[CS 11 25.1] HOPE 2g – XML

Cheatsheet is available here: https://oj.dcs.upd.edu.ph/cs11cheatsheet/

Submit solution [CS 11 25.1] HOPE 2

Points: 220 (partial)
Time limit: 3.0s
Memory limit: 2G

Problem type

Allowed languages
py3

Problem Statement

Note. For this problem, you are expected to use Python's builtin `xml.etree.ElementTree` module.

XML is a popular file format used to represent data. In XML, data is represented using *tags*; each tag is one of two types:

- a pair consisting of a *opening tag* and a *closing tag*; such a tag may contain either a string or another tag.
- a single tag that contains zero or more *attributes*. Each attribute is of the form `key="value"`. Note that single tags do not contain strings or other tags.

You are given the contents of an `.xml` file, which looks like this:

```
<data>
    <student>
        <name>Hoshino</name>
        <attack>Piercing</attack>
        <armor>Heavy</armor>
        <height>145</height>
        <friend name="Shiroko"/>
        <friend name="Nonomi"/>
        <friend name="Serika"/>
        <friend name="Ayane"/>
        <friend name="Hina"/>
    </student>
    <student>
        <name>Hina</name>
        <armor>Heavy</armor>
        <attack>Explosive</attack>
        <height>142</height>
        <friend name="Ako"/>
        <friend name="Hoshino"/>
        <friend name="Chinatsu"/>
        <friend name="Iori"/>
    </student>
    <student>
        <name>Koyuki</name>
        <height>149</height>
        <attack>Mystic</attack>
        <armor>Heavy</armor>
    </student>
</data>
```

Note that the tags inside `student` tags can be in any order.

Turn this data into a dictionary that looks like this:

```
{
    "Hoshino": {
        "attack": "Piercing",
        "armor": "Heavy",
        "height": 145,
        "friends": ["Ayane", "Hina", "Nonomi", "Serika",
"Shiroko"],
    },
    "Hina": {
        "attack": "Explosive",
        "armor": "Heavy",
        "height": 142,
        "friends": ["Ako", "Chinatsu", "Hoshino", "Iori"],
    },
    "Koyuki": {
        "attack": "Mystic",
        "armor": "Heavy",
        "height": 149,
        "friends": [],
    },
}
```

The friends of each student must be listed in alphabetical order.

Task Details

Your task is to implement a function named `reshape_data`, which should start like this:

```
def reshape_data(s):
```

Here, `s` is a string referring to the contents of an XML file.

The function must return a dictionary corresponding to the reshaped data.

Restrictions

- You may import most functions from `xml.etree.ElementTree`.
- Loops and lists are allowed.
- Sets and dictionaries are allowed.
- Recursion is **disallowed**. (The recursion limit has been greatly reduced.)
- Generators and comprehensions are allowed.
- Your source code must have at most 900 bytes.

Examples

Example 1 Function Call

```
reshape_data(
    """\
<data>
    <student>
        <name>Hoshino</name>
        <attack>Piercing</attack>
        <armor>Heavy</armor>
        <height>145</height>
        <friend name="Shiroko"/>
        <friend name="Nonomi"/>
        <friend name="Serika"/>
        <friend name="Ayane"/>
        <friend name="Hina"/>
    </student>
    <student>
        <name>Hina</name>
        <armor>Heavy</armor>
        <attack>Explosive</attack>
        <height>142</height>
        <friend name="Ako"/>
        <friend name="Hoshino"/>
        <friend name="Chinatsu"/>
        <friend name="Iori"/>
    </student>
    <student>
        <name>Koyuki</name>
        <height>149</height>
        <attack>Mystic</attack>
        <armor>Heavy</armor>
    </student>
</data>"""
)
```

Example 1 Return Value

```
{
    "Hoshino": {
        "attack": "Piercing",
        "armor": "Heavy",
        "height": 145,
        "friends": ["Ayane", "Hina", "Nonomi", "Serika",
"Shiroko"],
    },
    "Hina": {
        "attack": "Explosive",
        "armor": "Heavy",
        "height": 142,
        "friends": ["Ako", "Chinatsu", "Hoshino", "Iori"],
    },
    "Koyuki": {
        "attack": "Mystic",
        "armor": "Heavy",
        "height": 149,
        "friends": [],
    },
}
```

Constraints

- The function `reshape_data` will be called at most 200 times.
- The number of entries in each call to `reshape_data` is at most 200.
- The given entries are valid.

Scoring

Note: New tests may be added and all submissions may be rejudged at a later time. (All future tests will satisfy the constraints.)

- You get 80 ❤️ $+ 140$ 🔴 points if you solve all test cases.

Clarifications

Report an issue

No clarifications have been made at this time.