

## 1.3 Dictionaries

---

### 1.3 Dictionaries

Dictionaries are used to store data values in key:value pairs. A dictionary is a collection which is ordered\*, changeable, indexed, and does not allow duplicates. Dictionaries are written with curly brackets and have keys and values.

\*As of Python version 3.7, dictionaries are ordered. In Python 3.6 and earlier, dictionaries are unordered.

Note: Both the primary data types and structured data types such as lists, tuples, and sets; store each data element as a single value. In contrast, dictionaries store each data element as a pair of values called a key and a value.

The key in a dictionary must be unique and it must be immutable. We can create dictionaries using mixed data types for the key.



#### When to use a dictionary?

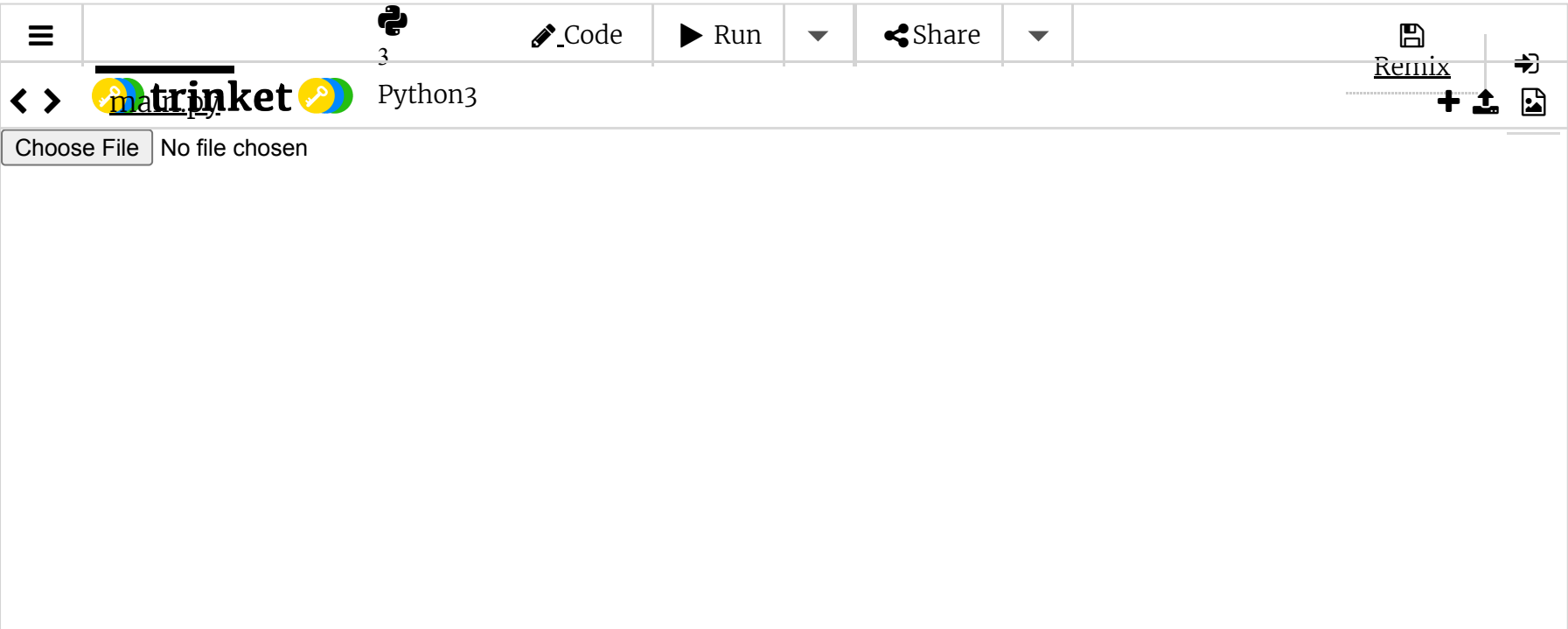
- Storing elements as key-value pairs
  - Unlike any other type of collection in Python
- Quick access to data is required
  - Find a value instantly without the need for searching through the whole collection

#### Dictionary vs List

List	Dictionary
Access using index	Access using keys
Collection of elements	Collection of key-value pairs.
Allows duplicate items	No duplicate items
The indices of the list are integers starting from 0	The keys of the dictionary can be of any data type

#### Defining a Dictionary

- Using Curly Brackets to initialize a dictionary.  
To create an empty dictionary, we simply use the curly braces { } without any data elements inside.



#### 2. Use of dict() built-in function

## 2. USE OF dict() BUILT-IN FUNCTION.

A common and useful way of creating dictionaries is to use a sequence (say, a list) of pair-type tuples where the first value in the pair is the key and pass it to the Python built-in function `dict()`.

[illegible]

## Accessing elements of a dictionary

Dictionary is quite a useful data structure in programming that is usually used to hash a particular key with value so that they can be retrieved efficiently.

You can access the items of a dictionary by referring to its key name, inside square brackets.

There is also a method called `get()` that will give you the same result.

[illegible]

The **keys()** method will return a list of all the keys in the dictionary.

The **values()** method will return a list of all the values in the dictionary.

The **items()** method will return each item in a dictionary, as tuples in a list.

Python3

main.py

Choose File No file chosen

Remix

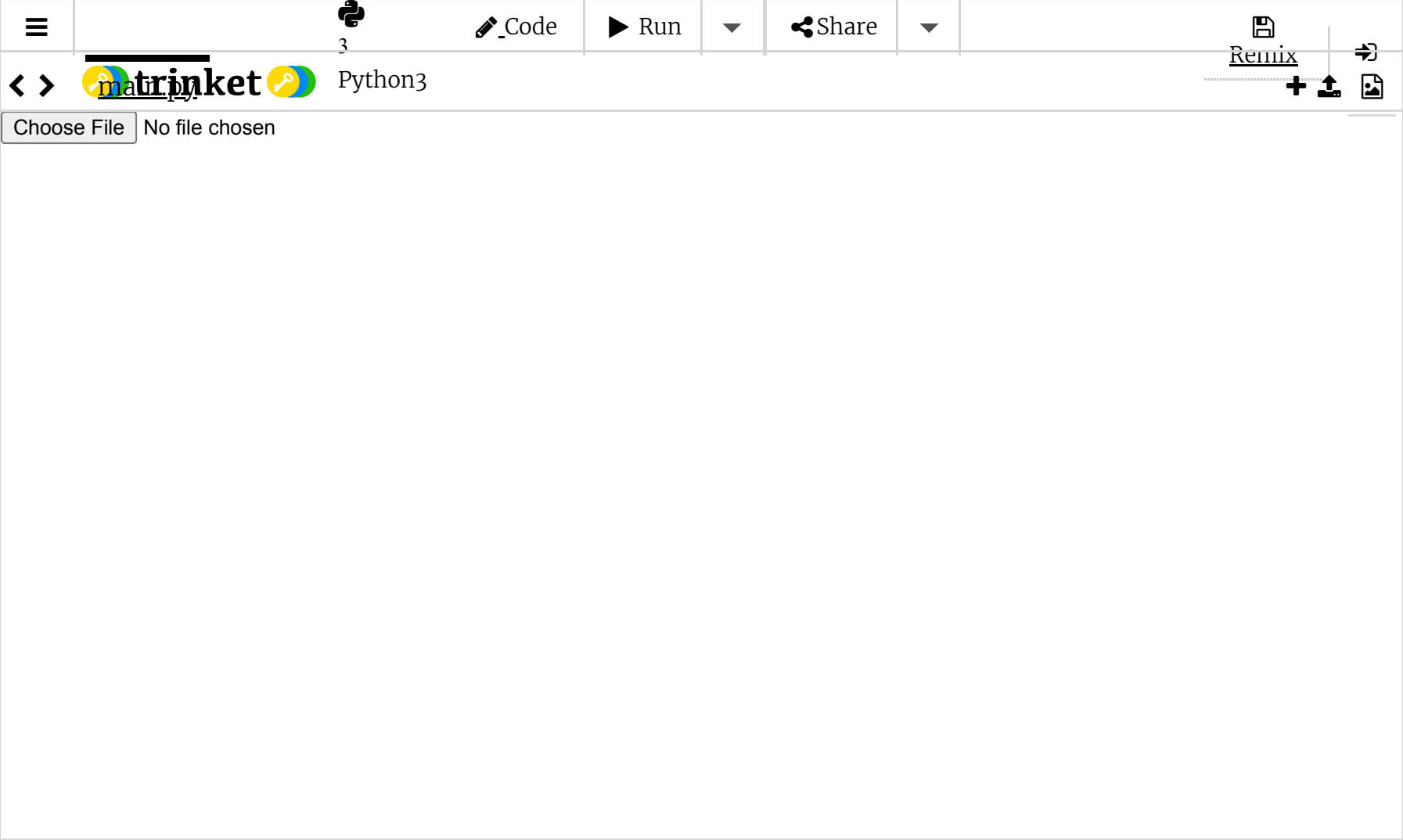
+ ↕ 📎

## Python Dictionary Operations

Refer to the examples below on adding and updating elements.

[illegible]

**Note:** Both the Subscript Notation and the built-in update() function can be used to add an element or update an existing element of a dictionary.



**Removing an element**

- 1. Using the del keyword.

The keys in a Python dictionary can be deleted using the del keyword. There are 2 ways in which the del keyword can be used.

We can delete either specific entries in the dictionary with the use of selected keys. We can delete the entire dictionary.(Once we delete a dictionary, it is no longer available in a program.)

- 2. Using the built-in function pop().

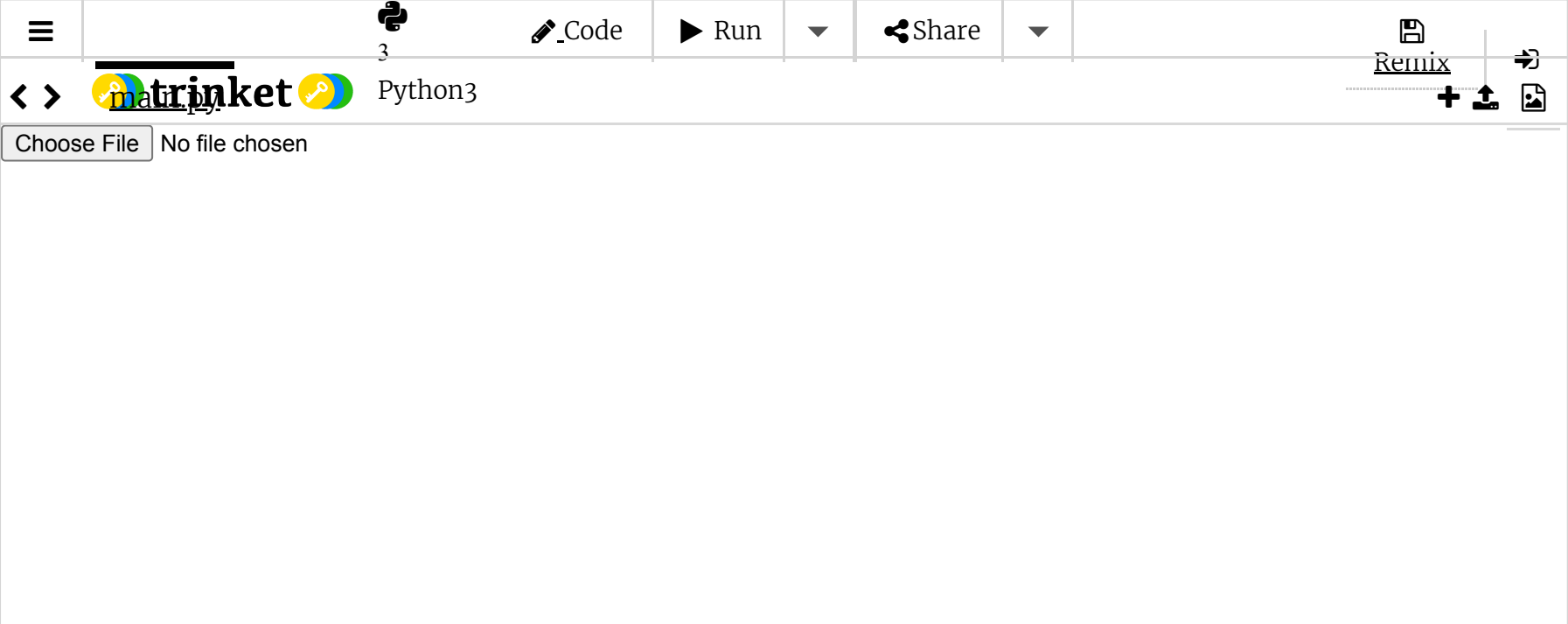
An entry in a dictionary can be removed using the built-in function pop() and specifying the key as an argument. The pop() method will return the value from the dictionary while deleting the full entry.

- 3. Using the built-in function popitem().

The popitem() method will return an arbitrary element as a tuple (that is, a (key, value) pair) from the dictionary while deleting the full entry.

**Removing all elements**

Using the built-in function clear() method will delete all the items from a dictionary at once. Unlike the deletion of a dictionary that we can do with del keyword, a cleared dictionary is still available in a program.



[illegible]

## Using `enumerate()`

The image shows the top portion of the Trinket IDE web interface. At the top, the text "Using enumerate()" is visible. Below it is a navigation bar with a hamburger menu icon on the left, followed by the Trinket logo (three overlapping circles in yellow, green, and blue) and the text "Python3". To the right of the logo are icons for a code editor (pencil), a run button (play triangle), and a share button (network icon). Further right are icons for saving (floppy disk), a plus sign, a download icon, and a document icon. At the bottom of the interface is a file explorer bar with a "Choose File" button and the text "No file chosen".

### Using class method **items()**

### Using list comprehension

The screenshot shows the Trinket.io web interface. At the top, there's a navigation bar with icons for menu, Python logo, code editor, run button, share button, and a 'Remix' button. Below the navigation bar, the main area displays a Python3 code editor with a list comprehension example: 

```
names = ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace', 'Heidi', 'Ivy', 'Jack', 'Karen', 'Leo', 'Mia', 'Noah', 'Olivia', 'Pamela', 'Quinn', 'Ryan', 'Sophia', 'Tyler', 'Uma', 'Victor', 'Wendy', 'Xavier', 'Yara', 'Zoe']
```

 The code is highlighted in a light blue color. The interface is clean and modern, with a white background and a light blue header.


## When not to use a Dictionary?


- ### When not to use a Dictionary?
- For storing data that shouldn't be modified


- Remember that dictionaries are mutable
- When the resource (memory) usage is a concern
  - Dictionaries take up more memory compared to other collection types!

Jump to...

## GET IN TOUCH

 University of Moratuwa  
Centre for Open & Distance Learning  
CODL

 011 308 2787/8

 011 265 0301 ext. 3850,3851

 open@uom.lk

 [University Website](#)

 [CODL Website](#)

[Data retention summary](#)