# db-report-01

May 29, 2024

[ ]:
```python
import json
import re

keys = set()  # TSV unique key set for headers (column names)


def flatten_json(nested_json, parent_key='', sep='.'):
    """
    Flatten the Pokémon's that are given as a json object
    """
    items = []
    for k, v in nested_json.items():
        new_key = f"{parent_key}{sep}{k}" if parent_key else k
        if isinstance(v, dict):
            items.extend(flatten_json(v, new_key, sep=sep).items())
            keys.add(new_key)
        elif isinstance(v, list):
            for i, item in enumerate(v):
                if isinstance(item, dict):
                    items.extend(flatten_json(item, f"{new_key}.{i}", sep=sep).
 ↪items())
                    keys.add(f"{new_key}.{i}")
                else:
                    items.append((f"{new_key}.{i}", item))
                    keys.add(f"{new_key}.{i}")
        else:
            if isinstance(v, str):
                match = re.search(r'\((.*?)\)', v)
                if match:
                    v = match.group(1)
            items.append((new_key, v))
            keys.add(new_key)
    return dict(items)


# open json file to get data and then flatten it
with open('pokedex.txt') as json_file:
```

```
    data = json.load(json_file)
    flattened_data = sorted([flatten_json(pokemon) for pokemon in data],␣
 ↪key=lambda x: x.get("name", ""))


# Filter out empty columns from each dictionary
flattened_data_filtered = [{key: value for key, value in pokemon.items() if key␣
 ↪in keys} for pokemon in flattened_data]

sorted_columns = sorted(keys, key=lambda x: (x != "name", x))

# Write to TSV file
with open('pokedex.tsv', 'w') as tsv_file:
    # Find the maximum width for each column
    max_widths = {header: max(len(header), max(len(str(pokemon.get(header,␣
 ↪''))) for pokemon in flattened_data_filtered)) for header in sorted_columns}

    # Write headers
    header_row = '\t'.join('{:<{width}}'.format(header,␣
 ↪width=max_widths[header]) for header in sorted_columns) + '\n'
    tsv_file.write(header_row)

    # Write data rows to TSV file
    for pokemon in flattened_data_filtered:
        row = '\t'.join('{:<{width}}'.format(str(pokemon.get(header, '')),␣
 ↪width=max_widths[header]) for header in sorted_columns) + '\n'
        tsv_file.write(row)

print("Data processed, done")
```