Type `dict`

- Chapter 11.3 Storing Data Using Dictionaries

Dictionary

Another way to store collections of data is using Python's dictionary type: `dict`.

The general form of a dictionary is:

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

Keys must be unique. Values may be duplicated. For example:

```
asn_to_grade = {'A1': 80, 'A2': 90, 'A3': 90}
```

In the example above, the keys are unique: `'A1'`, `'A2'` and `'A3'`. The values are not unique: `80`, `90` and `90`.

How To Modify Dictionaries

Dictionaries are mutable: they can be modified. There are a series of operations and methods you can apply to dictionaries which are outlined below.

Operation	Description	Example

`object in dict`	Checks whether `object` is a key in `dict`.	`>>> asn_to_grade = {'A1': 80, 'A2': 90, 'A3': 90}`
		`>>> 'A1' in asn_to_grade`
		`True`
		`>>> 80 in asn_to_grade`
		`False`

`len(dict)`	Returns the number of keys in `dict`.	`>>> asn_to_grade = {'A1': 80, 'A2': 90, 'A3': 90}`
		`>>> len(asn_to_grade)`
		`3`

`del dict[key]`	Removes a key and its associated value from `dict`.	`>>> asn_to_grade = {'A1': 80, 'A2': 90, 'A3': 90}`
		`>>> del asn_to_grade['A1']`
		`>>> asn_to_grade`
		`{'A3': 90, 'A2': 90}`

`dict[key] = value`	If `key` does not exist in `dict`, adds `key` and its associated `value` to `dict`.
	If `key` exists in `dict`, updates `dict` by setting the value associated with `key` to `value`.
		`>>> asn_to_grade = {'A1' : 80, 'A2': 90, 'A3' : 90}`
		`>>> asn_to_grade['A4'] = 70`
		`>>> asn_to_grade`
		`{'A1': 80, 'A3': 90, 'A2': 90, 'A4': 70}`

Accessing Information From Dictionaries

Dictionaries are unordered. That is, the order the key-value pairs are added to the dictionary has no effect on the order in which they are accessed. For example:

```
>>> asn_to_grade = {'A1': 80, 'A2': 70, 'A3': 90}
>>> for assignment in asn_to_grade:
    print(assignment)


A1
A3
A2
```

The for-loop above printed out the keys of the dictionary. It is also possible to print out the values:

```
>>> asn_to_grade = {'A1': 80, 'A2': 70, 'A3': 90}
>>> for assignment in asn_to_grade:
    print(asn_to_grade[assignment])


80
90
70
```

Finally, both the keys are values can be printed:

```
>>> asn_to_grade = {'A1': 80, 'A2': 70, 'A3': 90}
>>> for assignment in asn_to_grade:
    print(assignment, asn_to_grade[assignment])


A1 80
A3 90
A2 70
```

Empty Dictionaries

A dictionary can be empty. For example:

```
d = {}
```

Heterogeneous Dictionaries

A dictionary can have keys of different types. For example, one key can be of type `int` and another of type `str`:

```
d = {'apple': 1, 3: 4}
```

Immutable Keys

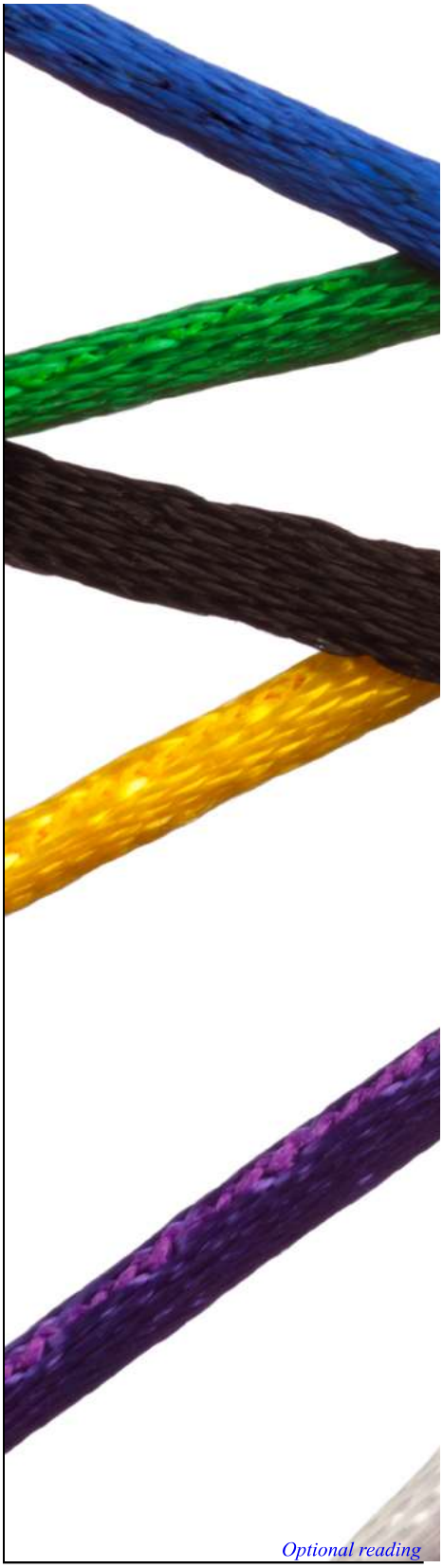
The keys of a dictionary must be immutable. Therefore, lists, dictionary and other mutable types cannot be used as keys. The following results in an error:

```
d[[1, 2]] = 'banana'
```

Since lists are mutable, they cannot be keys. Instead, to use a sequence as a key, type `tuple` can be used:

```
d[(1, 2)] = 'banana'
```

Jennifer Campbell • Paul Gries
University of Toronto


Optional reading