Práctica 4

Trabaje los problemas en archivos por separado de acuerdo con el problema planteado. **Para todos los casos, emplee manejo de errores y funciones según corresponda.**

Empleo de Librerías:

Problema 1:

Bitcoin es una forma de moneda digital, también conocida como criptomoneda. En lugar de depender de una autoridad central como un banco, Bitcoin se basa en una red distribuida, también conocida como cadena de bloques, para registrar transacciones.

En este problema debe generar un programa que realice:

- Solicite al usuario por línea de comando un valor de "n" el cual representa la cantidad de bitcoins que posee el usuario.
- Consulte la API del índice de precios de Bitcoin de CoinDesk en el siguiente link (https://api.coindesk.com/v1/bpi/currentprice.json), la cual retornará un objeto JSON, entre cuyas claves anidadas encontrará el precio actual de Bitcoin como un número decimal. Asegúrese de detectar cualquier excepción, como el siguiente código:

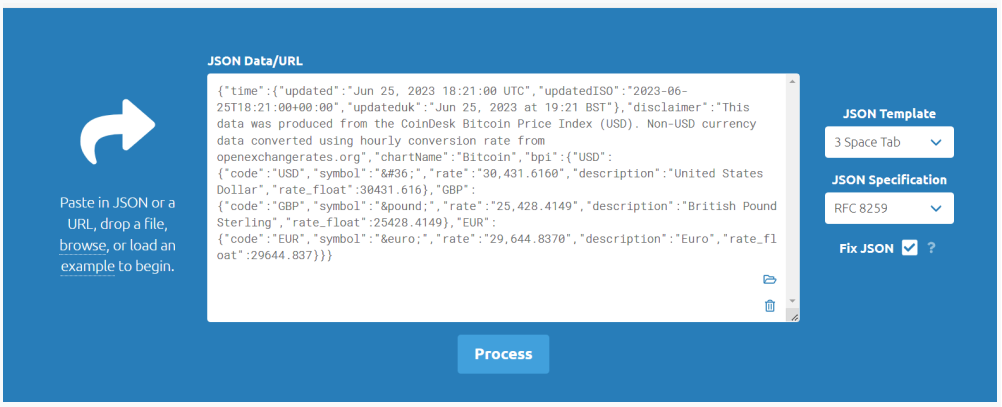
```
import requests

try:
    ...
except requests.RequestException:
    ...
```

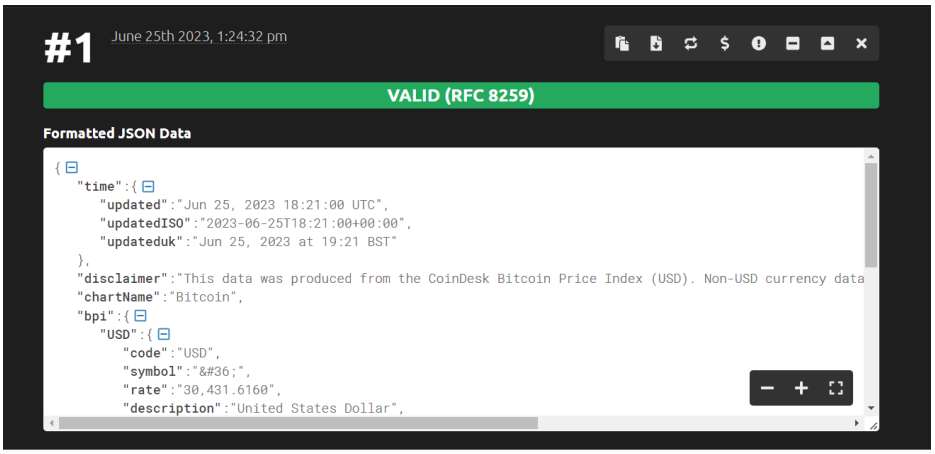
- Muestra el costo actual de "n" Bitcoins en USD con cuatro decimales, usando , como separador de miles.

Nota: El empleo de format string es apropiado para brindar formatos a nuestros datos. Le será de utilidad el siguiente comando: `print(f"${amount:,.4f}")`

Recuerde instalar la librería Requests mediante el comando: `pip install requests`

Puede apoyarse de un formateador de json para que le facilite la busqueda de información: https://jsonformatter.curiousconcept.com/#





Problema 2:

FIGlet, llamado así por las cartas de Frank, Ian y Glen, es un programa de principios de la década de 1990 para hacer letras grandes a partir de texto ordinario, una forma de arte ASCII:

- En la siguiente web puede ver una lista de fuentes admitidas por FIGlet figlet.org/examples.html
- Desde entonces, FIGlet ha sido portado a Python como un módulo llamado pyfiglet.

```
 _ _ _          _   _     _
| (_) | _____  | |_| |__ (_)___
| | | |/ / _ \ | __| '_ \| / __|
| | |   <  __/ | |_| | | | \__ \
|_|_|_|\_\___|  \__|_| |_|_|___/
```

Cree un programa el cual cumpla con las siguientes especificaciones:

- Solicite al usuario el nombre de una fuente a utilizar. En caso no sé ingrese ninguna fuente, su programa deberá seleccionar de forma aleatoria la fuente a utilizar.
- Solicite al usuario un texto.
- Finalmente, su programa deberá imprimir el texto solicitado usando la fuente apropiada.

Notas:

- Instalar la librería usando: `pip install pyfiglet`
- Para usar la librería, debe hacer:

```
from pyfiglet import Figlet
figlet = Figlet()
```

- Puede obtener la lista de fuentes disponibles usando: `figlet.getFonts()`
- Para seleccionar el fondo a utilizar emplee: `figlet.setFont(font=fuente_seleccionada)`
- Finalmente podrá imprimir el texto usando : `print(figlet.renderText(texto_imprimir))`
- Recuerde que random tiene un método random choice

Ejemplo:

- Con una fuente aleatoria y un texto “hello, world” podríamos obtener

```
  _          _ _                                  _       _
 | |__   ___| | | ___    __      _____  _ __| | __| |
 | '_ \ / _ \ | |/ _ \   \ \ /\ / / _ \| '__| |/ _` |
 | | | |  __/ | | (_) |   \ V  V / (_) | |  | | (_| |
 |_| |_|\___|_|_|\___( )   \_/\_/ \___/|_|  |_|\__,_|
                     |/
```

- Ingresando la fuente “rectangles” y texto “hello, world” obtenemos

Problema 3:

Del siguiente URL https://images.unsplash.com/photo-1546527868-ccb7ee7dfa6a?q=80&w=2070&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D

Descargue la imagen que más le agrade, según lo revisado en la clase. Posteriormente crear un programa que permita el almacenamiento de la imagen como un archivo zip. Finalmente cree un código que permita hacer un unzip al archivo zipeado.

Manejo de Archivos:

Problema 4:

Tienes un fichero **temperaturas.txt** que contiene registros de temperaturas diarias en formato CSV. Cada línea del fichero tiene la siguiente estructura: fecha,temperatura. Debes leer el fichero, calcular la temperatura promedio, la temperatura máxima y la mínima. Finalmente, debes escribir los resultados en un nuevo fichero **resumen_temperaturas.txt.**

link_fichero:
https://github.com/gdelgador/ProgramacionPython202407/blob/main/Modulo4/src/temperaturas.txt

Problema 5:

Escriba un programa que realice las siguientes tareas (Puede usar clases y/o funciones, también puede usar un menú para organizar su programa):

- Solicite un número entero entre 1 y 10 y guarde en un fichero con el nombre tabla-n.txt la tabla de multiplicar de ese número, donde n es el número introducido.
- Solicite un número entero entre 1 y 10, lea el fichero tabla-n.txt con la tabla de multiplicar de ese número, donde “n” es el número introducido, y la muestre por pantalla. Si el fichero no existe debe mostrar un mensaje por pantalla informando de ello.
- Solicite dos números n y m entre 1 y 10, lea el fichero tabla-n.txt con la tabla de multiplicar de ese número, y muestre por pantalla la línea m del fichero. Si el fichero no existe debe mostrar un mensaje por pantalla informando de ello.

Notas:

- Note que dentro del manejo de errores existe una excepción de tipo `FileNotFoundError` la cual le será de mucha utilidad.
- Revise los métodos de cadena
- Dentro del open() el método “readlines” le podría ser de utilidad.

Problema 6:

Una forma de medir la complejidad de un programa es contar su número de líneas de código (LOC), excluyendo las líneas en blanco y los comentarios. Por ejemplo, un programa como

```
# Say hello

name = input("What's your name? ")
print(f"hello, {name}")
```

Del código se observa que este solo tiene dos líneas de código, no cuatro, ya que su primera línea es un comentario y su segunda línea está en blanco (es decir, solo espacios en blanco). Esto no es tanto, por lo que es probable que el programa no sea tan complejo.

Implemente un programa donde se le solicitará al usuario la ruta de un archivo .py (nombre y ruta). Y retorne la cantidad de líneas de código de ese archivo, excluyendo los comentarios y líneas en blanco. Si el usuario ingresa una ruta inválida o si el nombre del archivo no termina en .py, su programa no retornará ningún resultado.

Notas:

- Note que dentro del manejo de errores existe una excepción de tipo `FileNotFoundError` la cual le será de mucha utilidad.
- Revise los métodos de cadena
- Dentro del open() el método "readlines" le podría ser de utilidad.

Ejemplo:

- archivo: "hello.py", número de líneas 2 (Asumiendo que hello.py es el código mostrado anteriormente)

BaseDatos:

Puede revisar: https://docs.hektorprofe.net/python/bases-de-datos-sqlite/

Problema 7:

Del ejercicio de Clase. Emplee el API de SUNAT que corresponda para obtener el precio de compra y venta del dólar durante todo el 2023. Almacene dicha información en base de datos sqlite 'base.db' con nombre de tabla sunat_info y en mongo db.

Finalmente deberá mostrar el contenido de dicha tabla.

Lee la documentación del API: https://apis.net.pe/api-tipo-cambio.html

Problema 8:

Empleando el ejercicio visto en clase de Procesamiento con Ficheros. Supongamos que el precio brindado en el archivo "`ventas.csv`" ha sido dato en dolares. Deberá solarizar el precio según la fecha de compra, para esto deberá leer la información almacenada de tipo de cambio de la base mongodb o sqlite (elegir una). Finalmente mostrar el precio total en dolares y soles por cada producto.

Problema 9 (Opcional):

Desarrollar el siguiente problema (Empleando modulo CSV):
https://github.com/gdelgador/ProgramacionPython202409/blob/main/Modulo4/Ejercicios/ProblemaBonus.ipynb