

ANÁLISIS DE LAS CRYPTOMONEDAS



Universidad
de Navarra

Python para análisis de datos

Máster en Big Data Science

Universidad de Navarra

Raúl Artigues Femenía

Alexandre Pérez Reina

1.	Objetivos del proyecto.....	2
2.	Descripción del script.....	2
2.1	Librerías utilizadas y carga de los datos.....	2
2.2	Creación de las pestañas.....	4
2.2.1	Primera pestaña.....	5
2.2.2	Segunda pestaña.....	7
2.2.3	Tercera pestaña.....	13
3.	Integración Cloud (AWS).....	19

1. Objetivos

El objetivo del presente proyecto es crear una pestaña, con 3 ventanas, en las que se encuentra la información y métricas de una serie de crypto monedas, las cuales son las más populares.

Para ello, el código creado se conecta a la API de Kraken para la descarga de los valores de las monedas y calcula la media móvil y el rsi y, posteriormente las plotea.

2. Descripción del script

A continuación, se va a explicar el procedimiento de la lógica del código creado para poder plotea y calcular distintas métricas que se basan en las crypto monedas más populares de hoy en día.

El presente script se puede ejecutar en cualquier consola que soporte Python 3.9.12 y no es necesario la descarga de ninguna librería, ya que en este existe el comando “! pip” que nos instala los paquetes necesarios de forma automática a la hora de ejecutar el código.

2.1 Librerías y carga de los datos

Para poder realizar las múltiples acciones en el script es necesario la instalación y carga de las librerías requeridas. En cuanto a la lectura de los datos, es necesaria la descarga del paquete krakenex y KrakenApi. Esta última es muy importante para poder conectarnos a la API de Kraken ya que nos da los permisos necesarios para la descarga de los datos. Por otra parte, para la realización de la pestaña que contiene toda la información respecto a las crypto monedas, se ha utilizado la librería Tkinter. Esta nos permite crear ventanas en el propio ordenador (como si se tratase de Google Chrome) para informar o plotear cualquier tipo de dato. Para su correcta implementación es necesario descargarse una serie de librerías, las cuales se dividen en 2 bloques. El primer de ellos hacen referencia al módulo matemático, pandas y numpy. Estas nos permiten calcular las métricas necesarias (rsi, media móvil, etc.), además de crear la estructura óptima de los datos para poder manejarlos de una mejor forma. Por la otra parte, están las librerías de matplotlib, matplotlib.figure, matplotlib.backend_tkagg y datetime. Todas estas hacen referencia a la construcción de los gráficos que aparecen en la pestaña creada.

```
#Instalamos y cargamos las librerias necesarias

!pip install pykrakenapi
!pip install tk
!pip install mplfinance

import krakenex
from pykrakenapi import KrakenAPI
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tkinter import *
from tkinter import ttk
from datetime import datetime, timedelta
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg,
NavigationToolbar2Tk)
from tkinter import messagebox

#####
#Cargamos la api
api = krakenex.API()
k = KrakenAPI(api)
```

Todas estas son las librerías que se han utilizado para el correcto funcionamiento del código que muestra la información y métricas de las crypto monedas más populares.

Una vez instaladas y cargadas las librerías se produce a realizar la conexión a la API de kraken para la lectura de los datos. Esta lectura se obtiene mediante la función get.ohlc_data y poniendo como parámetros que moneda se quiere descargar, el intervalo de tiempo (un día, un mes o varios años) y si quiere de forma ordenada.

Posteriormente, se crean tantos DataFrame como cryptos, ya que cuando se descargan los datos vienen en formato tupla y con esta estructura no es tan fácil trabajar.

Un matiz que hay que destacar es que cuando se trabaja con series temporales referentes a valores de acciones en el ámbito bursátil siempre se trabaja con el valor de cierre de estas. Es por eso, que se crear de nuevo tantos DataFrame como cryptos haya y se elige el valor de cierre de estas mediante la función .close.

Otra variable muy importante que, en este caso hay que crear, es la del tiempo. Para generar esta se utiliza la librería Datetime.

Como los datos de nuestra variable objetivo se descargan en función del día actual, es decir, el último registro será hoy y el primero será del año actual menos 2, hay que crear una variable tiempo que se corresponda con los valores de las monedas a analizar. Para esto se crea un bucle *for* que irá creando registros con el siguiente esquema: fecha_hoy – fecha_inicio (hoy – 2 años menos) + 1.

Una vez los datos de las monedas han sido estructurados adecuadamente y la variable tiempo ha sido creada satisfactoriamente, se concatenan en el mismo Dataframe para empezar con el análisis de estas.



```
#Leemos los datos
data1= k.get_ohlc_data("ETHUSD", interval=1440, ascending = True)
data2= k.get_ohlc_data("BTCUSD", interval=1440, ascending = True)
data3= k.get_ohlc_data("LTCUSD", interval=1440, ascending = True)
data4= k.get_ohlc_data("USDTUSD", interval=1440, ascending = True)
data5= k.get_ohlc_data("USDCUSD", interval=1440, ascending = True)
data6= k.get_ohlc_data("DOGEUSD", interval=1440, ascending = True)

#Convertimos la tupla a DataFrame
df1= pd.DataFrame(data1[0])
df2= pd.DataFrame(data2[0])
df3= pd.DataFrame(data3[0])
df4= pd.DataFrame(data4[0])
df5= pd.DataFrame(data5[0])
df6= pd.DataFrame(data6[0])

#Creamos un único dataframe
dfFinal= pd.DataFrame()
dfFinal["ETH"]= df1.close
dfFinal["BTC"]= df2.close
dfFinal["LTC"]= df3.close
dfFinal["USDT"]= df4.close
dfFinal["USDC"]= df5.close
dfFinal["DOGE"]= df6.close

#Establecemos período de tiempo
fecha_inicio= datetime.now() - timedelta(days=719)
fecha_hoy= datetime.now()

lista_fechas = [fecha_inicio + timedelta(days=d) for d in range((fecha_hoy - fecha_inicio).days + 1)]
dfFinal["Fecha"]= lista_fechas
```

2.2 Creación de las pestañas

En este proyecto se ha optado la decisión de utilizar la librería Tkinter para la visualización tanto de las crypto monedas como de las distintas métricas.

Para la construcción de las 3 subpestañas es necesario indicar las dimensiones de estas, su nomenclatura para posteriormente indicar que figuras se quieren representar en cada una de ellas y cerrar el loop. Además, se han creado una serie de setting de diseño para cada una de ellas, por ejemplo: cuando se selecciona la 1º pestaña cambia a color rojo y así con todas. Para verlo mejor, se presenta el código.



```
#####
#Creamos la pestaña general
window = Tk()
window.title("PROYECTO FINAL PYTHON")
window.geometry('1000x1000')
window.configure(background= '#440c29')

#Estilo de las distintas pestañas
style = ttk.Style()
settings = {"TNotebook.Tab": {"configure": {"padding": [10, 1],
                                             "background": "#fdd57e"
                                            },
                               "map": {"background": [("selected", "#C70039"),
                                                    ("active", "#fc9292")],
                                       "foreground": [("selected", "#ffffff"),
                                                     ("active", "#000000")]
                                      ]}}}
style.theme_create("mi_estilo", settings=settings)
style.theme_use("mi_estilo")

#Definimos las pestañas
tab_control = ttk.Notebook(window)
tab1 = ttk.Frame(tab_control)
tab2 = ttk.Frame(tab_control)
tab3 = ttk.Frame(tab_control)
tab_control.add(tab1, text='Inicio')
tab_control.add(tab2, text='Cotizaciones')
tab_control.add(tab3, text='Indicadores')
```

2.2.1 Primera pestaña

Cuando se ejecuta el código aparece la primera ventana la cual muestra la información esencial y básica de las crypto monedas más populares del momento. Indica el nombre de cada una de ellas, su nomenclatura y el valor en \$ a día de hoy (presente). Esta información se encuentra en una tabla como se puede observar a continuación.



PROYECTO FINAL PYTHON

Inicio Cotizaciones Indicadores

Comparador de cryptomonedas

Cryptomonedas disponibles

Nombre	Nomenclatura	Valor de hoy en \$
Ethereum	ETH	1097,93
Bitcoin	BTC	15623,11
Litecoin	LTC	60,30
Tether	USDT	0,99
USD coin	USDC	1,00
Dogecoin	DOGE	0,07

Para la creación de la tabla mencionado anteriormente se utiliza la función .Treeview que proporciona la librería Tkinter. En esta, indicamos el número de columnas que se quieren crear, el título y los valores que irán en cada una de las celdas. Cabe destacar, que las dimensiones de la página pueden cambiar, por tanto la tabla debe quedar centrada en cualquier situación. Para ello, se utiliza la función anchor=center, que se ajusta de forma automática a las dimensiones de la pestaña. Se ha elegido este método ya que por defecto tendrá un valor de 1000x1000 pero el usuario puede hacer más pequeña o grande la ventana. Por ello, se han utilizado conceptos y métodos del diseño web (css y html), como se puede ver en el siguiente código adjuntado.



```
#####
#Etiquetas de inicio
label1= Label(tab1, text="Comparador de cryptomonedas", font=("Verdana Bold",30))
label1.pack(anchor=CENTER)

label2= Label(tab1, text="Cryptomonedas disponibles")
label2.pack(anchor=CENTER)

#Tabla donde se encuentran las distintas cryptos
tv = ttk.Treeview(tab1, columns=( "col1", "col2"))
tv.column("#0", width=150)
tv.column("col1", width=90, anchor= CENTER)
tv.column("col2", width=150, anchor= CENTER)

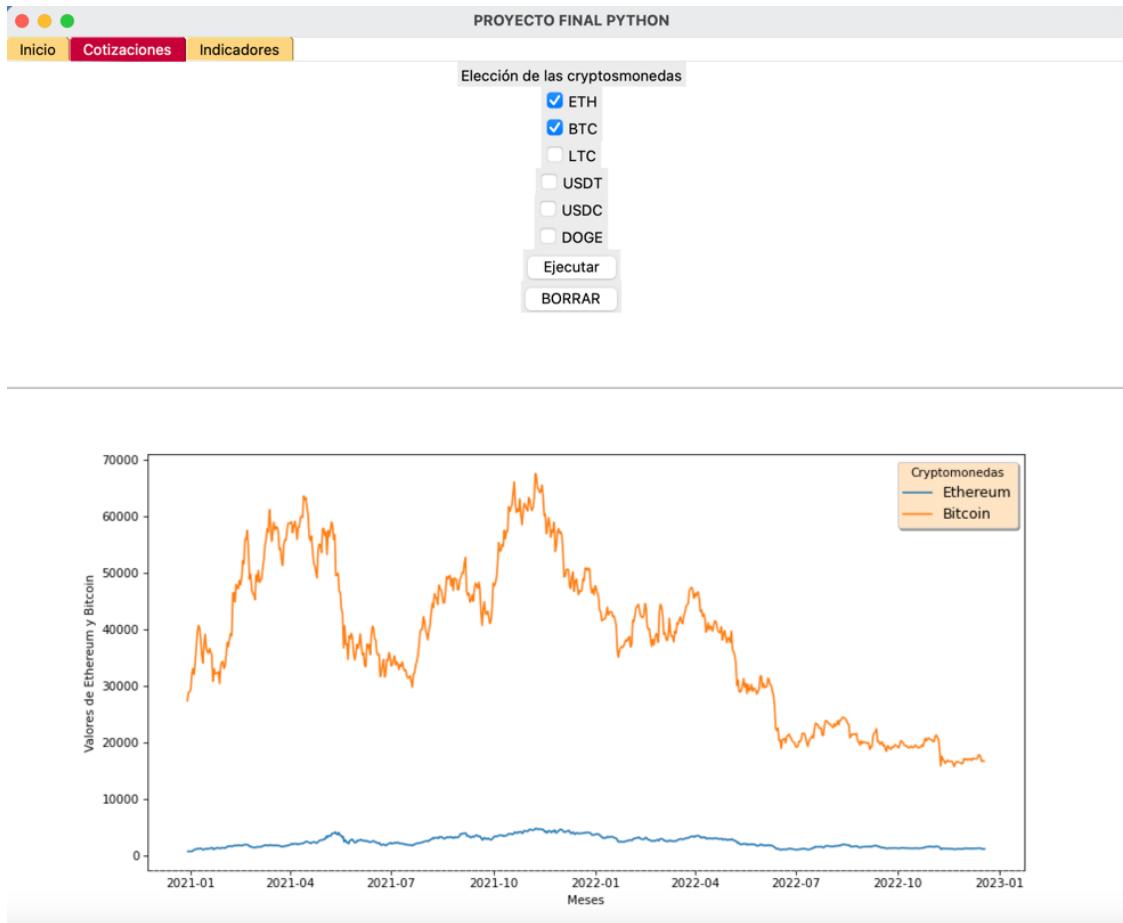
tv.heading("#0", text="Nombre", anchor= CENTER)
tv.heading("col1", text="Nomenclatura", anchor= CENTER)
tv.heading("col2", text="Valor de hoy en $", anchor= CENTER)

tv.insert("", END, text="Ethereum", values=( "ETH", "1097,93"))
tv.insert("", END, text="Bitcoin", values=( "BTC", "15623,11"))
tv.insert("", END, text="Litecoin", values=( "LTC", "60,30"))
tv.insert("", END, text="Tether", values=( "USDT", "0,99"))
tv.insert("", END, text="USD coin", values=( "USDC", "1,00"))
tv.insert("", END, text="Dogecoin", values=( "DOGE", "0,07"))

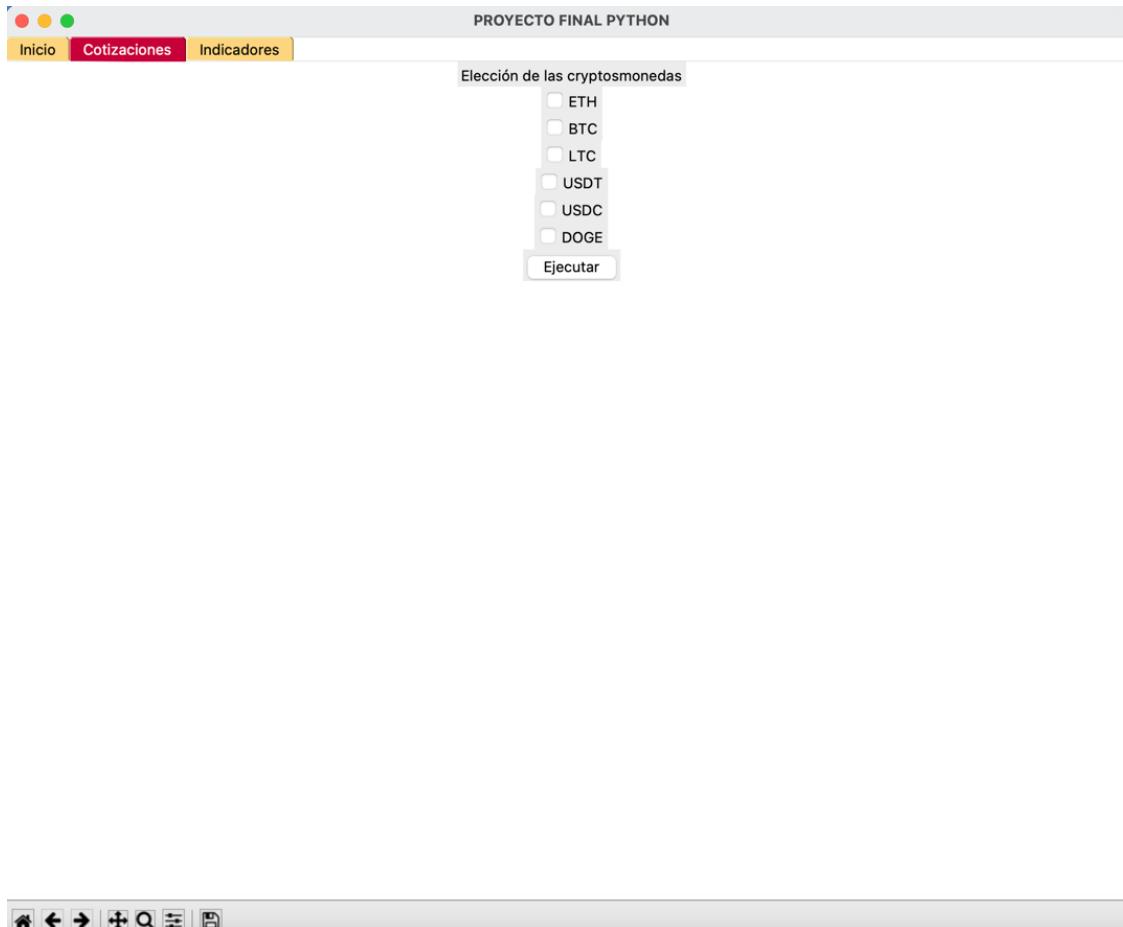
tv.pack(side='top')
```

2.2.2 Segunda pestaña

En la segunda pestaña se encuentran los valores de las crypto monedas en formato de gráfico. Esta ventana está compuesta por la elección de aquellas monedas que se quieren representar (elección múltiple) y reinicio de la figura. Como ejemplo, se presenta la imagen de como quedaría la ventana si se selecciona Ethereum y Bitcoin.



Si el usuario quiere eliminar la figura creada y plotear otros valores de una o varias crypto monedas, únicamente le debe dar click al botón de borrar y la figura se resetea, quedando con este aspecto.



A continuación, se procede a explicar cómo se ha realizado la ventana anterior en Python. Para ello en primer lugar es necesario crear tantos checkbutton como monedas haya mediante la función Checkbutton de la librería Tkinter. Esta función requiere que se le asigne una variable booleana (True or False) una etiqueta o texto y en qué pestaña irá situada, en este caso en la pestaña número 2.



```
#Segundo pestaña. Plot de las distintas cryptos
label3= Label(tab2, text="Elección de las cryptomonedas")
label3.pack(anchor=CENTER)

#Creamos los chechobox para la elecciones de las monedas
Asid1_state = BooleanVar()
Asid1_state.set(False)
Asid1 = Checkbutton(tab2, text='ETH', var=Asid1_state)
Asid1.pack(anchor= CENTER)

Asid2_state = BooleanVar()
Asid2_state.set(False)
Asid2 = Checkbutton(tab2, text='BTC', var=Asid2_state)
Asid2.pack(anchor= CENTER)

Asid3_state = BooleanVar()
Asid3_state.set(False)
Asid3 = Checkbutton(tab2, text='LTC', var=Asid3_state)
Asid3.pack(anchor= CENTER)

Asid4_state = BooleanVar()
Asid4_state.set(False)
Asid4 = Checkbutton(tab2, text='USDT', var=Asid4_state)
Asid4.pack(anchor= CENTER)

Asid5_state = BooleanVar()
Asid5_state.set(False)
Asid5 = Checkbutton(tab2, text='USDC', var=Asid5_state)
Asid5.pack(anchor= CENTER)

Asid6_state = BooleanVar()
Asid6_state.set(False)
Asid6 = Checkbutton(tab2, text='DOGE', var=Asid6_state)
Asid6.pack(anchor= CENTER)
```

Una vez se tiene la opción de elección múltiple creada hay que asignar esos valores al gráfico que se muestra en la ventana del propio ordenador. Este método se crea mediante una función denominada “cotización” que se encarga de plotear aquella/as monedas que el usuario ha seleccionado.

A su vez, dentro de dicha función se ha creado otra llamada “visu” donde recibe como parámetros el nombre de la criptomoneda y su abreviación. De esta manera, creamos un código más eficiente en las sentencias if, ahorrando líneas de código.



```
#Función para plotear el gráfico con las cotizaciones de las monedas elegidas
def cotizacion():

    #Función para plotear el gráfico de las criptomonedas
    def visu(cripto, cripto_ext):

        plot1.plot(dfFinal.Fecha,dfFinal[cripto], label=cripto_ext)
        plot1.set_xlabel("Meses")
        plot1.set_ylabel("Valores de la criptomonedada")
        plot1.spines['bottom'].set_linestyle("-")
        plot1.spines['bottom'].set_linewidth(1)
        plot1.spines['bottom'].set_color('#404040')
        plot1.legend(fontsize=12, shadow=True, facecolor="Bisque",title="Cryptomoneda")

        if (Asid1_state.get()==0 and Asid2_state.get()==0 and Asid3_state.get()==0 and
            Asid4_state.get()==0 and Asid5_state.get()==0 and Asid6_state.get()==0):
            messagebox.showinfo('Error', 'Selecciona una o más cryptos')

    else:
        try:

            #Creamos la figura
            fig= Figure(figsize = (10, 10),
                        dpi = 100)
            plot1 = fig.add_subplot(111)

            #En este apartado elegiremos la/ las monedas elegidas
            if (Asid1_state.get()==1) :
                visu('ETH','Ethereum')

            if (Asid2_state.get()==1) :
                visu('BTC','Bitcoin')

            if (Asid3_state.get()==1) :
                visu('LTC','Litecoin')

            if (Asid4_state.get()==1) :
                visu('USDT','Tether')

            if (Asid5_state.get()==1) :
                visu('USDC','USD coin')

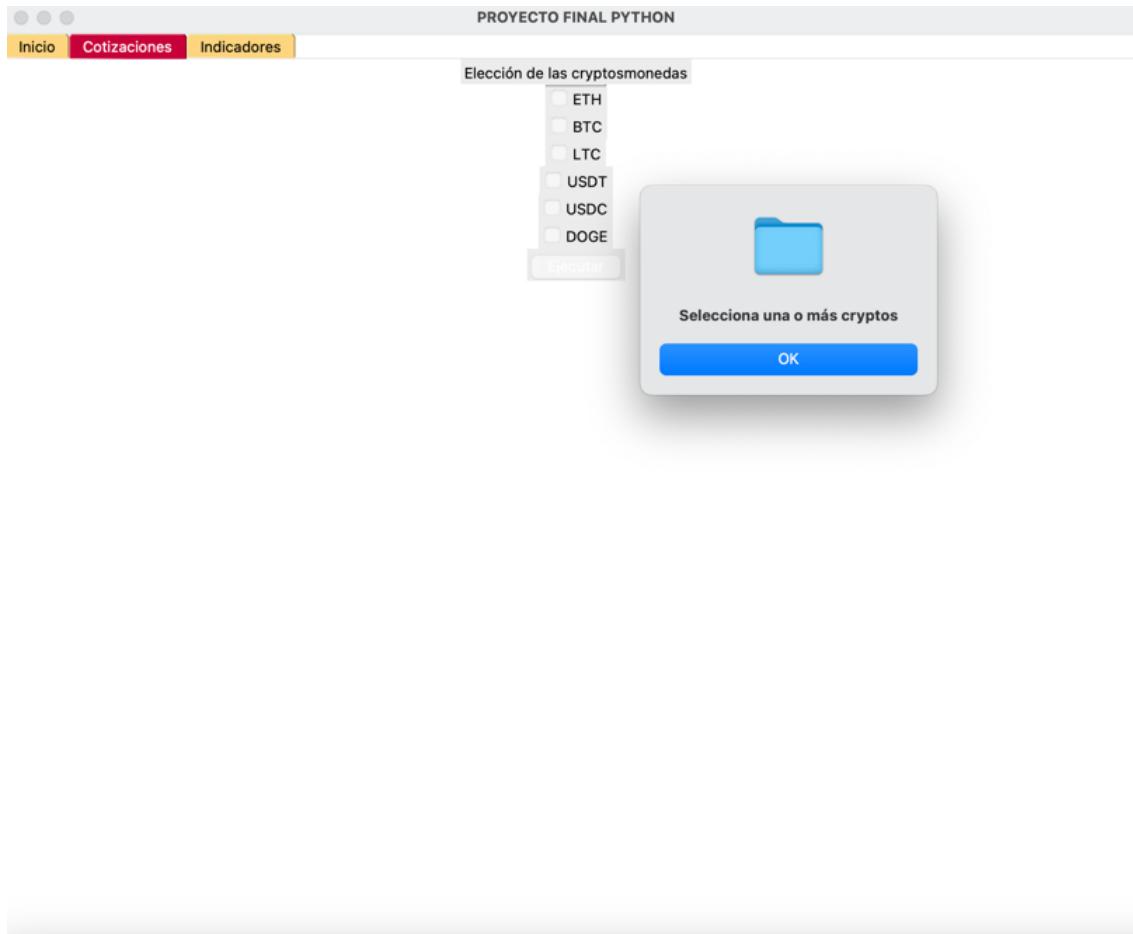
            if (Asid6_state.get()==1) :
                visu('DOGE','Dogecoin')

            #Creamos el gráfico en Tkinter
            canvas = FigureCanvasTkAgg(fig,master = window)
            canvas.draw()
            canvas.get_tk_widget().pack()
            toolbar = NavigationToolbar2Tk(canvas,window)
            toolbar.update()
            canvas.get_tk_widget().pack(side=TOP, fill=BOTH, expand=0)
            def Limpiar1():
                canvas.get_tk_widget().destroy()
                btn2.destroy()

            #Botón y función para eliminar los resultados anteriores
            btn2 = Button(tab2, text="BORRAR", command=Limpiar1)
            btn2.pack(anchor=CENTER)

        except:
            messagebox.showinfo('Error', 'Selecciona una o más cryptos')
```

Como se observa en la imagen anterior, se ha integrado la opción de *try* y *except* por si un usuario quiere plotear una cotización y no ha seleccionado ninguna moneda le saltará un mensaje de error, diciéndole que tiene que seleccionar alguna. Para este mensaje, se utiliza la librería *messagebox* que se encuentra dentro de Tkinter.



Cuando el usuario selecciona las monedas y le da click al botón de ejecutar, llama a la función cotización y esta se encarga de recopilar los datos de las variables seleccionarlas, comprobar que no haya ningún error y plotearlas. La figura generada en la pestaña es gracias a las funciones de las librerías matplotlib.figure y matplotlib.backend_tkagg. Estas comprueban que los datos de la gráfica a generar son los adecuados y, posteriormente, lo asigna a la ventana introducida.

Por otro lado, se crea una función “Limpiar1” dentro de la función anterior que permite reiniciar el gráfico de los valores de las monedas/as seleccionadas la cual está vinculada a un botón. Cuando un usuario lo selecciona, este llama a la función y ejecuta el código que contenga que, en este caso, tiene la opción de destroy.



```
#Creamos el gráfico en Tkinter
canvas = FigureCanvasTkAgg(fig,master = window)
canvas.draw()
canvas.get_tk_widget().pack()
toolbar = NavigationToolbar2Tk(canvas,window)
toolbar.update()
canvas.get_tk_widget().pack(side=TOP, fill=BOTH, expand=0)
def Limpiar1():
    canvas.get_tk_widget().destroy()
btn2.destroy()

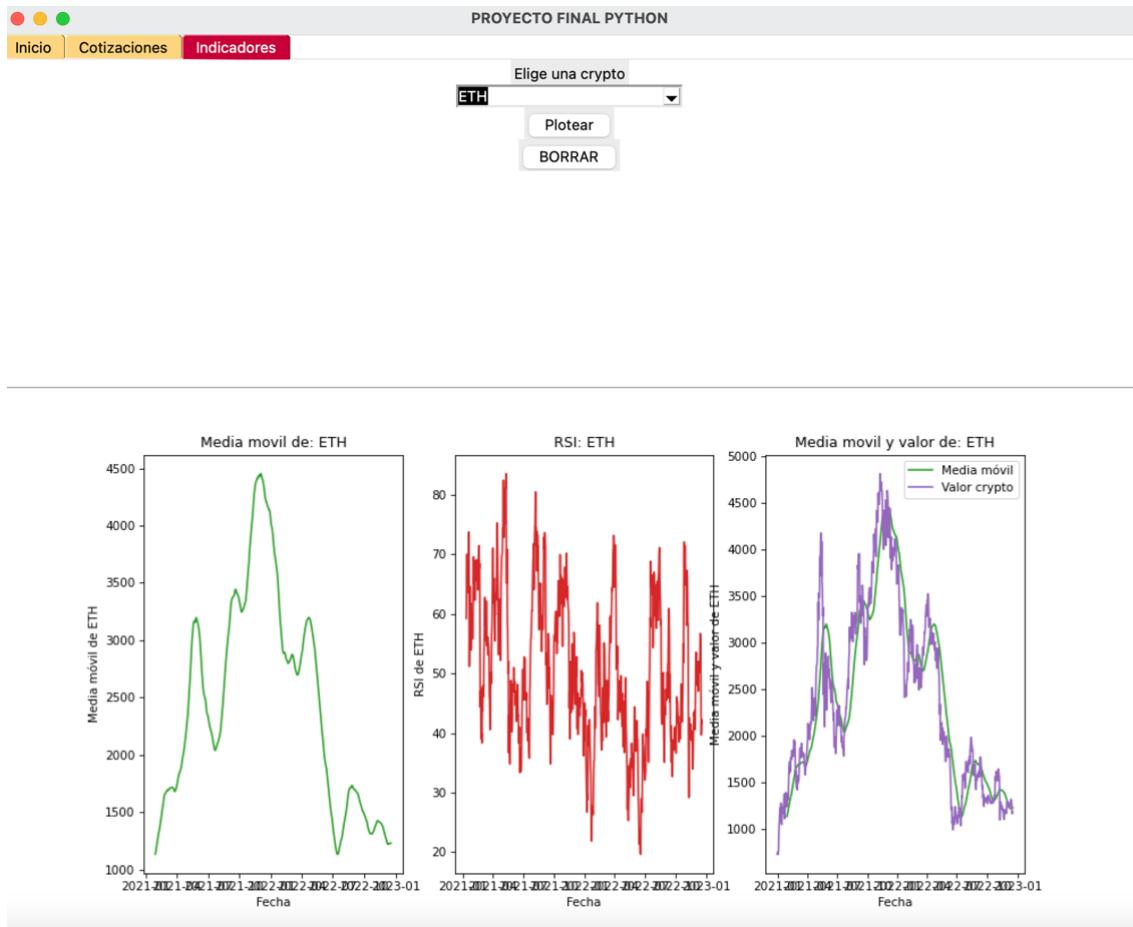
#Boton y función para eliminar los resultados anteriores
btn2 = Button(tab2, text="BORRAR", command=Limpiar1)
btn2.pack(anchor=CENTER)

except:
    messagebox.showinfo( 'Error', 'Selecciona una o más cryptos')

#Boton para ejecutar la elección
btn1 = Button(tab2, text="Ejecutar", command= cotizacion)
btn1.pack(anchor=CENTER)
```

2.2.3 Tercera Pestaña

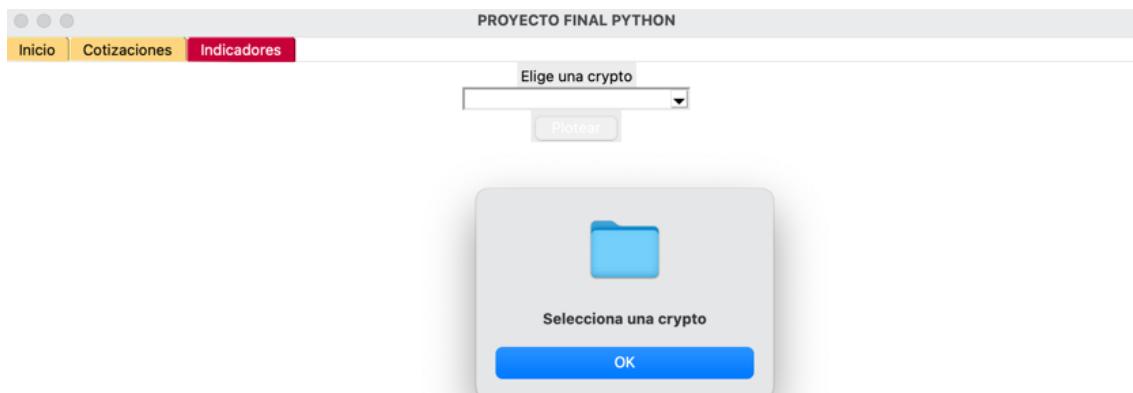
En la última pestaña se encuentran representadas las métricas (rsi y media móvil) de las cryptos monedas. En esta la elección es única, es decir, solo se pueden representar los valores de una única moneda mediante una lista desplegable. Como se puede observar en la siguiente imagen, existe un botón de plotear/ejecutar y de borrar que tienen la misma función que en la 2º pestaña. Existen 3 figuras, la media móvil, rsi y la combinación del valor de la moneda y su media móvil.



Si el usuario quiere ver las métricas de otra moneda, únicamente le dará al botón de borrar y podrá seleccionar otra, teniendo este aspecto.



En esta pestaña ocurre exactamente lo mismo que en la anterior, si la hora de plotear las métricas el usuario no selecciona ninguna moneda, saltará un mensaje de error.



A continuación se procede a la explicación del código de Python que es capaz de ejecutar/realizar las acciones mencionadas anteriormente. En primer lugar, se define la lista desplegable que ofrece Tkinter, Combobox. Esta función tiene que contener los siguientes parámetros: a qué pestaña se quiere introducir y cuáles son los valores de las etiquetas/textos.

```
#Tercera pestaña. Plot de las medias móviles y RSI de las distintas cryptos
label4= Label(tab3, text="Elige una crypto")
label4.pack(anchor=CENTER)

combo= ttk.Combobox(tab3,state="readonly",
                     values=["ETH", "BTC", "LTC", "USDT", "USDC", "DOGE"])
combo.pack(anchor=CENTER)
combo.current(None)
```

Posteriormente, se define una función denominada gráfica, que se encarga de calcular la media móvil de la moneda seleccionada. Para esta métrica se utiliza la librería pandas para poder utilizar la función Rolling, la cual calcula la media móvil. Además, se crea un bucle for para conseguir que las fechas del eje tengan la misma dimensión que está métrica.

```
#####
#Media móvil de la crypto selecciona mediante el combobox

def grafica():
    try:
        fig= Figure(figsize = (20, 20),
                    dpi = 100)
        ax= fig.subplots(1,3)

        n=30
        seleccion= str(combo.get())
        dat=pd.Series(dfFinal[seleccion]).rolling(window =n).mean().iloc[n-1: ].values

        fecha_inicio= datetime.now() - timedelta(days=719)
        fecha_hoy= datetime.now()

        datos1=[]
        contador=0
        for i in range(720):
            if i>=720-n:
                contador +=1
                if contador==1:
                    datos1.insert(0,None)
                else:
                    datos1.insert(1,None)
            else:
                datos1.append(dat[i])

        lista_fechas = [fecha_inicio + timedelta(days=d) for d in range((fecha_hoy - fecha_inicio).days + 1)]
        dfFinal["Fecha"] = lista_fechas
```

Dentro de la función anterior, se encuentra otra función denominada rsi que, como indica su nombre, se encarga de calcular la métrica rsi de la moneda seleccionada. Para el cálculo de esta se utilizan varios funciones pero las más importantes son up.ewm y down.ewm, que son los coeficientes claves para aplicar la función de esta métrica= $100 - \frac{100}{\left(1 + \frac{\text{up.ewm}}{\text{down.ewm}}\right)}$



```
#####
def rsi(periods = 14, ema = True):
    close_delta = dfFinal[seleccion].diff()
    up = close_delta.clip(lower=0)
    down = -1 * close_delta.clip(upper=0)

    if ema == True:
        ma_up = up.ewm(com = periods - 1, adjust=True, min_periods = periods).mean()
        ma_down = down.ewm(com = periods - 1, adjust=True, min_periods = periods).mean()
    else:
        ma_up = up.rolling(window = periods, adjust=False).mean()
        ma_down = down.rolling(window = periods, adjust=False).mean()

    rsi1= ma_up / ma_down
    rsi1= 100 - (100/(1 + rsi1))

#####
# Gráfico de RSI #####
ax[1].plot(dfFinal.Fecha, rsi1, color = 'tab:red')
ax[1].set_title(f"RSI: {seleccion}")
ax[1].set_xlabel("Fecha")
ax[1].set_ylabel(f"RSI de {seleccion}")

rsi()
```

Por último, se encuentra la figura donde se representa los valores de la moneda seleccionada y sus media móvil. Como anteriormente ya se había representado los valores y el cálculo de la media móvil, no hay misterio. Se crea una nueva subfigura y se ajusta su dimensionamiento. Además, se crea la función de destroy de la figura en su conjunto y se realiza el except, como en la 2º página.

```
#####
# Gráfico de las medias móviles #####
ax[0].plot(dfFinal.Fecha, datos1, color = 'tab:green')
ax[0].set_title(f"Media móvil de: {seleccion} ")
ax[0].set_xlabel("Fecha")
ax[0].set_ylabel(f"Media móvil de {seleccion}")

#####
# Gráfico de las medias móviles junto a sus valores#####
ax[2].plot(dfFinal.Fecha, datos1, color = 'tab:green', label="Media móvil")
ax[2].plot(dfFinal.Fecha, dfFinal[seleccion], color = 'tab:purple', label="Valor crypto")
ax[2].set_title(f"Media móvil y valor de: {seleccion} ")
ax[2].set_xlabel("Fecha")
ax[2].set_ylabel(f"Media móvil y valor de {seleccion}")
ax[2].legend()

#####
# Imprimir los 3 gráficos en tkinter #####
canvas = FigureCanvasTkAgg(fig,master = window)
canvas.draw()
canvas.get_tk_widget().pack()
toolbar = NavigationToolbar2Tk(canvas,window)
toolbar.update()
canvas.get_tk_widget().pack(side=TOP, fill=BOTH, expand=0)

def Limpiar1():
    canvas.get_tk_widget().destroy()
    btn2.destroy()

#Botón y función para eliminar los resultados anteriores
btn2 = Button(tab3, text="BORRAR", command=Limpiar1)
btn2.pack(anchor=CENTER)

except:
    messagebox.showinfo('Error', 'Selecciona una crypto')

btn3 = Button(tab3, text="Plotear", command= grafica)
btn3.pack(anchor=CENTER)
```



Para el correcto funcionamiento del código creado y su representación en 3 pestañas es necesario introducir la sintaxis siguiente para que se ejecute en el propio ordenador.

```
#Ejecutar la pestaña
tab_control.pack(expand=1,fill='both')
window.mainloop()
```

5. Integración Cloud (AWS)

Para un mejor flujo de trabajo, el programa aquí propuesto se ha implementado en Amazon Web Services. Dado que este proyecto no es de grandes dimensiones, se ha podido utilizar sin problema la capa gratuita de AWS.

El servidor en la nube utilizado ha sido del grupo EC2 que tiene Amazon a nuestra disposición.



Primero, hemos creado una sencilla máquina virtual con el sistema operativo Debian.

▼ Detalles de la instancia Información	
Plataforma	ID de AMI
<input checked="" type="checkbox"/> Debian (Inferrido)	<input checked="" type="checkbox"/> ami-0136e0e [REDACTED]
Detalles de la plataforma	Nombre de AMI
<input checked="" type="checkbox"/> Linux/UNIX	<input checked="" type="checkbox"/> debian-11-amd64-20221205-1220
Detener la protección	Hora de lanzamiento
desactivado	<input checked="" type="checkbox"/> Fri Dec 09 2022 11:12:50 GMT+0100 (hora estándar de Europa central) (10 days)
Recuperación automática de instancias	Ciclo de vida
Predeterminada	normal
Índice de lanzamiento de AMI	Nombre del par de claves
0	<input checked="" type="checkbox"/> P_Python
Especificación de crédito	ID de kernel
standard	-
Operación de uso	ID de disco RAM
<input checked="" type="checkbox"/> Runinstances	-
Compatibilidad con enclaves	Modo de arranque
-	-
Comportamiento de detención de hibernación desactivado	
Motivo de transición de estado	
- Mensaje de transición de estado	
Propietario	
<input checked="" type="checkbox"/> 51067 [REDACTED]	
Permitir etiquetas en los metadatos de la instancia desactivado	

Para las opciones de conexión no se ha puesto ningún firewall, dado que no hay indicios que haya alguien interesado en acceder a nuestro servidor con intención maliciosa.

Resumen de instancia de i-0bed9726b8e9034ed (Proyecto Python) Información	
Se ha actualizado hace less than a minute	Conectar Estado de la instancia ▾ Acciones
ID de la Instancia	Direcciones IPv4 privadas
<input checked="" type="checkbox"/> i-0bed9726b8e9034ed (Proyecto Python)	<input checked="" type="checkbox"/> 172.31.3 [REDACTED]
Dirección IPv6	DNS de IPv4 pública
-	<input checked="" type="checkbox"/> ec2-3-66-188-62.eu-central-1.compute.amazonaws.com dirección abierta
Tipo de nombre de anfitrión	
Nombre de IP: ip-172-31-38-133.eu-central-1.compute.internal	
Responder al nombre DNS de recurso privado	
IPv4 (A)	
Dirección IP asignada automáticamente	
<input checked="" type="checkbox"/> 3.66.1 [REDACTED] lica]	
Rol de IAM	
-	
Direcciones IP elásticas	
-	
Hallazgo de AWS Compute Optimizer	
<input checked="" type="checkbox"/> Suscríbase a AWS Compute Optimizer para recibir recomendaciones. Más información	
Nombre del grupo de Auto Scaling	
-	

Instancias (1) Información										
<input type="checkbox"/> Buscar instancia por atributo o etiqueta (case-sensitive)										
Name	ID de la instancia	Estado de la i...	Tipo de inst...	Comprobación ...	Estado de la ...	Zona de dispon...	DNS de IPv4 pública	Dirección IP...	IP elástica	Lanzar instancias
<input checked="" type="checkbox"/> Proyecto Python	<input checked="" type="checkbox"/> i-0bed9726b8e9034ed	<input checked="" type="checkbox"/> En ejecución	<input checked="" type="checkbox"/> t2.micro	<input checked="" type="checkbox"/> 2/2 comprobador	Sin alarmas	<input checked="" type="checkbox"/> eu-central-1b	<input checked="" type="checkbox"/> ec2-3-66-188-62.eu-ce...	<input checked="" type="checkbox"/> 3.66.188.62	-	<input type="button" value="Lanzar instancias"/>

Una vez creada la instancia, nos descargamos las claves privadas para poder acceder de manera remota al servidor mediante el protocolo ssh. Gracias a esto, podemos realizar cambios en tiempo real en el código.