Data Formats Worksheet

Data

pokedex.json contains 801 *Pokémons*.
The data is in **JSON** format, the following example represents a Pokémon:

```
{
    "id": "001", ❶
    "name": "Bulbasaur",
    "species": "Seed Pokemon",
    "type": [ ❷
      "Grass",
      "Poison"
    ],
    "height": "2ft.4in. (0.71m)",
    "weight": "15.2 lbs (6.9 kg)",
    "abilities": [ ❸
      "Overgrow",
      "Chlorophyll"
    ],
    "stats": { ❹
      "hp": 45,
      "attack": 49,
      "defense": 49,
      "sp.atk": 65,
      "sp.def": 65,
      "speed": 45,
      "total": 318
    },
    "evolution": [ ❺
      "Bulbasaur",
      "Ivysaur",
      "Venusaur"
    ],
    "description": "For some time after its birth, it grows by gaining nourishment from the seed
on its back.",
    "gen": 1
 }
```

1. Pokémon ID, unique string (3 digits).
2. Pokémon type(s). Order-sensitive.
3. Pokémon abilities. Order-insensitive.
4. Pokémon numeric characteristics. Nested.
5. Pokémon evolution family. Order-sensitive.

Exercise

Convert the given **JSON** file to **TSV** (Tab separated value) file.

- The above JSON contains hierarchy (example: stats → hp).
  You should flatten this hierarchy by concatenating nested field names with a dot (stats.hp).
  (Think about what to do with dots within field names)
- Fields that contains arrays should be flatten in a similar way (using zero-based element index as "field name", i.e. `type.0`, `type.1`).
- Rows should be ordered by “name”, in an ascending order.
- Units should be only metric.

  For example, pounds is not metric, it is imperial. Kg is metric.

Guidelines

◓ Do not use any converting library (implement converting mechanism).
◓ You can use libraries to read\write files.
◓ Python 3.5+
◓ Your solution **can** be tailored for this specific data schema.

Extra Challenge

Try to implement a generative converter (using the same hierarchy transformation).
i.e. a program that reads **any** JSON file, with any schema, and converts it to a TSV.