# Dictionary Basics & Properties

August 11, 2021

A dictionary is a kind of upgraded set, wherein you can map the elements to keys that you can define yourself. Remember that in a set, you cannot index or refer to specific elements of a set. Dictionary solves that problem using keys.

An item is a combination of a value with a key. Every item in a dictionary is present as if in a set. Each item is written in the format key:value...

[35]:
```
myDictionary = {1:"Pranav", 2:"Arnav", 3:"Hrithik"}
print("The full dictionary:", myDictionary)
print(myDictionary.keys())
print(myDictionary.values())
print(myDictionary.items())
```

```
The full dictionary: {1: 'Pranav', 2: 'Arnav', 3: 'Hrithik'}
dict_keys([1, 2, 3])
dict_values(['Pranav', 'Arnav', 'Hrithik'])
dict_items([(1, 'Pranav'), (2, 'Arnav'), (3, 'Hrithik')])
```

## 1 Referring to specific elements using keys

Note that keys can be any data type. More will be discussed when we talk about the heterogeneity of dictionaries...

[2]:
```
myDictionary = {1:"Pranav", 2:"Arnav", 3:"Hrithik"}
myDictionary[3]
```

[2]:
```
'Hrithik'
```

[3]:
```
# Pairing games with star ratings
myDictionary = {"Gladihoppers":8, "Minecraft":9, "Call of duty":6}
myDictionary["Gladihoppers"]
```

[3]:
```
8
```

## 2 Looping through a dictionary

[4]:
```
# Keys only
print("\nKeys only (without specifying)...")
for i in myDictionary:
    print(i)
# Keys only
print("\nKeys only (after specifying)...")
for i in myDictionary.keys():
    print(i)
# Values only
print("\nValues only...")
for i in myDictionary.values():
    print(i)
# Whole items i.e. keys and values
print("\nWhole items...")
for i in myDictionary.items():
    print(i)
```

```
Keys only (without specifying)…
Gladihoppers
Minecraft
Call of duty

Keys only (after specifying)…
Gladihoppers
Minecraft
Call of duty

Values only…
8
9
6

Whole items…
('Gladihoppers', 8)
('Minecraft', 9)
('Call of duty', 6)
```

As we can see, when we simply iterate i through 'a', i takes the values of only the keys in the dictionary. We can also specify this using a.keys() to return the set of keys for i to iterate through. To make sure i takes the values of only values in the dictionary, we use a.values() to return the set of values for i to iterate through. To make sure i takes the values of the whole items in the dictionary, we use a.items() to return the set of items for i to iterate through.

# 3 Properties

## 3.1 Potentially heterogenous in every way

Both keys and values can be heterogenous, and data type combinations for each item can also be heterogenous.

[40]:
```
myDictionary = {"a":1, 2:"b", 4.5:3}
print("The full dictionary:", myDictionary)
print(myDictionary.keys())
print(myDictionary.values())
print(myDictionary.items())
```

```
The full dictionary: {'a': 1, 2: 'b', 4.5: 3}
dict_keys(['a', 2, 4.5])
dict_values([1, 'b', 3])
dict_items([('a', 1), (2, 'b'), (4.5, 3)])
```

## 3.2 No duplicate keys

Similar to non-duplication in sets, but only applicable for keys. In the case that two keys are duplicated, only the last mention of the key in the dictionary will be counted. As with sets, it does not matter if the duplicate key values are present as different variables.

[37]:
```
myDictionary = {1:"Pranav", 2:"Arnav", 3:"Hrithik", 3:"Gabriel"}
myDictionary
```

[37]:
```
{1: 'Pranav', 2: 'Arnav', 3: 'Gabriel'}
```

[38]:
```
x = 3
y = 3
myDictionary = {1:"Pranav", 2:"Arnav", x:"Hrithik", y:"Gabriel"}
myDictionary
```

[38]:
```
{1: 'Pranav', 2: 'Arnav', 3: 'Gabriel'}
```

## 3.3 Mutable

### 3.3.1 Changing values

Like a set, a dictionary is also mutable. Unlike a set, we can access specific elements of the dictionary using the respective key. This increases our options, when we want to change, add or remove values from a dictionary.

[1]:
```
myDictionary = {1:"Pranav", 2:"Arnav", 3:"Hrithik", 4:"Gabriel"}
myDictionary[2] = "Gopinath"
myDictionary
```

[1]:
```
{1: 'Pranav', 2: 'Gopinath', 3: 'Hrithik', 4: 'Gabriel'}
```

### 3.3.2 Adding items

**Invalid operations**

[9]:
```
myDictionary = {1:"Pranav", 2:"Arnav", 3:"Hrithik"}
myDictionary + {4:"Gabriel"}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-30cdc21a15d0> in <module>
      1 myDictionary = {1:"Pranav", 2:"Arnav", 3:"Hrithik"}
----> 2 myDictionary + {4:"Gabriel"}

TypeError: unsupported operand type(s) for +: 'dict' and 'dict'
```

**Adding items** To add single or multiple values, you use the same function, update. This function adds the dictionary passed as the argument to the dictionary that is calling the function.

**Implicitly creating a new key**

[1]:
```
# Single item
myDictionary = {1:"Pranav", 2:"Arnav", 3:"Hrithik"}
myDictionary[4] = "Ronaldo"
myDictionary
# Here you see that a new key is implicitly created.
```

[1]:
```
{1: 'Pranav', 2: 'Arnav', 3: 'Hrithik', 4: 'Ronaldo'}
```

Hence, when an unused key is used for the assigment of a new value, that key is automatically added to the dictionary with the value assigned to it.

**Using the update function**

[16]:
```
# Single item
myDictionary = {1:"Pranav", 2:"Arnav", 3:"Hrithik"}
myDictionary.update({4:"Mani"})
myDictionary
```

[16]:
```
{1: 'Pranav', 2: 'Arnav', 3: 'Hrithik', 4: 'Mani'}
```

[13]:
```
# Multiple items
myDictionary = {1:"Pranav", 2:"Arnav", 3:"Hrithik"}
myDictionary.update({4:"Manu", 5:"Yani", 6:"Rupa"})
myDictionary
```

[13]:
```
{1: 'Pranav', 2: 'Arnav', 3: 'Hrithik', 4: 'Manu', 5: 'Yani', 6: 'Rupa'}
```

As usual, if you have duplicate key, the last item with that key alone is counted.

[12]:
```
# If duplicate keys are present after updation
myDictionary = {1:"Pranav", 2:"Arnav", 3:"Hrithik"}
myDictionary.update({3:"Manu", 4:"Yani", 5:"Rupa"})
myDictionary
```

[12]:
```
{1: 'Pranav', 2: 'Arnav', 3: 'Manu', 4: 'Yani', 5: 'Rupa'}
```

### 3.3.3 Removing items

**Removing single items** To remove single or multiple values, you use the same function, del. This function removes the dictionary values in the arguments, referred using their respective keys, from the identified dictionary.

[18]:
```
# Removing a single item
myDictionary = {1:"Pranav", 2:"Arnav", 3:"Hrithik", 4:"Gabriel"}
del(myDictionary[2])
myDictionary
```

[18]:
```
{1: 'Pranav', 3: 'Hrithik', 4: 'Gabriel'}
```

[21]:
```
# Removing multiple items
myDictionary = {1:"Pranav", 2:"Arnav", 3:"Hrithik", 4:"Gabriel"}
del(myDictionary[2], myDictionary[4])
myDictionary
```

[21]:
```
{1: 'Pranav', 3: 'Hrithik'}
```

**Emptying the entire dictionary**

[20]:
```
myDictionary = {1:"Pranav", 2:"Arnav", 3:"Hrithik", 4:"Gabriel"}
myDictionary.clear()
myDictionary
```

[20]:
```
{}
```