Lecture 3A: Quotes example: More examples on using collection

Version 1.0. Prepared by Makzan.

In this example, we explore different ways to store multiple- collections. They are:

- List/Tuple with string spliting
- Nested List/Tuple
- Dictionary
- namedtuple

List and tuple

If we do not need to separate the multiple columns, we dont necessary need to separate them.

A list simply fulfils the job:

In [1]:
```
import random

quotes = (
    "I want to put a ding in the universe.—Steve Jobs",
    "Life is 10% what happens to you and 90% how you react to it.—Charles F
    "Family is not an important thing. It's everything.—Michael J. Fox",
    "Nothing is impossible, the word itself says 'I'm possible'!—Audrey Hep
    "There are two ways of spreading light: to be the candle or the mirror
    "Try to be a rainbow in someone's cloud.—Maya Angelou",
    "Be brave enough to live life creatively. The creative place where no c
    "The secret of getting ahead is getting started.—Mark Twain",
)

print( random.choice(quotes) )
```

```
There are two ways of spreading light: to be the candle or the mirror that
reflects it.—Edith Wharton
```

Using string split

Given that the data is only a list of string. Each string contains two data: the quote content and who was saying that. They are separated by a dash `—` .

We can use `split()` to split any string by given "separator".

For instance, we can use `split("—")` .

In [2]:
```
import random

quotes = (
    "I want to put a ding in the universe.—Steve Jobs",
    "Life is 10% what happens to you and 90% how you react to it.—Charles F
    "Family is not an important thing. It's everything.—Michael J. Fox",
    "Nothing is impossible, the word itself says 'I'm possible'!—Audrey Hep
    "There are two ways of spreading light: to be the candle or the mirror
    "Try to be a rainbow in someone's cloud.—Maya Angelou",
    "Be brave enough to live life creatively. The creative place where no c
    "The secret of getting ahead is getting started.—Mark Twain",
)

quote = random.choice(quotes).split("—")

print(f'{quote[1]} said: {quote[0]}')
```

```
Steve Jobs said: I want to put a ding in the universe.
```

Line splitting may be used often when we cannot control the data source. Here is the pros and cons of using string splitting:

Pros:

- Simple data structure

Cons:

- Separating the columns relies on the content. If there is more than one dash in the string, the logic breaks.

Aside: Joining string together

We can split string by using `split` . On the other hand, we can merge a list

In [2]:
```
string = "Steven said: Hello World"

string.split(": ")
```

Out[2]:
```
['Steven said', 'Hello World']
```

In [1]:
```
sample_list = ['Steven said', 'Hello World']

": ".join(sample_list)
```

Out[1]:
```
'Steven said: Hello World'
```

Nested list/tuple

In [3]:
```
import random


quotes = (
            ("I want to put a ding in the universe.", "Steve Jobs"),
            ("Life is 10% what happens to you and 90% how you react to it.'
            ("Family is not an important thing. It's everything.", "Michael
            ("Nothing is impossible, the word itself says 'I'm possible'!",
            ("There are two ways of spreading light: to be the candle or th
            ("Try to be a rainbow in someone's cloud.", "Maya Angelou"),
            ("Be brave enough to live life creatively. The creative place w
            ("The secret of getting ahead is getting started.", "Mark Twain
        )

quote = random.choice(quotes)

print(f'{quote[1]} said: {quote[0]}')
```

```
Edith Wharton said: There are two ways of spreading light: to be the candle
or the mirror that reflects it.
```

Pros and cons of nested list

Pros:

- We often meet this format when reading CSV or tabular data from external sources/files.
- Quite easy to use
- Separation not related to the content.

Cons:

- Correctness depends on the order of the list items.

Using Dictionary

In [2]:
```
import random

quotes = (
    {"quote":"I want to put a ding in the universe.", "source":"Steve Jobs'
    {"quote":"Life is 10% what happens to you and 90% how you react to it.'
    {"quote":"Family is not an important thing. It's everything.", "source'
    {"quote":"Nothing is impossible, the word itself says 'I'm possible'!",
    {"quote":"There are two ways of spreading light: to be the candle or th
    {"quote":"Try to be a rainbow in someone's cloud.", "source":"Maya Ange
    {"quote":"Be brave enough to live life creatively. The creative place w
    {"quote":"The secret of getting ahead is getting started.", "source":"M
)

quote = random.choice(quotes)
content = quote["quote"]
source = quote["source"]

print(f'{source} said: {content}')
```

```
Alan Alda said: Be brave enough to live life creatively. The creative place
where no one else has ever been.
```

Pros and cons of using dictionary:

Pros:

- Using key instead of order to improve code readability
- Easier to maintain

Cons:

- A typo in the key may causes error that is not easily identifiable.

A key typo can result in error.

In [3]:
```
quote["sooooource"]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-3-9197b19c2227> in <module>
----> 1 quote["sooooource"]

KeyError: 'sooooource'
```

We may avoid key error by using `.get` . But it still does not prevent typo defined from source.

In [4]:
```
quote.get("source")
```

Out[4]:
```
'Alan Alda'
```

In [7]:
```
quote.get("soooource", "Not found")
```

Out[7]:
```
'Not found'
```

Using namedtuple

In [5]:
```
import random
from collections import namedtuple

Quote = namedtuple("Quote", "content, source")

quotes = (
    Quote(content="I want to put a ding in the universe.", source="Steve Jo
    Quote(content="Life is 10% what happens to you and 90% how you react to
    Quote(content="Family is not an important thing. It's everything.", sou
    Quote(content="Nothing is impossible, the word itself says 'I'm possibl
    Quote(content="There are two ways of spreading light: to be the candle
    Quote(content="Try to be a rainbow in someone's cloud.", source="Maya A
    Quote(content="Be brave enough to live life creatively. The creative pl
    Quote(content="The secret of getting ahead is getting started.", source
)

quote = random.choice(quotes)
print(f'{quote.source} said: {quote.content}')
```

```
Alan Alda said: Be brave enough to live life creatively. The creative place
where no one else has ever been.
```

Pros and cons of using namedtuple

Pros

- The column named is pre-defined, so the usage is predictable.
- Typos will raise error at runtime, so it is easy to be identifiable.
- Using column name instead of order to improve code readability

Cons

- Less flexible than the previous solutions

❓ Pop Quiz

Question 1: Assuming now we want to store a list of names. Just names and nothing else. We may need to expand the list later. Which data structure shall we use?

1. tuple
2. list
3. dictionary
4. namedtuple

Question 2: Assuming now we want to store a list of student records. Each record contains first name, last name, and email. Which data structure shall we use?

1. tuple
2. dictionary
3. list of strings
4. list of dictionaries

Summary

In this section, we explored different way to store multi-columns data. It includes:

- List/tuple
- Nested list/tuple
- Dictionary
- namedtuple

In future, we will further use pandas DataFrame and reading data from CSV and Excel for tabular data.