Dictionaries

In this chapter, we will work with Python dictionaries.

WE'LL COVER THE FOLLOWING

Dictionaries

Dictionaries are data structures that index values by a given key `(key-value pairs)`.

Dictionaries are written with curly brackets `{}`, and they have keys and values.

The general syntax for creating a dictionary is:

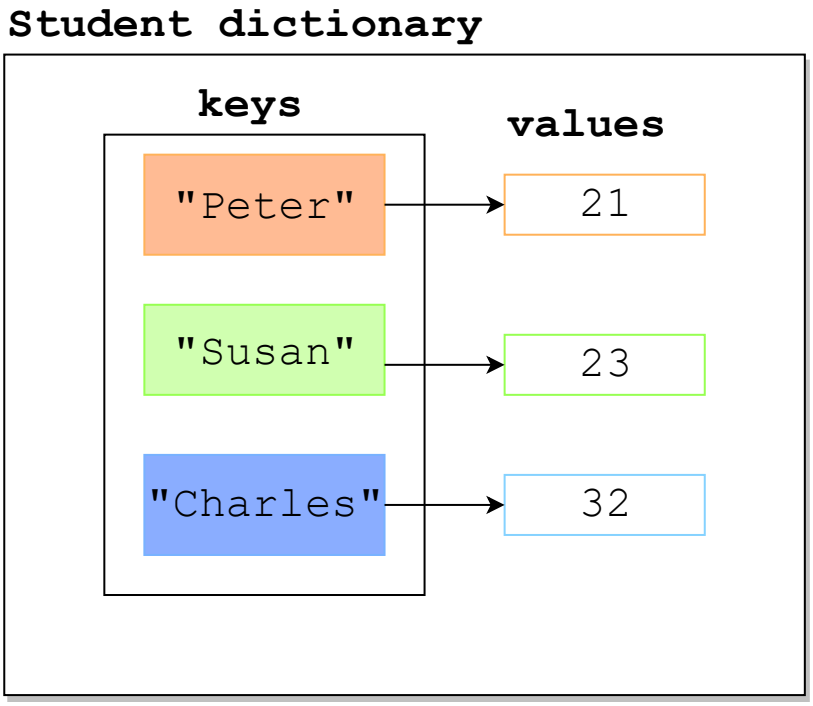
```
DictionaryName {

key1: value1,
key2: value2,
.
.
.
keyN: valueN,
```

```
}
```

Every key in a dictionary must be **unique** so that we know which value to return for a given key; however, dictionaries are NOT sorted. What makes dictionaries useful is that we assign a **key** to each value, instead of a numerical index like we do for a list.

Here is the visual example of a dictionary named `Student` with keys as student names and values as student ages.



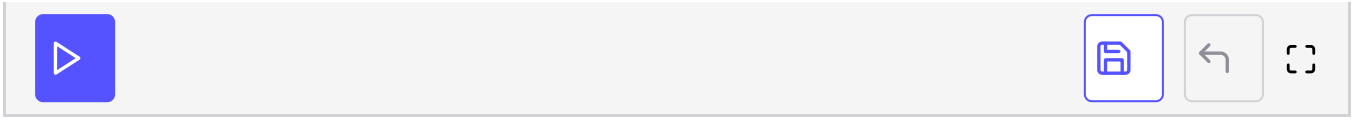
Create and Print a Dictionary

The following example shows a dictionary that indexes students' ages by name.

```
ages = {
    "Peter": 10,
    "Isabel": 11,
    "Anna": 9,
    "Thomas": 10,
    "Bob": 10,
    "Joseph": 11,
    "Maria": 12,
    "Gabriel": 10,
}
# print one item
print("Get age of peter")
print(ages["Peter"])

## print the whole dictionary
```

```
print("Get age of all persons")
for key, value in ages.items():
    print(key, value)
```



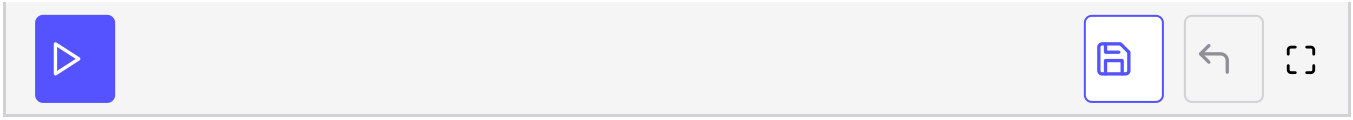
Call dictionary with no parameters using the `dict` keyword

```
new_dict = dict()
```

or simply write dictionary name followed by equal to `=` and `{}`

```
new_dict = {}
```

```
ages = dict()
ages['Peter'] = 12
ages['Susan'] = 13
for key, value in ages.items():
    print(key, value)
```



> **Note:** The order in which the keys are inserted is not maintained when the elements are printed on the console. It changes every time when the code is run.

You can create an **ordered dictionary** which preserves the order in which the keys are inserted. This is done by importing the `OrderedDictionary` from the `collections` library, and call the `OrderedDictionary()` built-in method.

1. `from collections import OrderedDict`
2. dictionary_name = `OrderedDict()`

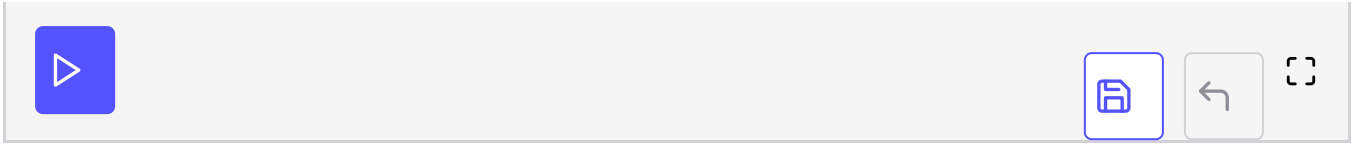
```
from collections import OrderedDict

ages = OrderedDict()

ages['Peter'] = 12
ages['Susan'] = 10
ages['Maria'] = 5

for key, value in ages.items():
```

```
    print(key, value)
```

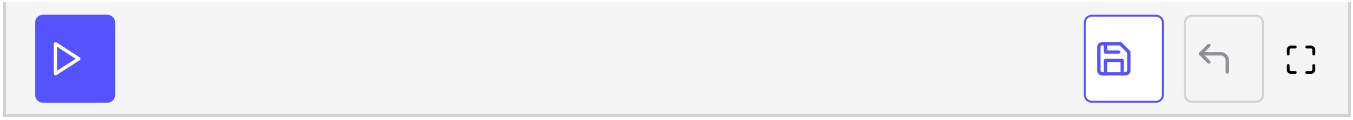


Accessing Items of a Dictionary

However, dictionary keys can be immutable object and don't necessarily need to be strings.

```
d = {
    0: [0, 0, 0],
    1: [1, 1, 1],
    2: [2, 2, 2],
}

print d[2]
```



Loop Through the Dictionary

Using `for` loop, we can iterate through the loop.

Loop to Get All Keys

To get all keys from the dictionary, use the following syntax:

> 📝 Note that the dictionary isn't necessarily printed in the order it was saved.

```
ages = {
    "Peter": 10,
    "Isabel": 11,
    "Anna": 9,
    "Thomas": 10,
    "Bob": 10,
    "Joseph": 11,
    "Maria": 12,
    "Gabriel": 10,
}

for x in ages:
  print(x)
```

Loop to Get All Values

To get all values from the dictionary use the following syntax:

```
ages = {
    "Peter": 10,
    "Isabel": 11,
    "Anna": 9,
    "Thomas": 10,
    "Bob": 10,
    "Joseph": 11,
    "Maria": 12,
    "Gabriel": 10,
}

for x in ages:
  print(ages[x])
```

Another method to return values of a dictionary is to use `values()` function:

```
ages = {
    "Peter": 10,
    "Isabel": 11,
    "Anna": 9,
    "Thomas": 10,
    "Bob": 10,
    "Joseph": 11,
    "Maria": 12,
    "Gabriel": 10,
}

for x in ages.values():
  print(x)
```

Loop to Get Both Keys and Values

It is possible to iterate over the contents of a dictionary using `items()`, like this:

```
ages = {
    "Peter": 10,
    "Isabel": 11,
```

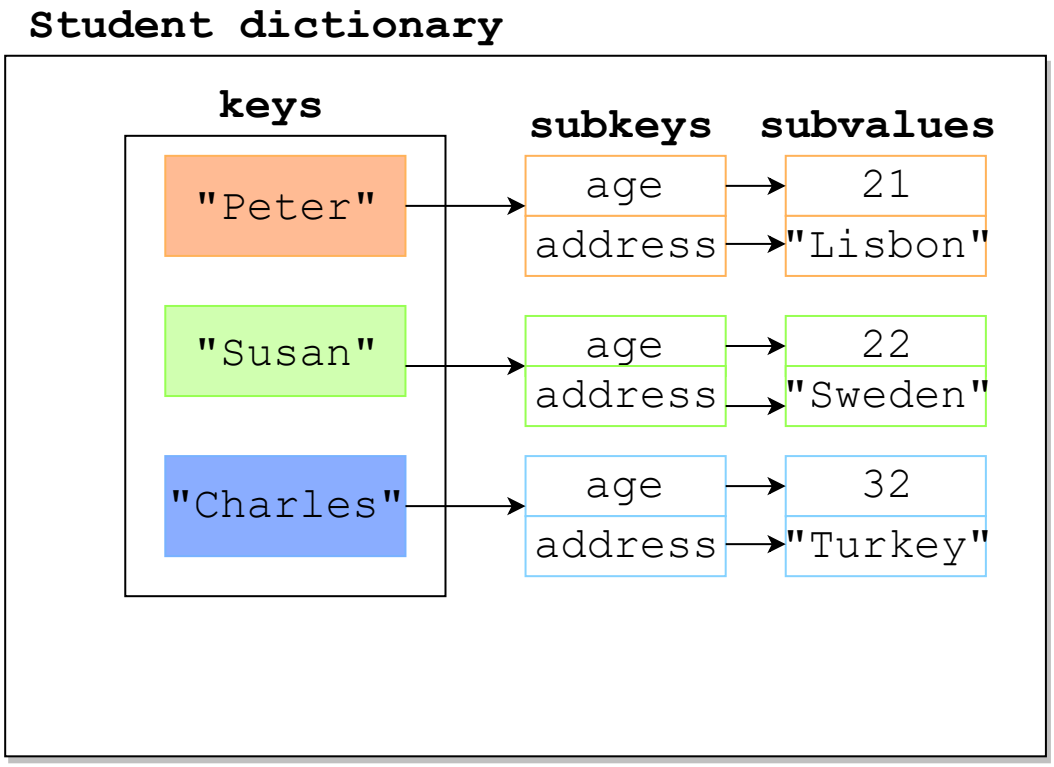
```
    "Anna": 9,
    "Thomas": 10,
    "Bob": 10,

    "Joseph": 11,
    "Maria": 12,
    "Gabriel": 10,
}

for name, age in ages.items():
  print name, age
```

Nested Dictionary

A dictionary can be made within a dictionary, and you can also use other dictionaries as values.



The following python code demonstrates the concept of nested dictionary.

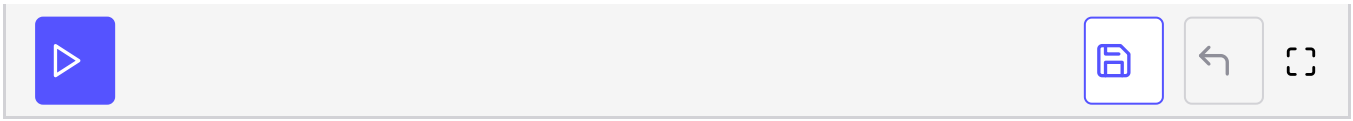
```
students = {
    "Peter": {"age": 10, "address": "Lisbon"},
    "Isabel": {"age": 11, "address": "Sesimbra"},
    "Anna": {"age": 9, "address": "Lisbon"},
}
print students
print students['Peter']
print students['Peter']['address']
```

Looping Through a Nested Dictionary

We can loop through the nested dictionary using a nested for loop.

```
students = {
    "Peter": {"age": 10, "address": "Lisbon"},
    "Isabel": {"age": 11, "address": "Sesimbra"},
    "Anna": {"age": 9, "address": "Lisbon"},
}

for p_id, p_info in students.items():
    print("\nPerson Name:", p_id)
    for key in p_info:
        print(key + ':', p_info[key])
```



This is quite useful to structure hierarchical information.

Now that the idea of “Dictionaries” in python is clear, let’s check your knowledge in the upcoming exercises before moving on to the next chapter —‘Classes’.