Week 8

Dictionaries Python

Dictionary

- In real life, a dictionary is an object that contains words, and each word has a meaning associated with it.
- In Python a dictionary is also an object indexed by keys (words) that have associated values (meanings).

Dictionary

- Python dictionaries have the following characteristics:
  - They maintain the order in which the keys are inserted.
  - They are mutable, which allows them to add, delete and modify their elements.
  - Keys must be unique. Strings are often used as keys, but actually it could be any immutable data type: integers, floats, tuples (among others).
  - They have very quick access to their items, due to the way they are implemented internally

Creating

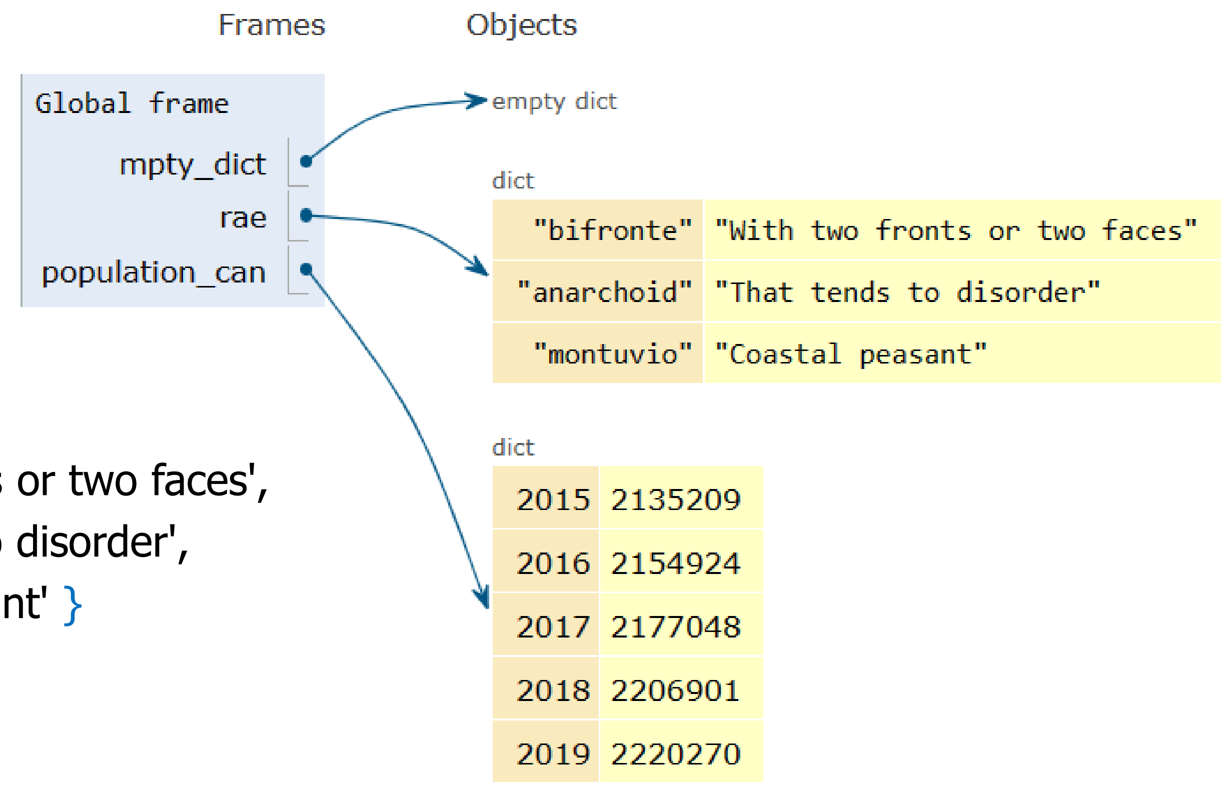
- To create a dictionary we use braces {} surrounding **key:value** assignments that are separated by commas

```
>>> empty_dict = {}
>>> rae = {
... 'bifronte': 'With two fronts or two faces',
... 'anarchoid': 'That tends to disorder',
... 'montuvio': 'Coastal peasant' }
>>> population_can = {
... 2015: 2_135_209,
... 2016: 2_154_924,
... 2017: 2_177_048,
... 2018: 2_206_901,
... 2019: 2_220_270
... }
```

```
>>> purse = dict{}
>>> purse['money'] = 12
>>> purse['candy'] = 3
>>> purse['tissues'] = 75
>>> print(purse)
{'money': 12, 'tissues': 75, 'candy': 3}

>>> print(purse['candy'])
3
>>> purse['candy'] = purse['candy'] + 2
>>> print(purse)
{'money': 12, 'tissues': 75, 'candy': 5}
```

The Python Interpreter



```
>>> empty_dict = {}
>>> rae = {
... 'bifronte': 'With two fronts or two faces',
... 'anarchoid': 'That tends to disorder',
... 'montuvio': 'Coastal peasant' }
>>> population_can = {
... 2015: 2_135_209,
... 2016: 2_154_924,
... 2017: 2_177_048,
... 2018: 2_206_901,
... 2019: 2_220_270
... }
```

Creating Dictionary

- It is possible to create a dictionary by specifying its keys and a single "padding" value

```
>>> dict.fromkeys('aeiou', 0)
{'a': 0, 'e': 0, 'i': 0, 'o': 0, 'u': 0}
```

Add or modify an element

- To **add** an element to a dictionary it is only necessary to refer to the key and **assign** it a value
- If the key already **existed** in the dictionary, the existing value is **replaced** with the new one
- If the key is new, it is added to the dictionary with its value. We're **not** going to get **a error** unlike lists.

Add or modify an element

```
>>> rae = {
... 'bifronte': 'With two fronts or two faces',
... 'anarchoid': 'That tends to disorder',
... 'montuvio': 'Coastal peasant' }
```

```
>>> rae['prosecute'] = 'Submit a matter for examination, discussion
and judgment`
>>> rae {'bifronte': 'Of two fronts or two faces',
'anarchoid': 'That tends to disorder',
'montuvio': 'Coastal peasant',
'prosecute': 'Submit a matter for examination, discussion and judgment'}
```

Creating from empty

```
>>> VOWELS = 'aeiou'
>>> enum_vowels = {}
>>> for i, vowel in enumerate(VOWELS, start=1):
... enum_vowels[vowel] = i
...
>>> enum_vowels {'a': 1, 'e': 2, 'i': 3, 'o': 4, 'u': 5}
```

Check exists

- It is an error to reference a key which is **not in** the dictionary
- We can use the in operator to see if a key is in the dictionary

```
>>> 'bifronte' in rae
True
>>> 'almohada' in rae
False
>>> 'montuvio' not in rae
False
```

Exercise

- Use dictionary, count the number of occurrences of each item in a list.

```
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
```

```
Counts = {}
        {'csev': 2, 'zqian': 1, 'cwen': 2}
```

Simplified Counting with get()

- We can use get() and provide a default value of zero when the key is not yet in the dictionary - and then just add one

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names :
    counts[name] = counts.get(name, 0)
print(counts)
```

Default

{'csev': 2, 'zqian': 1, 'cwen': 2}

Get all items

```
>>> rae.keys()
dict_keys(['two-faced', 'anarchoid',
'montuvio', 'prosecute'])
```

```
>>> for word in rae.keys():
... print(word)
bifronte
anarcoide
montuvio
enjuiciar
```

```
>>> rae.values()
dict_values([
'De dos frentes o dos caras',
'Que tiende al desorden',
'Campesino de la costa',
'Instruir, juzgar o sentenciar una
causa'
])
```

```
>>> for meaning in rae.values():
... print(meaning)
```

```
Two fronts or two faces t
hat tends to disorder
coast farmer Instruct, judge or
sentence a cause
```

Comparing Lists and Dictionaries

You can get a list of keys, values, or items (both) from a dictionary

```
>>> jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
>>> print(list(jjj))
['jan', 'chuck', 'fred']
>>> print(list(jjj.keys()))
['jan', 'chuck', 'fred']
>>> print(list(jjj.values()))
[100, 1, 42]
>>> print(list(jjj.items()))
[('jan', 100), ('chuck', 1), ('fred', 42)]
>>>
```

Exercise

```
>>> words = ('sun', 'space', 'rocket', 'earth')
>>> words_length = {word: len(word) for word in words}
???
```

```
>>> words = ('sun', 'space', 'rocket', 'earth')
>>> words_length = {w: len(w) for w in words if w[0] not in ‘aeiou’}
???
```

Exercise

1. Counting Words in Text
2. Counting Words in File
3. Display the longest word, frequency o f the word

Exercise Solution

```
name = input('Enter file:')
handle = open(name)

counts = dict()
    for line in handle:
        words = line.split()
    for word in words:
        counts[word] = 
counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count 
> bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

```
python words.py
Enter file: words.txt
to 16
```

```
python words.py
Enter file: clown.txt
the 7
```