Currency Converter

This will cover six different ways to get the most recent foreign exchange rates, some of them parse the rates from public web pages such as [X-RATES](https://www.x-rates.com/table/?from=USD&amount=1) and [Xe](https://www.xe.com/), and others use official APIs for more commercial and reliable use, such as [Fixer API](https://fixer.io/), [Currency Conversion API](https://currencyapi.com/), and [ExchangeRate API](https://www.exchangerate-api.com/), feel free to use any one of these.

1. [Scraping X-RATES](https://thepythoncode.com/article/make-a-currency-converter-in-python#x-rates)
2. [Scraping Xe](https://thepythoncode.com/article/make-a-currency-converter-in-python#xe)
3. [Scraping Yahoo Finance](https://thepythoncode.com/article/make-a-currency-converter-in-python#yahoo-fin)
4. [Using ExchangeRate API](https://thepythoncode.com/article/make-a-currency-converter-in-python#erapi)
5. [Using Fixer API](https://thepythoncode.com/article/make-a-currency-converter-in-python#fixerapi)
6. [Using Currency Conversion API](https://thepythoncode.com/article/make-a-currency-converter-in-python#currencyapi)

[**Scraping X-RATES**](https://thepythoncode.com/article/make-a-currency-converter-in-python#x-rates)

To get started, we have to install the required libraries for all the methods below:

$ pip install python-dateutil requests bs4 yahoo\_fin

**Scraping X-RATES**

In this section, we will extract the data from the x-rates.com website. If you go to [the target web page](https://www.x-rates.com/table/?from=USD&amount=1), you'll see most of the currencies along with the most recent date and time. Let's scrape the page:

import requests

from bs4 import BeautifulSoup as bs

from dateutil.parser import parse

from pprint import pprint

The following function is responsible for making a request to that page and extracting the data from the tables:

def get\_exchange\_list\_xrates(currency, amount=1):

# make the request to x-rates.com to get current exchange rates for common currencies

content = requests.get(f"https://www.x-rates.com/table/?from={currency}&amount={amount}").content

# initialize beautifulsoup

soup = bs(content, "html.parser")

# get the last updated time

price\_datetime = parse(soup.find\_all("span", attrs={"class": "ratesTimestamp"})[1].text)

# get the exchange rates tables

exchange\_tables = soup.find\_all("table")

exchange\_rates = {}

for exchange\_table in exchange\_tables:

for tr in exchange\_table.find\_all("tr"):

# for each row in the table

tds = tr.find\_all("td")

if tds:

currency = tds[0].text

# get the exchange rate

exchange\_rate = float(tds[1].text)

exchange\_rates[currency] = exchange\_rate

return price\_datetime, exchange\_rates

The above function takes the currency and the amount as parameters and returns the exchange rates of most currencies along with the date and time of the last update.Ezoic

The time of the last update is in a span tag that has the class of ratesTimestamp. Notice we use the parse() function from [dateutil.parser](https://dateutil.readthedocs.io/en/stable/parser.html) module to automatically parse the string into a Python DateTime object.

The exchange rates are located in two tables. We extract them using the find\_all() method from the BeautifulSoup object and get the currency name and the exchange rate in each row in the tables, and add them to our exchange\_rates dictionary that we will return. Let's use this function:

if \_\_name\_\_ == "\_\_main\_\_":

import sys

source\_currency = sys.argv[1]

amount = float(sys.argv[3])

target\_currency = "GBP"

price\_datetime, exchange\_rates = get\_exchange\_list\_xrates(source\_currency, amount)

print("Last updated:", price\_datetime)

pprint(exchange\_rates)

Excellent, we use the built-in [sys](https://docs.python.org/3/library/sys.html) module to get the target currency and the amount from the command line. Let's run this:

$ python currency\_converter\_xrates.py EUR 1000

The above run is trying to convert 1000 Euros to all other currencies. Here is the output:

Last updated: 2022-02-01 12:13:00+00:00

{'Argentine Peso': 118362.205708,

'Australian Dollar': 1586.232315,

'Bahraini Dinar': 423.780164,

'Botswana Pula': 13168.450636,

'Brazilian Real': 5954.781483,

'British Pound': 834.954104,

'Bruneian Dollar': 1520.451015,

'Bulgarian Lev': 1955.83,

'Canadian Dollar': 1430.54405,

'Chilean Peso': 898463.818465,

'Chinese Yuan Renminbi': 7171.445692,

'Colombian Peso': 4447741.922165,

'Croatian Kuna': 7527.744707,

'Czech Koruna': 24313.797041,

'Danish Krone': 7440.613895,

'Emirati Dirham': 4139.182587,

'Hong Kong Dollar': 8786.255952,

'Hungarian Forint': 355958.035747,

'Icelandic Krona': 143603.932438,

'Indian Rupee': 84241.767127,

'Indonesian Rupiah': 16187150.010697,

'Iranian Rial': 47534006.535121,

'Israeli Shekel': 3569.191411,

'Japanese Yen': 129149.364679,

'Kazakhstani Tenge': 489292.515538,

'Kuwaiti Dinar': 340.959682,

'Libyan Dinar': 5196.539901,

'Malaysian Ringgit': 4717.485104,

'Mauritian Rupee': 49212.933037,

'Mexican Peso': 23130.471272,

'Nepalese Rupee': 134850.008728,

'New Zealand Dollar': 1703.649473,

'Norwegian Krone': 9953.078431,

'Omani Rial': 433.360301,

'Pakistani Rupee': 198900.635421,

'Philippine Peso': 57574.278782,

'Polish Zloty': 4579.273862,

'Qatari Riyal': 4102.552652,

'Romanian New Leu': 4946.638369,

'Russian Ruble': 86197.012666,

'Saudi Arabian Riyal': 4226.530892,

'Singapore Dollar': 1520.451015,

'South African Rand': 17159.831129,

'South Korean Won': 1355490.097163,

'Sri Lankan Rupee': 228245.645722,

'Swedish Krona': 10439.125427,

'Swiss Franc': 1037.792217,

'Taiwan New Dollar': 31334.286611,

'Thai Baht': 37436.518169,

'Trinidadian Dollar': 7636.35428,

'Turkish Lira': 15078.75981,

'US Dollar': 1127.074905,

'Venezuelan Bolivar': 511082584.868731}

That's about 1127.07 in USD at the time of writing this tutorial. Notice the last updated date and time; it usually updates every minute.

## Scraping Xe

[Xe](https://www.xe.com/) is an online foreign exchange tools and services company. It is best known for its online currency converter. In this section, we use [requests](https://docs.python-requests.org/en/latest/) and [BeautifulSoup](https://beautiful-soup-4.readthedocs.io/en/latest/) libraries to make a currency converter based on it.

Open up a new Python file and import the necessary libraries:

import requests

from bs4 import BeautifulSoup as bs

import re

from dateutil.parser import parse

Now let's make a function that accepts the source currency, target currency, and the amount we want to convert, and then returns the converted amount along with the exchange rate date and time:

def convert\_currency\_xe(src, dst, amount):

def get\_digits(text):

"""Returns the digits and dots only from an input `text` as a float

Args:

text (str): Target text to parse

"""

new\_text = ""

for c in text:

if c.isdigit() or c == ".":

new\_text += c

return float(new\_text)

url = f"https://www.xe.com/currencyconverter/convert/?Amount={amount}&From={src}&To={dst}"

content = requests.get(url).content

soup = bs(content, "html.parser")

exchange\_rate\_html = soup.find\_all("p")[2]

# get the last updated datetime

last\_updated\_datetime = parse(re.search(r"Last updated (.+)", exchange\_rate\_html.parent.parent.find\_all("div")[-2].text).group()[12:])

return last\_updated\_datetime, get\_digits(exchange\_rate\_html.text)

At the time of writing this tutorial, the exchange rate is located in the third paragraph on the HTML page. This explains the soup.find\_all("p")[2]. Make sure to change the extraction whenever a change is made to the HTML page. Hopefully, I'll keep an eye out whenever a change is made.

EzoicThe latest date and time of the exchange rate is located at the second parent of the exchange rate paragraph in the HTML DOM.

Since the exchange rate contains string characters, I made the get\_digits() function to extract only the digits and dots from a given string, which is helpful in our case.

Let's use the function now:

if \_\_name\_\_ == "\_\_main\_\_":

import sys

source\_currency = sys.argv[1]

destination\_currency = sys.argv[2]

amount = float(sys.argv[3])

last\_updated\_datetime, exchange\_rate = convert\_currency\_xe(source\_currency, destination\_currency, amount)

print("Last updated datetime:", last\_updated\_datetime)

print(f"{amount} {source\_currency} = {exchange\_rate} {destination\_currency}")

This time, we get the source and target currencies as well as the amount from the command-lines, trying to convert 1000 EUR to USD:

$ python currency\_converter\_xe.py EUR USD 1000

Output:

Last updated datetime: 2022-02-01 13:04:00+00:00

1000.0 EUR = 1125.8987 USD

That's great! Xe usually updates every minute too, so it's real-time!

## Scraping Yahoo Finance

Yahoo Finance provides financial news, currency data, stock quotes, press releases, and financial reports. This section uses the [yahoo\_fin](https://theautomatic.net/yahoo_fin-documentation/) library in Python to make a currency exchanger based on Yahoo Finance data.

Importing the libraries:

import yahoo\_fin.stock\_info as si

from datetime import datetime, timedelta

yahoo\_fin does an excellent job of extracting the data from the Yahoo Finance web page, and it is still maintained now; we use the get\_data() method from the stock\_info module and pass the currency symbol to it.

Below is the function that uses this function and returns the converted amount from one currency to another:

def convert\_currency\_yahoofin(src, dst, amount):

# construct the currency pair symbol

symbol = f"{src}{dst}=X"

# extract minute data of the recent 2 days

latest\_data = si.get\_data(symbol, interval="1m", start\_date=datetime.now() - timedelta(days=2))

# get the latest datetime

last\_updated\_datetime = latest\_data.index[-1].to\_pydatetime()

# get the latest price

latest\_price = latest\_data.iloc[-1].close

# return the latest datetime with the converted amount

return last\_updated\_datetime, latest\_price \* amount

We pass "1m" to the interval parameter in the get\_data() method to extract minute data instead of daily data (default). We also get minute data of the previous two days, as it may cause issues on the weekends, just to be cautious.

The significant advantage of this method is you can get historical data by simply changing start\_date and end\_date parameters on this method. You can also change the interval to be "1d" for daily, "1wk" for weekly, and "1mo" for monthly.

Let's use the function now:

if \_\_name\_\_ == "\_\_main\_\_":

import sys

source\_currency = sys.argv[1]

destination\_currency = sys.argv[2]

amount = float(sys.argv[3])

last\_updated\_datetime, exchange\_rate = convert\_currency\_yahoofin(source\_currency, destination\_currency, amount)

print("Last updated datetime:", last\_updated\_datetime)

print(f"{amount} {source\_currency} = {exchange\_rate} {destination\_currency}")

Running the code:

$ python currency\_converter\_yahoofin.py EUR USD 1000

Output:

Last updated datetime: 2022-02-01 13:26:34

1000.0 EUR = 1126.1261701583862 USD

## Using ExchangeRate API

As mentioned at the beginning of this tutorial, if you want a more reliable way to make a currency converter, you have to choose an API for that. There are several APIs for this purpose. However, we have picked two APIs that seem convenient and easy to get started.

[ExchangeRate API](https://www.exchangerate-api.com/) supports 161 currencies and offers a free monthly 1,500 requests if you want to try it out, and there is an open API as well that offers daily updated data, and that's what we are going to use:

import requests

from dateutil.parser import parse

def get\_all\_exchange\_rates\_erapi(src):

url = f"https://open.er-api.com/v6/latest/{src}"

# request the open ExchangeRate API and convert to Python dict using .json()

data = requests.get(url).json()

if data["result"] == "success":

# request successful

# get the last updated datetime

last\_updated\_datetime = parse(data["time\_last\_update\_utc"])

# get the exchange rates

exchange\_rates = data["rates"]

return last\_updated\_datetime, exchange\_rates

The above function requests the open API and returns the exchange rates for all the currencies with the latest date and time. Let's use this function to make a currency converter function:

def convert\_currency\_erapi(src, dst, amount):

# get all the exchange rates

last\_updated\_datetime, exchange\_rates = get\_all\_exchange\_rates\_erapi(src)

# convert by simply getting the target currency exchange rate and multiply by the amount

return last\_updated\_datetime, exchange\_rates[dst] \* amount

if \_\_name\_\_ == "\_\_main\_\_":

import sys

source\_currency = sys.argv[1]

destination\_currency = sys.argv[2]

amount = float(sys.argv[3])

last\_updated\_datetime, exchange\_rate = convert\_currency\_erapi(source\_currency, destination\_currency, amount)

print("Last updated datetime:", last\_updated\_datetime)

print(f"{amount} {source\_currency} = {exchange\_rate} {destination\_currency}")

Running it:

$ python currency\_converter\_erapi.py EUR USD 1000

Output:

Last updated datetime: 2022-02-01 00:02:31+00:00

1000.0 EUR = 1120.0 USD

The rates update daily, and it does not offer the exact exchange number as it's an open API; you can freely [sign up for an API key](https://app.exchangerate-api.com/sign-up) to get precise exchange rates.

## Using Fixer API

One of the promising alternatives is [Fixer API](https://fixer.io/). It is a simple and lightweight API for real-time and historical foreign exchange rates. You can easily create an account and get the API key.

After you've done that, you can use the /convert endpoint to convert from one currency to another. However, that's not included in the free plan and requires upgrading your account.

There is the /latest endpoint that does not require an upgrade and works in a free account just fine. It returns the exchange rates for the currency of your region. We can pass the source and target currencies we want to convert and calculate the exchange rate between both. Here's the function:

import requests

from datetime import datetime

API\_KEY = "<YOUR\_API\_KEY\_HERE>"

def convert\_currency\_fixerapi\_free(src, dst, amount):

"""converts `amount` from the `src` currency to `dst` using the free account"""

url = f"http://data.fixer.io/api/latest?access\_key={API\_KEY}&symbols={src},{dst}&format=1"

data = requests.get(url).json()

if data["success"]:

# request successful

rates = data["rates"]

# since we have the rate for our currency to src and dst, we can get exchange rate between both

# using below calculation

exchange\_rate = 1 / rates[src] \* rates[dst]

last\_updated\_datetime = datetime.fromtimestamp(data["timestamp"])

return last\_updated\_datetime, exchange\_rate \* amount

Below is the function that uses the /convert endpoint in case you have an upgraded account:

def convert\_currency\_fixerapi(src, dst, amount):

"""converts `amount` from the `src` currency to `dst`, requires upgraded account"""

url = f"https://data.fixer.io/api/convert?access\_key={API\_KEY}&from={src}&to={dst}&amount={amount}"

data = requests.get(url).json()

if data["success"]:

# request successful

# get the latest datetime

last\_updated\_datetime = datetime.fromtimestamp(data["info"]["timestamp"])

# get the result based on the latest price

result = data["result"]

return last\_updated\_datetime, result

Let's use either function:

if \_\_name\_\_ == "\_\_main\_\_":

import sys

source\_currency = sys.argv[1]

destination\_currency = sys.argv[2]

amount = float(sys.argv[3])

# free account

last\_updated\_datetime, exchange\_rate = convert\_currency\_fixerapi\_free(source\_currency, destination\_currency, amount)

# upgraded account, uncomment if you have one

# last\_updated\_datetime, exchange\_rate = convert\_currency\_fixerapi(source\_currency, destination\_currency, amount)

print("Last updated datetime:", last\_updated\_datetime)

print(f"{amount} {source\_currency} = {exchange\_rate} {destination\_currency}")

Before running the script, make sure to replace API\_KEY with the API key you get when registering for an account.

Running the script:

Last updated datetime: 2022-02-01 15:54:04

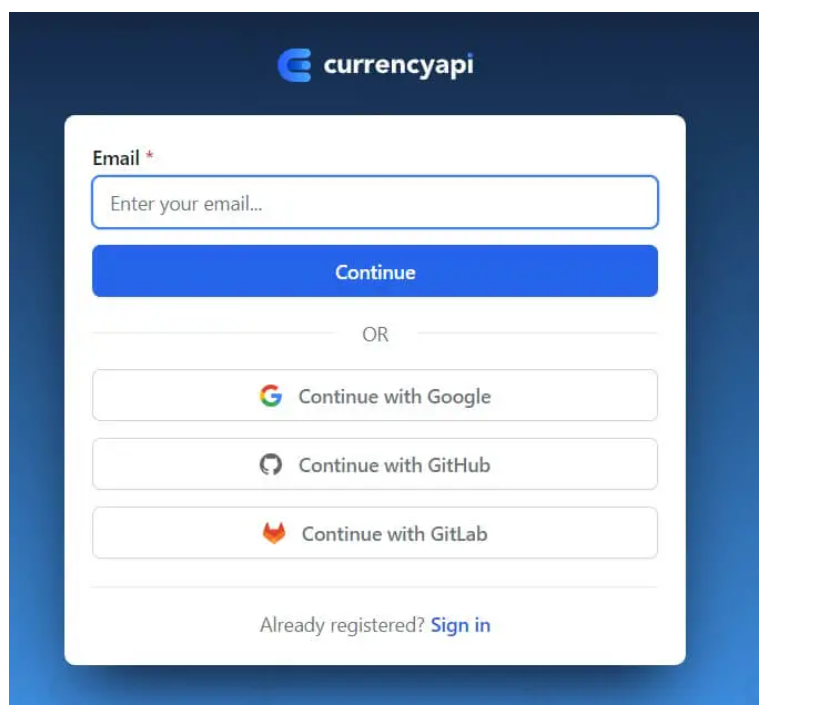
1000.0 EUR = 1126.494 USD

You can check the documentation of Fixer API [here](https://fixer.io/documentation).

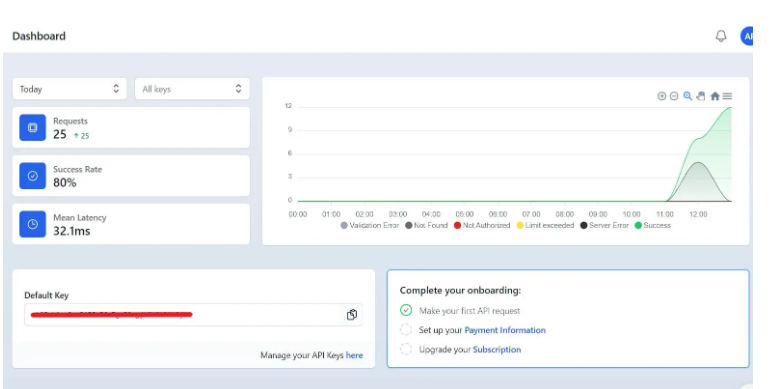
## Using Currency Conversion API

[Currency Conversion API](https://currencyapi.com/) helps you with current and historical foreign exchange rates. In this section, we're going to use the currencyapi.com API with Python.

To get started, you have to create an account. You can easily log in with your Gmail with a click of a button, or manually [sign up](https://app.currencyapi.com/register) with your email:



After signing up, you're immediately redirected to the [dashboard](https://app.currencyapi.com/dashboard):



You can see the API key hidden in my image, make sure to copy it, and let's get started with the code. Open up a new file named currency\_converter\_currencyapi.py and add the following:

import requests

import urllib.parse as p

API\_KEY = "<YOUR\_API\_KEY>"

base\_url = "https://api.currencyapi.com/v3/"

We're going to see two of the core endpoints of this API, which are the /latest to get the latest currency rates, and /historical to see the historical exchange rates. The response data format is the same, therefore, let's make a common function to handle both:

# utility function that both functions will use

def get\_currencyapi\_data(endpoint, date=None, base\_currency="USD", print\_all=True):

"""Get the list of currency codes from the API"""

# construct the url

url = p.urljoin(base\_url,

f"{endpoint}?apikey={API\_KEY}{'' if endpoint == 'latest' else f'&date={date}'}&base\_currency={base\_currency}")

# make the request

res = requests.get(url)

# get the json data

data = res.json()

# print all the currency codes and their values

if print\_all:

for currency\_code, currency\_name in data.get("data").items():

print(f"{currency\_code}: {currency\_name.get('value')}")

if endpoint == "latest":

# get the last updated date

last\_updated = data.get("meta").get("last\_updated\_at")

print(f"Last updated: {last\_updated}")

return data

This function takes the endpoint (either latest or historical), the date (in case of historical), the base currency (default is "USD"), and whether to print all the rates available, the default is True.

We're using the requests library to make the API call, retrieving the JSON data using the .json() method, printing all the rates if print\_all is True, and printing the last updated date in case it's the latest endpoint.Ezoic

Let's make the two functions for the /latest and /historical endpoints:

def get\_latest\_rates(base\_currency="USD", print\_all=True):

"""Get the latest rates from the API"""

return get\_currencyapi\_data(endpoint="latest", base\_currency=base\_currency, print\_all=print\_all)

def get\_historical\_rates(base\_currency="USD", print\_all=True, date="2023-01-01"):

"""Get the historical rates from the Currency API

`date` must be in the format of YYYY-MM-DD"""

return get\_currencyapi\_data(endpoint="historical", base\_currency=base\_currency, date=date, print\_all=print\_all)

Amazing. Let's use these functions:

if \_\_name\_\_ == "\_\_main\_\_":

latest\_rates = get\_latest\_rates()

print(f"\n{'-'\*50}\n")

# get the historical rates for the date 2021-01-01

historical\_rates = get\_historical\_rates(date="2021-01-01", print\_all=False)

# get EUR rate, for example

eur\_rate = historical\_rates.get("data").get("EUR").get("value")

print(f"EUR rate on 2021-01-01: {eur\_rate}")

First, we're getting the latest rates using our get\_latest\_rates() function, and printing them.

Second, we get the historical rates but pass False to print\_all, and then as an example, we get the EUR rate ("USD" is the base currency) of the 2021-01-01 date. Here's the output of the entire code:

ADA: 2.57927

AED: 3.672219

AFN: 86.155058

ALL: 101.250181

AMD: 388.059705

ANG: 1.79865

AOA: 510.500992

...

<SNIPPED>

...

XOF: 594.500673

XPF: 109.225174

XRP: 2.136364

YER: 250.300428

ZAR: 18.401632

ZMK: 9001.2

ZMW: 17.739516

ZWL: 321.999592

Last updated: 2023-05-01T23:59:59Z

--------------------------------------------------

EUR rate on 2021-01-01: 0.82132

The output is snipped as it's too long, a total of 177 global currencies. Please check [the CurrencyAPI documentation](https://currencyapi.com/docs) for more info.

There is an alternative to using raw API calls using [their Python wrapper here](https://pypi.org/project/currencyapicom/).

## Conclusion

There are many ways to make a currency converter, and we have covered six of them. If one method does not work for you, you can choose another one!