Python Dictionaries

In Python, a dictionary is an *ordered* (from Python > 3.7) collection of `key`: `value` pairs.

Example Dictionary:

```
my_cat = {
    'size': 'fat',
    'color': 'gray',
    'disposition': 'loud'
}
```

values()

The `values()` method gets the **values** of the dictionary:

```
>>> pet = {'color': 'red', 'age': 42}
>>> for value in pet.values():
...     print(value)
...
# red
# 42
```

keys()

The `keys()` method gets the **keys** of the dictionary:

```
>>> pet = {'color': 'red', 'age': 42}
>>> for key in pet.keys():
...     print(key)
...
# color
# age
```

items()

The `items()` method gets the **items** of a dictionary and return them as a Tuple:

```
>>> pet = {'color': 'red', 'age': 42}
>>> for item in pet.items():
...     print(item)
...
# ('color', 'red')
# ('age', 42)
```

Using the `keys()`, `values()`, and `items()` methods, a for loop can iterate over the keys, values, or key-value pairs in a dictionary, respectively.

```
>>> pet = {'color': 'red', 'age': 42}
>>> for key, value in pet.items():
...     print(f'Key: {key} Value: {value}')
```

```
...
# Key: color Value: red
# Key: age Value: 42
```

get()

The `get()` method returns the value of an item with by using the key. If the key doesn't exist, it returns `None`:

```
>>> wife = {'name': 'Rose', 'age': 33}

>>> f'My wife name is {wife.get("name")}'
# 'My wife name is Rose'

>>> f'She is {wife.get("age")} years old.'
# 'She is 33 years old.'

>>> f'She is deeply in love with {wife.get("husband")}'
# 'She is deeply in love with None'
```

You can also change the default `None` value for other of your choice:

```
>>> wife = {'name': 'Rose', 'age': 33}

>>> f'She is deeply in love with {wife.get("husband", "lover")}'
# 'She is deeply in love with lover'
```

Adding items with setdefault()

It's possible to add an item to a dictionary in this way:

```
>>> wife = {'name': 'Rose', 'age': 33}
>>> if 'has_hair' not in wife:
...     wife['has_hair'] = True
```

Using the `setdefault` method we can make the same code more short:

```
>>> wife = {'name': 'Rose', 'age': 33}
>>> wife.setdefault('has_hair', True)
>>> wife
# {'name': 'Rose', 'age': 33, 'has_hair': True}
```

Removing Items

pop()

The `pop()` method removes and return an item based on a given key.

```
>>> wife = {'name': 'Rose', 'age': 33, 'hair': 'brown'}
>>> wife.pop('age')
# 33
```

```
>>> wife
# {'name': 'Rose', 'hair': 'brown'}
```

popitem()

The `popitem()` method remove the last item in a dictionary and returns it.

```
>>> wife = {'name': 'Rose', 'age': 33, 'hair': 'brown'}
>>> wife.popitem()
# ('hair', 'brown')
>>> wife
# {'name': 'Rose', 'age': 33}
```

del()

The `del()` method removes an item based on a given key.

```
>>> wife = {'name': 'Rose', 'age': 33, 'hair': 'brown'}
>>> del wife['age']
>>> wife
# {'name': 'Rose', 'hair': 'brown'}
```

clear()

The`clear()` method removes all the items in a dictionary.

```
>>> wife = {'name': 'Rose', 'age': 33, 'hair': 'brown'}
>>> wife.clear()
>>> wife
# {}
```

Checking keys in a Dictionary

```
>>> person = {'name': 'Rose', 'age': 33}

>>> 'name' in person.keys()
# True

>>> 'height' in person.keys()
# False

>>> 'skin' in person # You can omit keys()
# False
```

Checking values in a Dictionary

```
>>>  person = {'name': 'Rose', 'age': 33}

>>> 'Rose' in person.values()
# True
```

```
>>> 33 in person.values()
# True
```

Pretty Printing

```
>>> import pprint

>>> wife = {'name': 'Rose', 'age': 33, 'has_hair': True, 'hair_color': 'brown',
'height': 1.6, 'eye_color': 'brown'}
>>> pprint.pprint(wife)
# {'age': 33,
#  'eye_color': 'brown',
#  'hair_color': 'brown',
#  'has_hair': True,
#  'height': 1.6,
#  'name': 'Rose'}
```

Merge two dictionaries

For Python 3.5+:

```
>>> dict_a = {'a': 1, 'b': 2}
>>> dict_b = {'b': 3, 'c': 4}
>>> dict_c = {**dict_a, **dict_b}
>>> dict_c
# {'a': 1, 'b': 3, 'c': 4}
```