates API' which I settled one major factor I was drawn to; it also includes historical records of different currencies exchange rates. Therefore, I knew it had the potential to give very diverse responses if I implemented the ability to further refine currency searches by a past date.



I wrote my Python application with Visual Studio, firstly updating the Python extensions packages to include requests. This enables me to use the import requests function and therefore perform the basic operations that will be required to utilise the API in my application.



After setting up Visual Studio, I began with the implementation of the API's URL, using this I am able to select a single currency to use as the base for my comparison. Since the URL defines the primary currency as "base", it made sense to also name the user input related to it as 'base'. The first segment of my application I developed was purely aimed at allowing the user to input a base currency, and then be sent a list of all supported currencies and their exchange rates in relation to that chosen base currency.

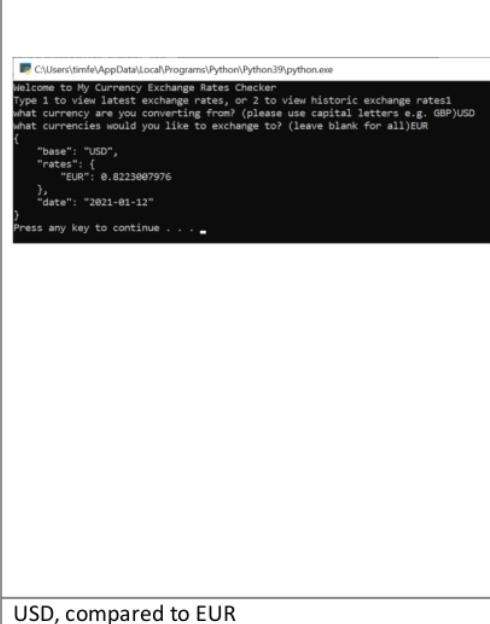
My next stage was to simply include another variable in the URL which was defined by the second currency a user could input. This way, the user may input the secondary currency (named "symbols" accordingly with API) which they want their primary currency compared to. Whereas if the user does not input a currency for symbols, the user may view a complete list of currencies compared to their chosen base. Additionally, I ran into an error where only the first URL (no symbols included) would be used regardless of symbol input. I quickly discovered this was due to the fact I had named

```
#!/usr/bin/env python3
#User may then choose another currency to compare against
#Users input stored accordingly
base=input("What currency are you converting from? (please use capital letters e.g. GBP)")
symbols=input("What currencies would you like to exchange to? (leave blank for all)")
#Result if user leaves comparison blank
if symbols=="":
    urlone = "https://api.ratesapi.io/api/latest?base="+ base
    response = requests.get(urlone)
    data = response.json()
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))
#Result if user chooses comparison
else:
    url = "https://api.ratesapi.io/api/latest?base="+ base + "&symbols=" + symbols
    response = requests.get(url)
    data = response.json()
    parsed = json.loads(data)
    print(json.dumps(parsed, indent=4))
```

Code for users' "base" and "symbols" choices

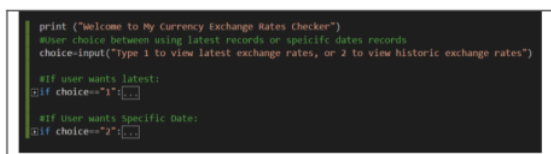
both URLs: "url". Therefore, I renamed the first URL to "urlone" to ensure that either URL can be distinctively called.

The following screenshots detail a search using a particular currency as comparison and another showing the list view result (no comparison defined).

 <pre> C:\Users\timfe\AppData\Local\Programs\Python\Python39\python.exe Welcome to My Currency Exchange Rates Checker Type 1 to view latest exchange rates, or 2 to view historic exchange rates! What currency are you converting from? (please use capital letters e.g. GBP)USD What currencies would you like to exchange to? (leave blank for all)EUR { "base": "USD", "rates": { "EUR": 0.8223007976 }, "date": "2021-01-12" } Press any key to continue . . . </pre>	 <pre> C:\Users\timfe\AppData\Local\Programs\Python\Python39\python.exe Welcome to My Currency Exchange Rates Checker Type 1 to view latest exchange rates, or 2 to view historic exchange rates! What currency are you converting from? (please use capital letters e.g. GBP)USD What currencies would you like to exchange to? (leave blank for all) { "base": "USD", "rates": { "GBP": 0.7354658334, "HKD": 7.7554477428, "IDR": 14344.9962999464, "ILS": 3.1610069462, "DKK": 6.1166844832, "INR": 73.2600115122, "CHF": 0.8890716224, "KRW": 19.9627316832, "CZK": 21.53609789, "SGD": 1.3276866679, "THB": 30.129923526, "HRK": 6.2334511964, "EUR": 0.8223007976, "NVR": 4.8574788259, "NOK": 8.5326042266, "CNY": 6.4613107475, "BGN": 1.6082559, "PHP": 48.0501318088, "PLN": 3.7207466491, "ZAR": 15.3991448072, "CAD": 1.2753885371, "ISK": 128.4433845981, "BRL": 5.47586654716, "RON": 4.0056738755, "NZD": 1.3929775512, "TRY": 7.4870487624, "JPY": 104.2184030919, "RUB": 74.0511293397, "KMD": 1869.9063399391, "USD": 1.0, "AUD": 1.2944659156, "HUF": 295.7487048762, "SEK": 8.2952059863 }, "date": "2021-01-12" } Press any key to continue . . . </pre>
USD, compared to EUR	USD, no particular comparison chosen

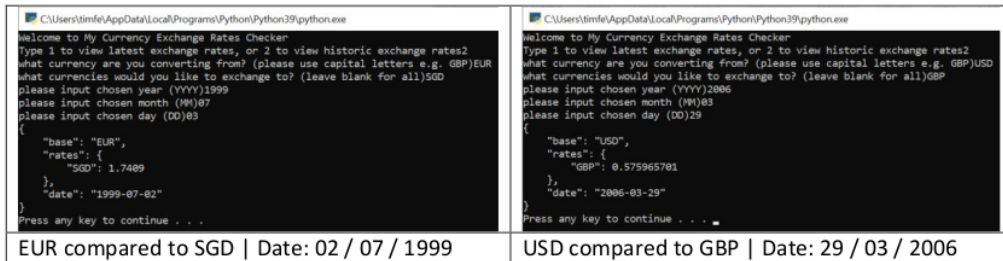
After implementing the ability to choose currency for comparison, I wanted to add the ability to define a specific date in history on top of the existing requests which users can make. To solve this, I observed the format of my current URL: <https://api.ratesapi.io/api/latest?base=> I understand that the "latest?" segment means the search defaults to the latest exchange rates on record, so by changing this to the right day / month / year format I could choose the associated date of results.

Before including this new format of URL, the application would need a way to distinguish between the existing 'latest' default and a new option where the date can be changed. My solution here was to ask the user before the currencies are chosen, whether they want latest records or a specific date. This allows me to segment the entire code for latest and for specific date under two separate 'if' statements.

 <pre> print("Welcome to My Currency Exchange Rates Checker") #user choice between using latest records or specific dates records choice=input("Type 1 to view latest exchange rates, or 2 to view historic exchange rates") #if user wants latest: if choice=="1": #if user wants Specific Date: if choice=="2": </pre>	'if' statements relative to Latest or Specific Date (nested content collapsed, to fit both into screenshot)
---	--

Once both choices for date had been defined, all which remained was to define how users could input date and translate that into the segments in a URL. The structure of the date in the URL is year, followed by month, followed

by day. The user is instructed to input these, which are then translated into URL the same way base and symbols is. The result of using varied dates and currencies is showed in the follwing screenshots.



This application has had a fairly straightforward developments cycle, implementing RESTful services functions such as GET in order to display information was never complicated. This project does feel left fairly open to improvements, as new features could be included for greater versatility of available services. One such improvements I have considered is to write a simple calculator feature. This would utilise the current system and allow the user to calculate an amount by the resulting rates of their first search. Similarly, to the current system, a variable would be created, for example, “basevalue” and “symbolssvalue”, which would grab their values from the output of the users search, instead of direct user input like before.

In addition, I would like include the ability to display multiple comparisons in the symbols. Such as choosing GBP for base but implement a way for the user to input more than one symbol to compare to the base. I have already theorised that simply using commas “,” and no spaces between currency names in the URL, might a be an acceptable format to use in the API.

Even though a prompt is present to advise the user what format to use when they input a currency, I think it could use a more intuitive design for users. Specifically, I have considered adding a list of the supported currencies which displays before users input their decisions. Ideally this could also be linked to the API’s database and therefore be able to update if any currencies change, add, or are removed.

1903054 - Timothy Ferfers Assignment 2

GRADEMARK REPORT

FINAL GRADE

83/100

GENERAL COMMENTS

Instructor

Implementation (70): 59

Report (30): 24

A good submission, with a lot of clear effort put into the application. Some very nice effort overall. A little more could potentially go into the output formatting, but the work is of a very high standard. Well done, Tim.

PAGE 1

PAGE 2

PAGE 3

PAGE 4
