DABBEL®

1.- WRONG ENCODING

Description

We are providing a file named 2002.json that contains the result of a BACnet scan from a controller that manages the HVAC systems. The content is formatted in JSON.
Our internal software that retrieves those values has incompatibilities with some controllers that use UTF-16 encoding because they are using wide characters and our software only reads one character per byte, generating text that looks like:


```
{
    "object_instance": 1,
    "properties": {
        "active-text": null,
        "cov-increment": null,
        "description": "\"\"",
        "inactive-text": null,
        "number-of-states": null,
        "object-identifier": "Vendor Proprietary Value:1",
        "object-name": "\".H.E.I.Z.G.R.U.P.P.E\"",
        "object-type": 130,
        "out-of-service": null,
        "present-value": null,
        "priority-array": null,
        "resolution": null,
        "state-text": null,
        "status-flags": null,
        "units": null
    }
}
```


We can identify 2 problems in the portion of data provided:

- *Heizgruppe* is separated by dots due to the character encoding incompatibility.
- The unwanted quotes that use escape characters should be removed. These only those that are at the beginning or at the end of the string.

Main goals

- Read and parse the JSON file in Python.
- Remove all the quotes that are at the beginning and the end of the strings (the ones that are using escape characters).
- Remove the key “out-of-service” from every object.
- Define a function that gets the string values, check if they are separated by dots, and returns a “cleaned” string *(check below to see some examples).*
- Export the JSON data into a new file.

DABBEL®

Further goals

- Using the unittest package:
    - Create unit tests to check the correct behavior of the functions defined before.
      For example, check what happens when you are parsing strings with 3 dots in a row or strings with dots in between of the characters but one dot is missing. Try to cover all the cases.
    - Check if the output JSON data has the same structure as the input, except for the modifications done in the process.

Additional requirements

- Use the style conventions defined in Python PEP 8.
- Change the style rules for the following cases:
    - Do not include whitespaces between symbols (=, +, - ...).
    - Use 2 spaces per indentation level.
    - Limit all lines to a maximum of 110 characters.

Test case examples

Below you will find a dictionary that contains some strings that must be used during the unit testing to check if these cases are covered.

The **keys** contain the **original string**, and the **values** have the **cleaned string**:

```
tests_dict = {
    "\".H.E.I.Z.G.R.U.P.P.E\"": "HEIZGRUPPE",
    "\".T.E.M.P.E.R.A.T.U.R.E...1.F...2.2.5\"": "TEMPERATURE.1F.225",
    ".H.E.I.Z.G.R.U.P.P.E": "HEIZGRUPPE",
    "H.E.I.Z.G.R.U.P.P.E.": "HEIZGRUPPE",
    "\"This text is using \"quotes\".\"": "This text is using \"quotes\".",
    "T.h.i.s. is a special test case": "T.h.i.s. is a special test case",
    ".............": ".......",
    "": "",
    "\"\"": ""
}
```