Python Dictionary Comprehension

Dictionary comprehension is an elegant and concise way to create dictionaries.

Example

Consider the following code:

```
timesTwoDict = dict()
for num in range(1, 6):
    timesTwoDict[num] = num*2
print(timesTwoDict)
# {1: 2, 2: 4, 3: 6, 4: 8, 5: 10}
```

Now, let's create the dictionary in the above program using dictionary comprehension.

```
timesTwoDict = {num: num*2 for num in range(1, 6)}
print(timesTwoDict)
# {1: 2, 2: 4, 3: 6, 4: 8, 5: 10}
```

The output of both programs will be the same.

```
{1: 2, 2: 4, 3: 6, 4: 8, 5: 10}
```

In both programs, we have created a dictionary timesTwoDict with number times 2 key/value pair.

However, using dictionary comprehension allowed us to create a dictionary in a single line.

Syntax

From the above example, we can see that dictionary comprehension should be written in a specific pattern.

The minimal syntax for dictionary comprehension is:

dictionary = {*key*: *value* for *vars* in *iterable*}

Let's compare this syntax with dictionary comprehension from the above example.

key -> num

value -> num*2

vars -> num

iterable -> range(1,6)

Now, let's see how we can use dictionary comprehension using data from another dictionary.

Example

```
# harga item dalam rupiah
hargaLama = {'susu': 15000, 'teh': 10000, 'roti': 20000}

kenaikanHarga = 1.1
# harga naik 10%

hargaBaru = {item: value*kenaikanHarga for (item, value) in hargaLama.items()}
print(hargaBaru)
# {'susu': 16500.0, 'teh': 11000.0, 'roti': 22000.0}
```

Here, we can see that we retrieved the item prices in rupiah and increase the price by 10%. Using dictionary comprehension makes this task much simpler and shorter.

Conditionals in Dictionary Comprehension

We can further customize dictionary comprehension by adding conditions to it. Let's look at an example.

Example : if Conditional Dictionary Comprehension

```
umurDict = {'andi': 24, 'budi': 32, 'ceci': 23, 'dedi': 51}

umurGanjilDict = {key: val for (key, val) in umurDict.items() if val % 2 != 0}
print(umurGanjilDict)
# {'ceci': 23, 'dedi': 51}
```

As we can see, only the items with odd value have been added, because of the if clause in the dictionary comprehension.

Example : Multiple if Conditional Dictionary Comprehension

```
umurDict = {'andi': 24, 'budi': 32, 'ceci': 23, 'dedi': 51}

umurGanjilDict = {key: val for (key, val) in umurDict.items() if val % 2 != 0 if val > 30}
print(umurGanjilDict)
# {'dedi': 51}
```

In this case, only the items with an odd value of higher than 30 have been added to the new dictionary.

It is because of the multiple if clauses in the dictionary comprehension. They are equivalent to and operation where both conditions have to be true.

Example : if-else Conditional Dictionary Comprehension

```
umurDict = {'andi': 24, 'budi': 32, 'ceci': 23, 'dedi': 51}

tuaMudaDict = {key: ('tua' if val > 50 else 'muda') for (key, val) in umurDict.items()}
print(tuaMudaDict)
# {'andi': 'muda', 'budi': 'muda', 'ceci': 'muda', 'dedi': 'tua'}
```

In this case, a new dictionary is created via dictionary comprehension.

The items with a value of more than 50 have the value of 'tua' while others have the value of 'muda'.

Advantages of Using Dictionary Comprehension

As we can see, dictionary comprehension shortens the process of dictionary initialization by a lot. It makes the code more pythonic.

Using dictionary comprehension in our code can shorten the lines of code while keeping the logic intact.

Warnings on Using Dictionary Comprehension

Even though dictionary comprehensions are great for writing elegant code that is easy to read, they are not always the right choice.

We must be careful while using them as :

- They can sometimes make the code run slower and consume more memory.
- They can also decrease the readability of the code.

We must not try to fit a difficult logic or a large number of dictionary comprehension inside them just for the sake of making the code single lined. In these cases, It is better to choose other alternatives like loops.