

Como asesor financiero deseas realizar un análisis con base en el uso de las criptomonedas como una opción para recomendar a tus clientes, para ello deberás escoger una criptomoneda y descargar el valor histórico respecto a otra criptomoneda (por ejemplo, Ethereum vs. USDT). Para facilitar la descarga y manipulación de información, se proveen las siguientes herramientas en Python:

**from functools import lru\_cache**

**from typing import Dict, Optional**

**from difflib import SequenceMatcher**

**import pandas as pd**

**import requests**

Código auxiliar en Python para explorar el conjunto de datos disponible en Binance:

**# Binance - Exchange & Symbols**

**@lru\_cache**

**def get\_binance\_exchange\_info() -> Dict:**

 **return requests.get("https://api.binance.com/api/v1/exchangeInfo").json()**

**def get\_binance\_symbols(pattern: Optional[str] = None, exact\_start: bool = False):**

 **exchange\_info = get\_binance\_exchange\_info()**

 **if exact\_start:**

 **pattern = pattern.lower()**

 **for item in exchange\_info["symbols"]:**

 **if item["status"] != "TRADING":**

 **continue**

 **if pattern is not None:**

 **if exact\_start and not item["symbol"].lower().startswith(pattern):**

 **continue**

 **elif not exact\_start and not SequenceMatcher(None, item["symbol"], pattern).ratio() > 0.5:**

 **continue**

 **yield item["symbol"]**

Ejemplo de uso: utiliza la función de “get\_binance\_symbols” para visualizar las criptomonedas disponibles en la plataforma. El argumento de “pattern” permite filtrar sobre las opciones para agilizar la búsqueda. Por ejemplo, la siguiente figura muestra cómo buscar todos los “traiding pairs” disponibles que comienzan con el ETH.

**list(get\_binance\_symbols(pattern="ETH", exact\_start=True))**



Figura 1. Criptomonedas disponibles en la plataforma.

Código auxiliar en Python para la descarga de información utilizando como base la API de KLines de Binance disponible públicamente:

**# Binance - KLines API**

**# Binance KLines API Metadata**

**binance\_metadata = dict(**

 **base\_url = "https://api.binance.com/api/v3/klines?symbol={symbol}&interval={interval}&limit=1000",**

 **columns = [**

 **"open\_time",**

 **"open\_price",**

 **"high\_price",**

 **"low\_price",**

 **"close\_price",**

 **"volume",**

 **"close\_time",**

 **"quote\_asset\_volume",**

 **"number\_of\_trades",**

 **"taker\_buy\_base\_asset\_volume",**

 **"taker\_but\_quoe\_asset\_volume",**

 **"temp",**

 **]**

**)**

**# Binance KLine API Data**

**def get\_binance\_kline\_records(symbol: str, interval: str, limit: int = 1000):**

 **print("Downloading data...")**

 **def fetch(url: str, end\_time: Optional[int] = None, n: int = 0):**

 **response = requests.get(url if not end\_time else f"{url}&endTime={end\_time}")**

 **if not response.ok:**

 **print(f"Early exit due to error when calling {url}")**

 **print(response.text)**

 **return []**

 **payload = response.json()**

 **earliest = min([ts for ts, \*\_ in payload])**

 **if end\_time is not None and end\_time <= earliest:**

 **print(f"Early exit due to 'end' reached: ", earliest, end\_time)**

 **return [**

 **[ts, \*item]**

 **for ts, \*item in payload**

 **if earliest <= ts < end\_time**

 **]**

 **size = n + len(payload)**

 **print(f"Fetch request: dataset size {size} rows with param end\_time: ", end\_time)**

 **return payload if size >= limit else payload + fetch(**

 **url=url,**

 **end\_time=earliest,**

 **n=size**

 **)[:-1]**

 **return fetch(**

 **url=binance\_metadata["base\_url"].format(symbol=symbol, interval=interval),**

 **end\_time=None,**

 **)[:limit]**

**# Binance KLine API Data as Pandas DataFrame**

**def get\_binance\_kline\_dataframe(symbol: str, interval: str, size: int = 1000):**

 **# Download data and create dataframe**

 **records = get\_binance\_kline\_records(symbol=symbol, interval=interval, limit=size)**

 **dataframe = pd.DataFrame(records, columns=binance\_metadata["columns"])\**

 **.drop("temp", axis=1)**

 **# Cast the open\_time & close\_time to datetime**

 **dataframe = dataframe.assign(**

 **open\_time=lambda df: pd.to\_datetime(df.open\_time, unit="ms"),**

 **close\_time=lambda df: pd.to\_datetime(df.close\_time, unit="ms"),**

 **)**

 **# Cast integer column**

 **dataframe = dataframe.astype({"number\_of\_trades": "int"})**

 **# Cast everything from object to float**

 **dataframe = dataframe.astype({col: "float" for col in dataframe.select\_dtypes(include=["object"]).columns})**

 **return dataframe.sort\_values(by="open\_time", ascending=False).reset\_index(drop=True)**

Ejemplo de uso: utiliza la función de “get\_binance\_kline\_dataframe” para descargar la información histórica de un “trading pair”. La información resultante es una agregación del valor “spot” de la serie de tiempo, en donde se tiene el precio de apertura, cierre, el mínimo y el máximo para una ventana de tiempo dada. Para este proyecto, se recomienda usar una ventana de tiempo de 6 horas (6h) lo cual resulta en 4 observaciones por día. El aprendedor puede elegir la ventana de tiempo que más le convenga. Finalmente, el argumento de “size” permite controlar cuántas observaciones interesa obtener.

El código está implementado de tal forma que se intentará descargar siempre la información más reciente. La siguiente figura muestra un ejemplo en donde se obtienen 5000 observaciones del “trading pair” ETHUSDT en intervalos de 6 horas.

**data = get\_binance\_kline\_dataframe(symbol="ETHUSDT", interval="6h", size=5000)**

**data**



Figura 2. Descarga de datos con la información solicitada.



Figura 3. Información resultante del “trading pair”.

Nota: se debe leer la documentación técnica de Binance, para familiarizarse con los datos y la plataforma que los provee.

Fuente: GitHub.Inc. (2022). *Binance-spot-api-docs.* Recuperado de https://github.com/binance/binance-spot-api-docs/blob/master/rest-api.md#klinecandlestick-data

Tecmilenio no guarda relación alguna con las marcas mencionadas como ejemplo. Las marcas son propiedad de sus titulares conforme a la legislación aplicable, estas se utilizan con fines académicos y didácticos, por lo que no existen fines de lucro, relación publicitaria o de patrocinio.​

Todos los derechos reservados @ Universidad Tecmilenio​

La obra presentada es propiedad de ENSEÑANZA E INVESTIGACIÓN SUPERIOR A.C. (UNIVERSIDAD TECMILENIO), protegida por la Ley Federal de Derecho de Autor; la alteración o deformación de una obra, así como su reproducción, exhibición o ejecución pública sin el consentimiento de su autor y titular de los derechos correspondientes es constitutivo de un delito tipificado en la Ley Federal de Derechos de Autor, así como en las Leyes Internacionales de Derecho de Autor. El uso de imágenes, fragmentos de videos, fragmentos de eventos culturales, programas y demás material que sea objeto de protección de los derechos de autor, es exclusivamente para fines educativos e informativos, y cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por UNIVERSIDAD TECMILENIO. Queda prohibido copiar, reproducir, distribuir, publicar, transmitir, difundir, o en cualquier modo explotar cualquier parte de esta obra sin la autorización previa por escrito de UNIVERSIDAD TECMILENIO. Sin embargo, usted podrá bajar material a su computadora personal para uso exclusivamente personal o educacional y no comercial limitado a una copia por página. No se podrá remover o alterar de la copia ninguna leyenda de Derechos de Autor o la que manifieste la autoría del material.