Código apicon.py

1. **Inicio del Programa:**
    - Se importan las bibliotecas necesarias: **requests**, **json**, y **datetime**.
2. **Definición de la Función writeFILE(texto, tipo, emp):**
    - Se define una función que escribe los datos proporcionados por la API en un archivo de texto.
    - Se convierte el tipo a cadena.
    - Se obtiene la fecha actual en formato 'dd-mm-yyyy'.
    - Se construye el nombre del archivo usando la fecha, el tipo de dato y otra información adicional.
    - Se intenta abrir el archivo en modo escritura ('w') y se escribe el texto en el archivo.
    - Si hay algún error, la función retorna 0.
3. **Definición de la Función makeRequest(url):**
    - Se define una función que realiza llamadas a la API.
    - Se intenta realizar una solicitud GET a la URL proporcionada.
    - Se comprueba si la respuesta tiene un código de estado HTTP 200.
    - Si la solicitud es exitosa, la respuesta se convierte a formato JSON.
    - Se manejan excepciones en caso de errores durante la solicitud.
    - Se retorna 0 en caso de error, o los datos en formato JSON.
4. **Definición de la Función crearlinkStock(func, symb):**
    - Se define una función que crea el nombre del enlace para solicitar datos de acciones.
    - La función convierte la función y el símbolo a mayúsculas.
    - Se construye el enlace dependiendo de la función y el símbolo proporcionados.
    - Retorna el enlace o 0 en caso de error.
5. **Definición de la Función crearlinkCurr(de, a):**
    - Se define una función que crea el nombre del enlace para solicitar tasas de cambio de moneda.
    - La función convierte las monedas de origen y destino a mayúsculas.
    - Se construye el enlace utilizando las monedas proporcionadas.

- Retorna el enlace.

6. **Definición de la Función procesarStock(json_data, func):**
    - Se define una función que procesa los datos de acciones según la función de la serie temporal especificada.
    - Se verifica la función deseada y se accede a los datos correspondientes en el formato JSON.
    - Se extraen las fechas y los valores de cierre de los datos.
    - Retorna una lista con las fechas y valores de cierre.
7. **Definición de la Función procesarCurr(jsonDatos):**
    - Se define una función que procesa los datos de tasas de cambio de moneda.
    - Se intenta extraer el valor numérico de '5. Exchange Rate' desde los datos JSON.
    - Retorna la tasa de cambio o **None** en caso de error.
8. **Fin del Programa:**
    - Fin del código.

Código modPia.py

1. **Inicio del Programa:**
    - Se importan las bibliotecas necesarias: **os** como **term** y **matplotlib.pyplot** como **plt**.
2. **Definición de la Función verOpc(archivo):**
    - Se define una función que permite al usuario visualizar la lista de empresas NASQAD y seleccionar un símbolo.
    - Se define otra función interna **mostrarListaNasqad(dict, numpag, tot, tamano=5)** para mostrar la lista de símbolos en páginas.
    - Se abre el archivo NASQAD y se crea un diccionario **symbols** con los símbolos y descripciones.
    - Se calcula el número total de páginas (**paginasTot**) para mostrar la lista en páginas.
    - Se inicia un bucle para mostrar la lista y recibir la entrada del usuario.
    - La función devuelve el símbolo seleccionado por el usuario.
3. **Definición de la Función validacion_Emp(emp, archivo='NASQAD.txt'):**

- Se define una función que valida si un símbolo de empresa está en la lista NASQAD.
- Se intenta abrir el archivo NASQAD y se obtiene la lista de símbolos.
- Se verifica si el símbolo proporcionado (**emp**) está en la lista de símbolos.
- La función devuelve **True** si el símbolo es válido, **False** si no lo es o si hay un error.

4. **Definición de la Función constGraf(lista):**
    - Se define una función que construye y muestra un gráfico de línea utilizando **matplotlib**.
    - Se extraen los intervalos y precios de la lista proporcionada.
    - Se crea el gráfico de línea utilizando **plt.plot**.
    - Se agregan etiquetas y título al gráfico.
    - Se muestra la gráfica con **plt.show()**.

5. **Definición de la Función estadísticas(precios de cierre):**
    - Se calcula la media usando .mean()**.**
    - Se calcula el estándar de desviación estándar usando .stdev().
    - Se calcula la volatilidad usando la desviación *(252**0.5).
    - Retorna cada una de las variables.
6. **Fin del Programa:**
    - Fin del código.

Código main.py

1. **Inicio del Programa:**
    - Se importan los módulos **os**, **apicon**, y **modPia**.
2. **Definición de la Función v_graficas(empresa):**
    - Se define la función **v_graficas** que toma el nombre de la empresa como argumento.
    - Se utiliza un bucle **while True** para mostrar opciones al usuario de visualización de gráficos de precios.
    - Se utiliza la función **input** para que el usuario seleccione una opción (**op**) y se realiza un manejo de flujo según la opción seleccionada.

3. **Manejo de Opciones en v_graficas:**
    - Si el usuario selecciona '1', se establece el tipo de gráfico como "DAILY".
    - Si el usuario selecciona '2', se establece el tipo de gráfico como "WEEKLY".
    - Si el usuario selecciona '3', se establece el tipo de gráfico como "MONTHLY".
    - Si el usuario selecciona '4', se sale de la función.
4. **Construcción del Enlace (link) y Consulta de Datos (datos):**
    - Se utiliza la función **apicon.crearlinkStock** para construir el enlace utilizando el tipo de gráfico y el nombre de la empresa.
    - Se utiliza la función **apicon.makeRequest** para hacer una solicitud a la API y obtener datos.
    - Los datos se procesan utilizando la función **apicon.procesarStock**.
5. **Escritura de Datos en un Archivo y Construcción de Gráficos:**
    - Se utiliza la función **apicon.writeFILE** para escribir los datos en un archivo.
    - Se utiliza la función **modPia.constGraf** para construir y mostrar un gráfico basado en los datos.
6. **Definición de la Función c_tipo_cambio:**
    - Se define la función **c_tipo_cambio** que permite al usuario consultar el tipo de cambio de una moneda.
    - Se utiliza un bucle **while True** para mantener la interacción con el usuario.
7. **Selección de Moneda y Construcción del Enlace (link):**
    - Se utiliza la función **modPia.verOpc** para permitir al usuario seleccionar una moneda.
    - Se utiliza la función **apicon.crearlinkCurr** para construir el enlace de la API para el tipo de cambio.
8. **Consulta de Datos y Procesamiento:**
    - Se utiliza la función **apicon.makeRequest** para hacer una solicitud a la API y obtener datos.
    - Los datos se escriben en un archivo utilizando **apicon.writeFILE**.
    - Se procesa el tipo de cambio utilizando **apicon.procesarCurr**.
9. **Impresión del Tipo de Cambio:**
    - Se imprime en la consola el tipo de cambio calculado.
10. **Consulta Adicional o Salida del Bucle:**

- Se le pregunta al usuario si desea consultar otro tipo de cambio.

- Se manejan diferentes respuestas posibles, y el bucle continúa o termina según la respuesta.

11. **Definición de la Función cambios_div:**
    - Se define la función **cambios_div** que permite al usuario realizar un cálculo de cambio de divisa.
12. **Selección de Moneda y Construcción del Enlace (link):**
    - Se utiliza la función **modPia.verOpc** para permitir al usuario seleccionar una moneda.
    - Se utiliza la función **apicon.crearlinkCurr** para construir el enlace de la API para el tipo de cambio.
13. **Consulta de Datos y Procesamiento:**
    - Se utiliza la función **apicon.makeRequest** para hacer una solicitud a la API y obtener datos.
    - Los datos se escriben en un archivo utilizando **apicon.writeFILE**.
    - Se procesa el tipo de cambio utilizando **apicon.procesarCurr**.
14. **Cálculo de Cambio de Divisa y Presentación al Usuario:**
    - Se solicita al usuario que ingrese el monto en MXN a cambiar a la otra divisa.
    - Se calcula el cambio multiplicando el monto por el tipo de cambio.
    - Se imprime en la consola el resultado del cálculo.
15. **Consulta Adicional o Salida del Bucle:**
    - Se le pregunta al usuario si desea realizar otro cálculo de cambio de divisa.
    - Se manejan diferentes respuestas posibles, y el bucle continúa o termina según la respuesta.
16. **Definición de la Función Pendiente datestadisticos(empresa):**

- while True: Inicia un bucle infinito para mantener el programa en ejecución hasta que el usuario decida salir.
- os.system('cls'): Limpia la pantalla para una presentación más limpia y legible en la consola.
- print("------Datos Estadisticos------"): Imprime un encabezado para indicar que se mostrarán datos estadísticos.
- print(" 1. Estadísticas Diarias"): Imprime la opción para ver estadísticas diarias.
- print(" 2. Estadísticas Semanales"): Imprime la opción para ver estadísticas semanales.
- print(" 3. Estadísticas Mensuales"): Imprime la opción para ver estadísticas mensuales.
- print(" 4. Salir"): Imprime la opción para salir del programa.

- op = input("Seleccione una opción: "): Solicita al usuario que seleccione una opción y almacena la entrada en la variable op.
- if op == '1':: Comprueba si la opción seleccionada es '1' (Estadísticas Diarias).
- tipo_periodo = "DAILY": Establece el tipo de periodo como "DAILY" si la opción es '1'.
- elif op == '2': Comprueba si la opción seleccionada es '2' (Estadísticas Semanales).
- tipo_periodo = "WEEKLY": Establece el tipo de periodo como "WEEKLY" si la opción es '2'.
- elif op == '3': Comprueba si la opción seleccionada es '3' (Estadísticas Mensuales).
- tipo_periodo = "MONTHLY": Establece el tipo de periodo como "MONTHLY" si la opción es '3'.
- elif op == '4': Comprueba si la opción seleccionada es '4' (Salir).
- os.system('cls'): Limpia la pantalla antes de salir y luego retorna de la función, finalizando el bucle infinito.
- else:: Si la opción no es ninguna de las anteriores, imprime un mensaje de error y continúa con la siguiente iteración del bucle.
- link = apicon.crearlinkStock(tipo_periodo, empresa): Crea un enlace usando la función crearlinkStock de la API con el tipo de periodo y la empresa.
- datos = apicon.makeRequest(link): Realiza una solicitud a la API utilizando el enlace y almacena los datos obtenidos.
- listas = apicon.procesarStock(datos, tipo_periodo): Procesa los datos utilizando la función procesarStock de la API.
- crear_excel(listas, tipo_periodo, empresa): Llama a la función crear_excel con los datos procesados, el tipo de periodo y la empresa.

17. Definición de la funcion crear_excel:

- Crear un DataFrame con los datos: Se crea un DataFrame de pandas utilizando los datos proporcionados como argumento.
- Obtener la fecha actual para incluirla en el nombre del archivo: Se obtiene la fecha y hora actuales en un formato específico (AñoMesDía_HoraMinutoSegundo) para incluir en el nombre del archivo de Excel.
- Crear un libro de trabajo de Excel: Se crea un objeto Workbook de la librería openpyxl, que representa un libro de trabajo de Excel.
- Añadir los datos al libro de trabajo: Se accede a la hoja activa del libro de trabajo.
- Se añaden los nombres de las columnas del DataFrame como encabezados de la hoja.
- Se recorren las filas del DataFrame y se añaden al libro de trabajo.
- Guardar el libro de trabajo como un archivo Excel: Se crea el nombre del archivo combinando el nombre de la empresa, el tipo de periodo y la fecha actual.
- Se guarda el libro de trabajo como un archivo Excel con el nombre especificado.

- Imprimir mensaje de éxito: Se imprime un mensaje indicando que el archivo de Excel ha sido creado con éxito.

18. **Definición de la Función main:**
    - Se define la función **main** que actúa como la función principal del programa.
    - Se utiliza un bucle **while True** para presentar un menú al usuario y realizar acciones según la opción seleccionada.
19. **Manejo de Opciones en main:**
    - Según la opción seleccionada por el usuario, se ejecutan diferentes funciones como **co_historial**, **v_graficas**, **datestadisticos**, **c_tipo_cambio**, **cambios_div**, o se sale del programa.
20. **Fin del Programa:**
    - El programa termina después de ejecutar todas las funciones y procesos definidos.