Lab 18 - Classes and operator overloading

**Objective:**

To practice, understand and design Python classes. To use operator overloading in a real problem.

**Problem:**

Extend the class Currency in the file Lab18skeleton.py in Week 11 folder. This class is to be instantiated with an amount and currency type (three character currency code) and is used to fetch information from the web. The class details are:

\_\_init\_\_: The constructor takes the following arguments (with defaults indicated):

a. An amount. Default: 1

b. A currency code. If the currency code provided is invalid, set the currency code to "" (an empty string) and the amount to 0. Default: EUR

convert\_to: This method takes a single argument, a currency code, with no default. It returns a new Currency object with the new currency code and the converted amount. This method is partially implemented. It uses the urllib.request library for opening URLs. You can see more about this library here <https://docs.python.org/3/library/urllib.request.html>

The <https://exchangerate.host/> website is used to perform conversions. The URL <https://api.exchangerate.host/convert?from=USD&to=EUR> returns a JSON string:

'{"motd":{"msg":"If you or your company use this project or like what we doing, please consider backing us so we can continue maintaining and evolving this project.","url":"https://exchangerate.host/#/donate"},"success":true,"query":{"from":"USD","to":"EUR","amount":1},"info":{"rate":0.841078},"historical":false,"date":"2021-04-13","result":0.841078}'

While the URL <https://api.exchangerate.host/convert?from=EUR&to=GBP> returns the JSON string :

'{"motd":{"msg":"If you or your company use this project or like what we doing, please consider backing us so we can continue maintaining and evolving this project.","url":"https://exchangerate.host/#/donate"},"success":true,"query":{"from":"EUR","to":"GBP","amount":1},"info":{"rate":0.86595},"historical":false,"date":"2021-04-13","result":0.86595}'

Python offers a built-in JSON package which can be used to easily manipulate JSON data. You can see a nice introduction here <https://www.w3schools.com/python/python_json.asp>

For example, it is possible to access the fields of the JSON string above in the following way:

**import** json

exchange\_info = json.loads(**'{"motd":{"msg":"If you or your company use this project or like what we doing, please consider backing us so we can continue maintaining and evolving this project.", "url":"https://exchangerate.host/#/donate"}, "success":true, "query":{"from":"EUR", "to":"GBP", "amount":1}, "info":{"rate":0.86595}, "historical":false, "date":"2021-04-13", "result":0.86595}'**)

print(exchange\_info[**"info"**][**"rate"**]) *# 0.86595*

print(exchange\_info[**"result"**]) *# 0.86595*

print(exchange\_info[**"query"**][**"from"**]) *# EUR*

print(exchange\_info[**"query"**][**"to"**]) *# GBP*

print(exchange\_info[**"date"**]) *# 2021-04-13*

You should extend convert\_to to extract the exchange rate from a JSON string of this type and return a new Currency object with the corrected converted amount.

**The following operations must be implemented.** For each method it should be able to handle operands of type Currency, float, or int (for example, your \_\_add\_\_ method should be able to handle curr1 + curr2, or handle curr1 + 5, or handle curr1 + 2.71, or handle 2.71 + curr1).

a. \_\_str\_\_: Return the amount and type as a string

b. \_\_add\_\_ and \_\_radd\_\_

c. \_\_sub\_\_ and \_\_rsub\_\_

d. \_\_gt\_\_

e. \_\_repr\_\_

**Include a main program that tests all your methods.**