

# Programación Avanzada de Computadoras

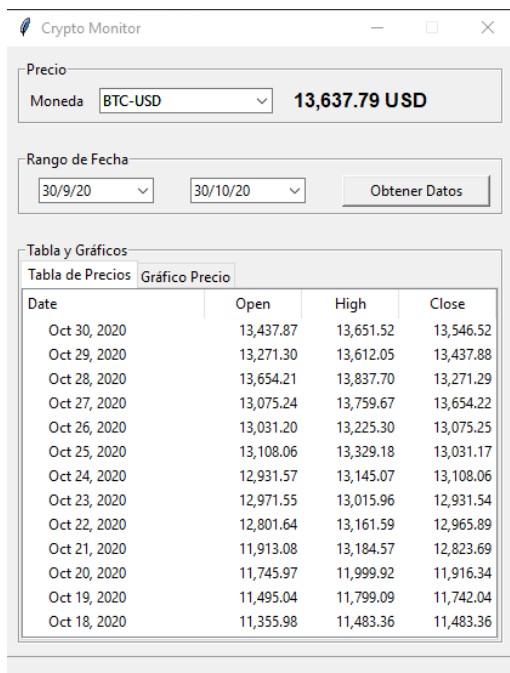
## Laboratorio Calificado 2

2020-2

Luis A. Muñoz

### Objetivo:

Debe de diseñar y codificar una aplicación en Python, utilizando la librería *tkinter*, *pandas*, *BeautifulSoup*, *tkcalendar*, *textwraper*, *requests*, *threading*, *datetime*, *matplotlib* y las que considere pertinentes, que registre el estado de un listado de criptomonedas, considerando su valor actual y sus registros históricos.



Date	Open	High	Close
Oct 30, 2020	13,437.87	13,651.52	13,546.52
Oct 29, 2020	13,271.30	13,612.05	13,437.88
Oct 28, 2020	13,654.21	13,837.70	13,271.29
Oct 27, 2020	13,075.24	13,759.67	13,654.22
Oct 26, 2020	13,031.20	13,225.30	13,075.25
Oct 25, 2020	13,108.06	13,329.18	13,031.17
Oct 24, 2020	12,931.57	13,145.07	13,108.06
Oct 23, 2020	12,971.55	13,015.96	12,931.54
Oct 22, 2020	12,801.64	13,161.59	12,965.89
Oct 21, 2020	11,913.08	13,184.57	12,823.69
Oct 20, 2020	11,745.97	11,999.92	11,916.34
Oct 19, 2020	11,495.04	11,799.09	11,742.04
Oct 18, 2020	11,355.98	11,483.36	11,483.36

Su aplicación debe de monitorear el precio en línea de un conjunto de pares de criptomonedas contra el dólar, y mostrar con colores (rojo/verde/negro) si es que la última variación es negativa, positiva o si no ha habido variación. Así también, debe de poder mostrar actualización diaria en forma de tabla con el resumen de los precios en un rango de fechas y la curva de movimiento de precio, en donde el rango de fechas se ingresará utilizando un control gráfico de calendario.

### Entregable:

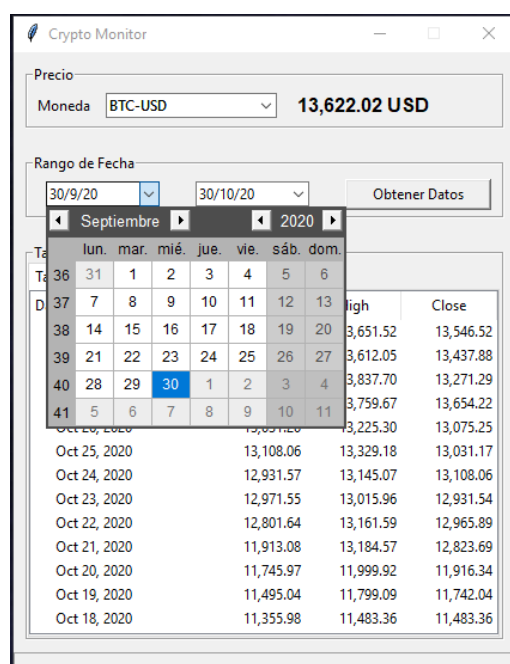
Archivo script criptomonitor.py

### Detalles del GUI:

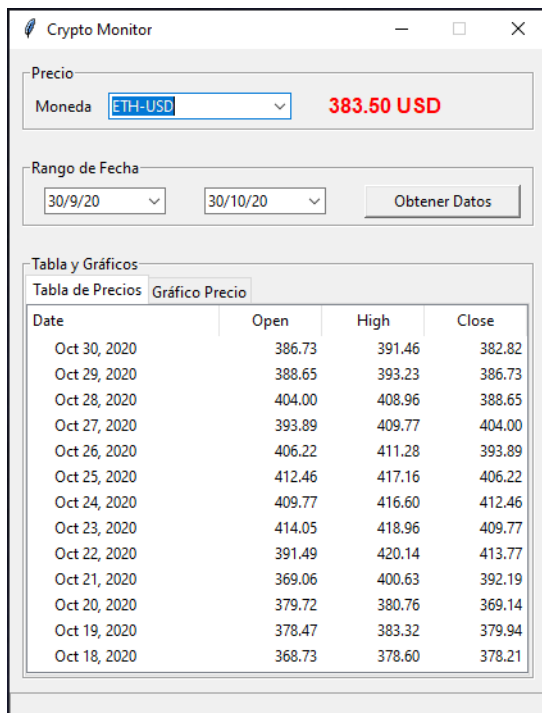
#### *LABEL FRAME: TOKEN*

Este frame muestra una caja de selección que muestra un listado de criptomonedas de las que se pueda obtener información de algún REST API que retorne un JSON. Puede consultar alguna fuente que le resulte conveniente, aunque en la sección Recursos se citan las fuentes que se pueden utilizar para obtener esta información.

En la sección derecha se mostrará el precio con una actualización de 5 segundos, en donde el texto se mostrará de color rojo si la nueva actualización de precio arroja un valor menor al anterior (lo que indica que el precio ha bajado) o de color verde si la nueva actualización de precio arroja un valor mayor al anterior (lo que indica que el precio ha subido) o de color negro si no ha habido variación.



Date	Open	High	Close
Oct 30, 2020	13,437.87	13,651.52	13,546.52
Oct 29, 2020	13,271.30	13,612.05	13,437.88
Oct 28, 2020	13,654.21	13,837.70	13,271.29
Oct 27, 2020	13,075.24	13,759.67	13,654.22
Oct 26, 2020	13,031.20	13,225.30	13,075.25
Oct 25, 2020	13,108.06	13,329.18	13,031.17
Oct 24, 2020	12,931.57	13,145.07	13,108.06
Oct 23, 2020	12,971.55	13,015.96	12,931.54
Oct 22, 2020	12,801.64	13,161.59	12,965.89
Oct 21, 2020	11,913.08	13,184.57	12,823.69
Oct 20, 2020	11,745.97	11,999.92	11,916.34
Oct 19, 2020	11,495.04	11,799.09	11,742.04
Oct 18, 2020	11,355.98	11,483.36	11,483.36



#### LABEL FRAME: RANGO FECHA

Esta sección contendrá los controles DateEntry de la librería *tkcalendar*, para especificar el rango de fecha con la evaluación de los datos.

**Por defecto, especifica los últimos treinta días.**

También habrá un botón que disparará la actualización de los recursos disponibles en el frame de pestañas múltiples debajo de este.

#### LABEL FRAME: TABLAS Y GRAFICOS

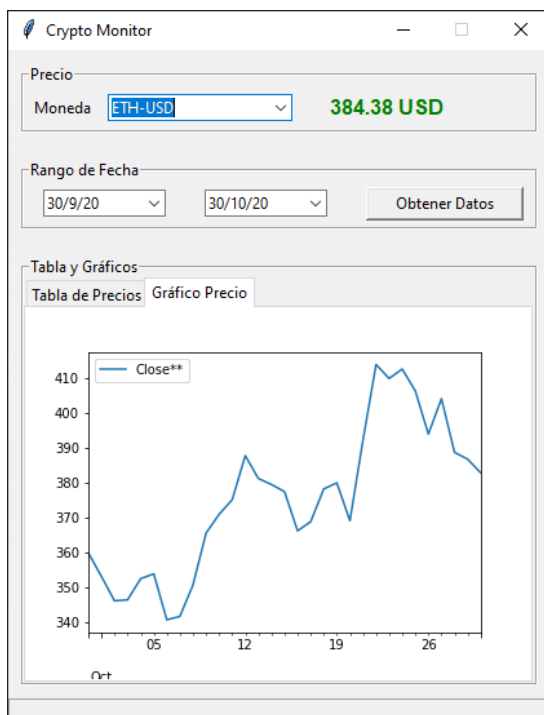
Esta sección presenta una colección de pestañas (*ttk.Notebooks*) que mostrará información pertinente sobre los datos seleccionados en las secciones superiores:

**Tabla:** Una tabla (*ttk.Treeview*) con la información de los precios diarios en el rango de fechas seleccionado en la frame de Rango Fecha y activado por el botón. Debe de mostrar información obtenida de una fuente actualizada (puede obtener información directamente de una fuente de la web) que consigne los precios por día con los detalles de variación diaria y precio de Apertura y Cierre.

**Gráfico:** Debe de mostrar un gráfico de la variación del precio de Cierre (Close\*\*) dentro del rango de fecha seleccionado en el frame de Rango Fecha y activado por el botón. Utilice un *TimeSeries (índice)* de *pandas* para obtener un gráfico con información en el tiempo en el eje vertical. Este gráfico es estático.

#### Condiciones de Operación:

Cuando la aplicación inicia, debe de estar seleccionado por defecto el primer par crypto-USD (esto es, BTC-USD), con lo que se mostrará el precio actual del Bitcoin contra el dólar, y la información histórica de los últimos treinta



días en la tabla y la variación de precios en la gráfica.

Cuando se actualice el Combobox por otra moneda, todos los datos (precio, tabla y gráfica) deben actualizarse (esto puede tomar algunos segundos, dependiendo de la conexión).

Si se hace click sobre el botón "Obtener Datos", se actualiza la tabla y los gráficos con los valores en el nuevo rango de fechas.

La barra de status indica las opciones de control sobre el Combobox y los DateEntry de las fechas, así como del botón de “Actualizar Datos” (especifique textos informativos de la función de estos widgets).

## Rúbrica de calificación

Descripción del alcance	Puntaje
La aplicación presenta una interfase limpia y organizada y cumple con los estándares de una aplicación gráfica de Windows.	6 pts
La aplicación muestra información en tiempo real en el Label Frame Token con variación de colores para indicar la tendencia.	4 pts
La sección de Rango Fecha permite seleccionar los datos pertinentes, así como iniciar con el rango de fecha de los últimos 30 días.	2 pts
La pestaña Tabla muestra la información de las transacciones en el rango de fecha seleccionado y se actualiza con el botón.	4 pts
La pestaña Gráfico muestra el gráfico con la variación de precios en el tiempo con fechas en el eje x y se actualiza con el botón.	4 pts

## Recursos

- REST API con los precios de Bitcoin (BTC) contra el dólar (par BTC-USD) en Coinbase: <https://api.coinbase.com/v2/prices/BTC-USD/spot>
- REST API con los precios de Ethereum (ETH) contra el dólar (par ETH-USD) en Coinbase: <https://api.coinbase.com/v2/prices/ETH-USD/spot>
- REST API con los precios de Litecoin (LTC) contra el dólar (par LTC-USD) en Coinbase: <https://api.coinbase.com/v2/prices/LTC-USD/spot>
- REST API con los precios de BitcoinCash (BTC) contra el dólar (par BCH-USD) en Coinbase: <https://api.coinbase.com/v2/prices/BCH-USD/spot>
- Información general del mercado de criptomonedas: <https://coinmarketcap.com/>

En el link anterior puede explorar los precios históricos de las monedas contra el dólar. Analice el URL para cada par de intercambio y la información de fecha. Por ejemplo:

- Información histórica de los precios de Bitcoin en los últimos 30 días: <https://coinmarketcap.com/currencies/bitcoin/historical-data/?start=20200930&end=20201030>

## Anexo técnico

### **tkcalendar**

La librería tkcalendar permite contar con un widget para ingresar un calendario en tkinter. La documentación está disponible en <https://pypi.org/project/tkcalendar/>

El widget a utilizar en este proyecto será DateEntry, que permite seleccionar una fecha en un calendario:

```
self.cal_ini = tkcalendar.DateEntry(frame)
self.cal_ini.grid(row=0, column=0, padx=5, pady=5)
```

Cuando se selecciona el DateEntry, se puede escoger una fecha (mes, día, año). Para leer el estado actual de la fecha:

```
self.cal_ini.get_date()
```

Lo que este método retorna es un objeto datetime.date().

Por otro lado, cuando se le asigna una fecha, de tal forma que el calendario ya tiene una fecha preasignada, se le carga un objeto datetime.date(). Por ejemplo, para asignarle la fecha del día de hoy:

```
self.cal_fin.set_date(datetime.datetime.now())
```

Apoyándose de los métodos de datetime, podrá utilizar este control en su aplicación.

## Notebook

Un Notebook es un Frame múltiple especial en tkinter que se apila en forma de pestañas.

Para utilizarlo se debe de proceder de la siguiente forma:

1. Definir el widget Notebook:
2. Crear los widgets o Frames que estarán en cada pestaña del Notebook.
3. Asignar cada widget o Frame en las pestañas del Notebook.
4. Colocar el Notebook con un Geometry Manager

Considere el siguiente código:

```
class App:
    def __init__(self, master):
        self.master = master

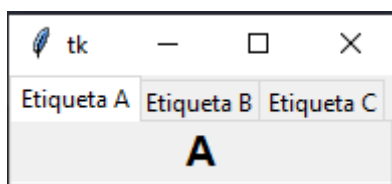
        # Paso 1
        self.notebook = ttk.Notebook(self.master)

        # Paso 2 (un Label en cada pestaña)
        self.lblA = tk.Label(self.notebook, text="A", font="Arial 16 bold")
        self.lblB = tk.Label(self.notebook, text="B", font="Arial 16 bold")
        self.lblC = tk.Label(self.notebook, text="C", font="Arial 16 bold")

        # Paso 3
        self.notebook.add(self.lblA, text="Etiqueta A")
        self.notebook.add(self.lblB, text="Etiqueta B")
        self.notebook.add(self.lblC, text="Etiqueta C")

        # Paso 4
        self.notebook.pack()

root = tk.Tk()
app = App(root)
root.mainloop()
```



Observe que no se gestiona con un Geometry Manager (pack o grid) los widgets o Frames que se colocan en el notebook: solo se crean y se asignan a la pestaña. En la aplicación se procederá de la misma forma: se crean dos pestañas, una tendrá un TreeView con la tabla de datos, en la otra un FigureCanvasTkAgg con el gráfico.