Trivia de números – Laboratorio 3

Instalar la librería request (api que devuelve datos curiosos sobre números).

En la terminal utilizar el comando: pip install requests

```
PS C:\Users\Paninatrix\Desktop\Laboratorio 3\pf-16-main> pip install requests
Defaulting to user installation because normal site-packages is not writeable
Collecting requests
```

Instalar la versión para Python 3 (evita errores con la versión)

Comando: pip3 install requests

```
PS C:\Users\Paninatrix\Desktop\Laboratorio 3\pf-16-main> pip3 install requests
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: requests in c:\users\paninatrix\appdata\local\packages\pythonsoftwarefoundation.p
ython.3.12_qbz5n2kfra8p0\localcache\local-packages\python312\site-packages (2.32.5)
```

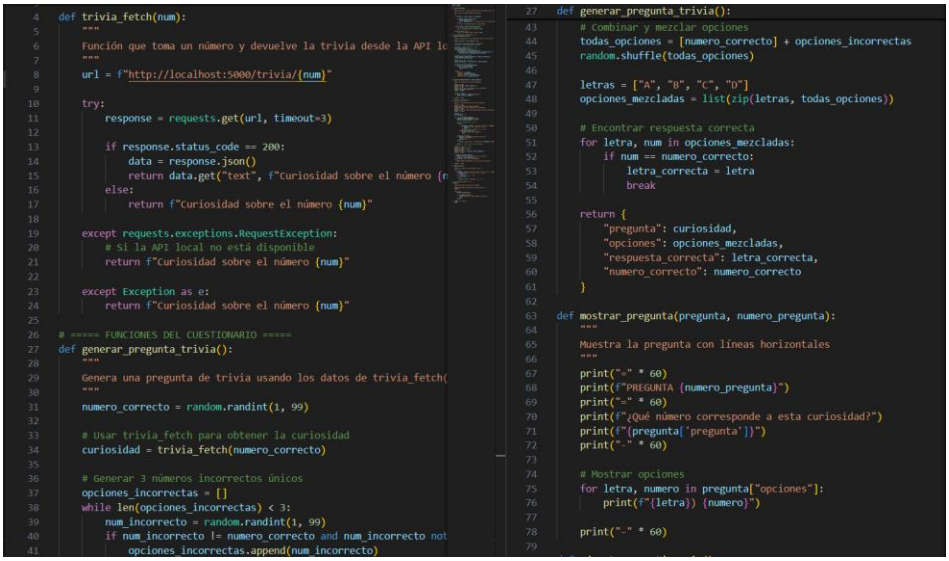
Instalar flask request (para creación de API local)

```
PS C:\Users\Paninatrix\Pictures\Repos_Alberto\Laboratorio3> pip install flask requests
Defaulting to user installation because normal site-packages is not writeable
Collecting flask
  Downloading flask-3.1.2-py3-none-any.whl.metadata (3.2 kB)
```

Reiniciar Visual Studio (para actualizar cambios).

Archivo main.py

El archivo es el controlador principal del programa importa los datos de la api y muestra al usuario un quizz de 4 preguntas, tomando números al azar y sincronizando con los índices en los datos de la api, muestra al usuario curiosidades y opciones de respuesta de forma automática, al final de las 4 pregunta muestra un resultado de los puntos obtenidos.



```python
def trivia_fetch(num):
    """
    Función que toma un número y devuelve la trivia desde la API lo
    """
    url = f"http://localhost:5000/trivia/{num}"

    try:
        response = requests.get(url, timeout=3)

        if response.status_code == 200:
            data = response.json()
            return data.get("text", f"Curiosidad sobre el número {n
        else:
            return f"Curiosidad sobre el número {num}"

    except requests.exceptions.RequestException:
        # Si la API local no está disponible
        return f"Curiosidad sobre el número {num}"

    except Exception as e:
        return f"Curiosidad sobre el número {num}"

# ===== FUNCIONES DEL CUESTIONARIO =====
def generar_pregunta_trivia():
    """
    Genera una pregunta de trivia usando los datos de trivia_fetch(
    """
    numero_correcto = random.randint(1, 99)

    # Usar trivia_fetch para obtener la curiosidad
    curiosidad = trivia_fetch(numero_correcto)

    # Generar 3 números incorrectos únicos
    opciones_incorrectas = []
    while len(opciones_incorrectas) < 3:
        num_incorrecto = random.randint(1, 99)
        if num_incorrecto != numero_correcto and num_incorrecto not
            opciones_incorrectas.append(num_incorrecto)
```

```python
def generar_pregunta_trivia():
    # Combinar y mezclar opciones
    todas_opciones = [numero_correcto] + opciones_incorrectas
    random.shuffle(todas_opciones)

    letras = ["A", "B", "C", "D"]
    opciones_mezcladas = list(zip(letras, todas_opciones))

    # Encontrar respuesta correcta
    for letra, num in opciones_mezcladas:
        if num == numero_correcto:
            letra_correcta = letra
            break

    return {
        "pregunta": curiosidad,
        "opciones": opciones_mezcladas,
        "respuesta_correcta": letra_correcta,
        "numero_correcto": numero_correcto
    }

def mostrar_pregunta(pregunta, numero_pregunta):
    """
    Muestra la pregunta con líneas horizontales
    """
    print("=" * 60)
    print(f"PREGUNTA {numero_pregunta}")
    print("=" * 60)
    print(f"¿Qué número corresponde a esta curiosidad?")
    print(f"{pregunta['pregunta']}")
    print("-" * 60)

    # Mostrar opciones
    for letra, numero in pregunta["opciones"]:
        print(f"{letra}) {numero}")

    print("-" * 60)
```

test_.py

Contiene pruebas automáticas para tu proyecto.
Verifica que la función trivia_fetch devuelve correctamente el número consultado en su respuesta.
Si el resultado es correcto para los números 42 y 1000, los tests pasan.

```python
from main import trivia_fetch

# Test 1
def test_trivia_42():
    assert trivia_fetch(42)["number"] == 42

# Test 2
def test_trivia_1000():
    assert trivia_fetch(1000)["number"] == 1000
```

App.py

API creado usando flask.
Toma los datos del archivo datos.json y permite su utilización mediante una conexión local.

```python
from flask import Flask, jsonify
import json
import random

app = Flask(__name__)

def cargar_datos_json():
    """Carga las curiosidades desde datos.json"""
    try:
        with open('datos.json', 'r', encoding='utf-8') as archivo:
            return json.load(archivo)
    except FileNotFoundError:
        return {"error": "Archivo datos.json no encontrado"}
    except json.JSONDecodeError:
        return {"error": "JSON corrupto"}

@app.route('/trivia/<int:numero>', methods=['GET'])
def obtener_trivia(numero):
    """Endpoint que devuelve la trivia para un número específico"""
    datos = cargar_datos_json()
    numero_str = str(numero)
```

```python
@app.route('/trivia/<int:numero>', methods=['GET'])
def obtener_trivia(numero):
    """Endpoint que devuelve la trivia para un número específico""
    datos = cargar_datos_json()
    numero_str = str(numero)

    if numero_str in datos:
        return jsonify({
            "number": numero,
            "text": datos[numero_str],
            "type": "trivia",
            "source": "api_local"
        })
    else:
        return jsonify({
            "error": f"No hay trivia para el número {numero}",
            "number": numero
        }), 404

if __name__ == '__main__':
    app.run(debug=True, port=5000, host='localhost')
```

datos.json

Archivo que contiene los datos de la API

```
  "1": "Primer número natural, base de todos los demás números",
  "2": "Único número par que es primo",
  "3": "Número mínimo de patas para que un mueble no cojee",
  "4": "Número de estaciones del año y puntos cardinales",
  "5": "Número de dedos en una mano",
  "6": "Número de lados en un cubo (hexaedro)",
  "7": "Días de la semana y maravillas del mundo antiguo",
  "8": "Número de bits en un byte",
  "9": "Número de planetas en el sistema solar (contando a Plutón)"
  "10": "Número de dedos en ambas manos",
  "11": "Jugadores en un equipo de fútbol soccer",
  "12": "Meses del año y horas en un reloj analógico",
  "13": "Considerado número de mala suerte en muchas culturas",
```

```
  "64": "Casillas en un tablero de ajedrez (8x8)",
  "65": "Edad de jubilación en muchos países",
  "66": "Ruta legendaria de los Estados Unidos",
  "67": "Años de la Revolución de Octubre (1917)",
  "68": "Porcentaje de agua en el cuerpo humano adulto",
  "69": "Número favorito de los adolescentes por su forma",
  "70": "Años de vida bíblicos (three score and ten)",
  "71": "Años del Imperio Romano",
  "72": "Grados en un ángulo de un pentágono regular",
  "73": "Número favorito de Sheldon Cooper en The Big Bang Theory",
  "74": "Años de la Guerra de Vietnam",
  "75": "Años de bodas de diamante",
  "76": "Trombones en la famosa canción de The Music Man",
```

Pasos realizados para el proyecto

1. Creación del archivo de datos (datos.json):
   Se hizo un archivo con curiosidades por números.

2. Instalación de dependencias:
   Se instalaron las librerías necesarias (flask y requests) usando pip.

3. Desarrollo de la API local (app.py):
   Se creó una API con Flask que lee el archivo datos.json y responde con la curiosidad de un número cuando se consulta.

4. Implementación de la función trivia_fetch:
   Esta función se encarga de hacer la petición a la API y devolver la información.

5. Desarrollo del programa (main.py):
   Se hizo un programa que generara un quiz basándose en 4 números aleatorios de la api generando opciones de respuesta para el usuario, otorgando puntos por responder correctamente y mostrando resultados después de las 4 preguntas.

6. Ejecución y prueba del sistema:
   Se inició la API local, se ejecuto el archivo main.py para comprobar el funcionamiento del software, al final sin terminar la ejecución de la API, se ejecuto el archivo test.py para comprobar resultados.

7. Creación de pruebas automáticas (test_.py):
   Se escribieron tests para verificar que la función trivia_fetch devuelve correctamente el número consultado.

Probar el test

```
[Running] python -u "c:\Users\Paninatrix\Desktop\Laboratorio 3\pf-l6-main\test.py"

[Done] exited with code=0 in 0.595 seconds
```