At the end of this lecture, students should be able to:

- understand what a dictionary is
- create a dictionary object
- add items to a dictionary
- retrieve items from a dictionary
- traverse the pairs in a dictionary

Python dictionaries

A dictionary is a mapping from a key to its associated data value.

- Each key maps to a value.
- The key has to be unique and an immutable object.

A phone book is an example of a mapping: the key is the person’s name (plus address) and the associated value is their phone number.

You can think of a dictionary as a group of pairs, where the first element of the pair, the key, is used to retrieve the second element, the corresponding value.

The key and its associated value is called a key-value pair or they can be called an item.
Creating an object of type dict

Curly braces are used for dictionaries and empty curly braces {} define an empty dictionary, i.e., containing no key-value pairs:

```python
def main():
    english_italian = {}
    print(english_italian)
    print(type(english_italian))
main()
```

Another way to create an empty dictionary object is (does exactly the same thing as the code above) is:

```python
def main():
    english_italian = dict()
    print(english_italian)
    print(type(english_italian))
main()
```

dict is a Python type

Note that the name, `dict`, is a Python type (`<class 'dict'>`) and should not be used as a variable name.

```python
def main():
    english_italian = dict()
main()
```

Creating a dictionary which contains pairs

A dictionary object can be initialised with key-value pairs:

Each associated pair is separated by `:` and the pairs are separated by commas.

```python
def main():
    english_italian = {"yes": "si", "bye": "ciao", "no": "no",
                       "maybe": "forse", "thank you": "grazie"}
    print(english_italian)
    contacts = {"Jill": 3456, "James": 3456, "Yi": 7654,
                       "Syed": 6754}
    print(contacts)
main()
```

Note: the keys MUST be unique but the associated values need not.

Visualising the dictionary

```python
contacts = {"Jill": 3456, "James": 3456, "Yi": 7654, "Syed": 6754}
print(contacts)
main()
```

Note: when the key-value pairs are printed, the order is not predictable.
The keys of the dictionary must be immutable
The keys of a dictionary must be of a type which is immutable such as: string, int, tuple.
The keys of a dictionary must be unique.
The values can be of any type and they do not need to be unique.

Adding an item to the dictionary
Key-value pairs can be added to the dictionary using assignment statements. For example,

```python
def main():
    contacts = {"Jill": 3456, "James": 3456, "Yi": 7654, "Syed": 6754}
    contacts["Mark"] = 7654
    contacts["Jerry"] = 7004
    print(contacts)
main()
```

Dictionaries are not ordered structures
Dictionary elements cannot be accessed using the index value. A dictionary is a collection of key:value pairs.

There is no predictable order to the key:value pairs in a dictionary (when printed, the order may change as new items are added and removed).

Changing the associated value in a dictionary
The associated value of a pair can be changed by assigning a different value to the dictionary key. This replaces the old value.

```python
def main():
    contacts = {"Jill": 3456, "James": 3456, "Yi": 7654, "Syed": 6754}
    contacts["Jill"] = 7654
    contacts["Yi"] = 7004
    print(contacts)
main()
```
Access the value associated with a key

The value associated with a certain key can be accessed using square brackets (enclosing the key):

def main():
    contacts = {"Jill": 3456, "James": 3456, "Yi": 7654, "Syed": 6754}
    name1 = "Jill"
    name2 = "James"
    print(name1, "is at extension:", contacts[name1])
    if contacts[name1] == contacts[name2]:
        print(name2, "has the same extension")
main()

The number of key-value pairs in a dictionary

The `len()` function can be used with a dictionary object to find out how many key-value pairs are currently in the dictionary:

def main():
    contacts = {"Jill": 3456, "James": 3456, "Yi": 7654, "Syed": 6754}
    print(len(contacts), "in dictionary")
    contacts["Yi"] = 7654
    contacts["Jerry"] = 7004
    print(len(contacts), "in dictionary")
main()

Check if a key is in the dictionary

The 'in' operator can be used to check if a key is in the dictionary:

def main():
    contacts = {"Jill": 3456, "James": 3456, "Yi": 7654, "Syed": 6754}
    name = "Jack"
    if name in contacts:
        print(name, "is at extension:", contacts[name])
    else:
        contacts[name] = 0
    if name in contacts:
        print(name, "is at extension:", contacts[name])
    print(contacts)
main()

Traversing the pairs in the dictionaries

Use a `for ... in` loop to traverse (visit) each key in the dictionary:

def main():
    contacts = {"Jill": 3456, "James": 3456, "Yi": 7654, "Syed": 6754}
    for name in contacts:
        print(name, "-", contacts[name])
main()
The in operator with dictionaries
An error is raised when accessing a key which is not in the dictionary. Always test before accessing a key-value pair.

```python
def main():
    contacts = {"Jill": 3456, "James": 3456, "Yi": 7654, "Syed": 6754}
    if "Jill" in contacts:
        # Test first
        print("Jill", "-", contacts["Jill"])
        print(contacts["Izzy"])  # Incorrect key

main()
```

Jill - 3456
Traceback (most recent call last):
  File "LectureCode.py", line 5, in <module>
    print(contacts["Izzy"])  
KeyError: 'Izzy'

Exercise

A small trouble is like a pebble Hold it too close to your eye and it fills the whole world and puts everything out of focus Hold it at the proper distance and it can be examined and properly classified Throw it at your feet and it can be seen in its true setting just another tiny bump on the pathway of life

```
def get_dictionary_from_file_words(filename): #complete the code

def display_results(vowel_words):
    # complete the code

main()
```

Note: For this program, the punctuation has been left in the text. This means that the word 'eye' is a different word to the word 'eye,'.

Exercise

"Story.txt" is a text file. The following program reads the text from the file, converts it to lower case, and creates a dictionary of all the unique words (sorted) which start with a vowel ("a", "e", "i", "o", "u"). Note: the key is the vowel and each word is added to the corresponding associated list (the associated list grows as the text is processed). Change the story text to lower case.

```
de

def main():
    vowel_words_dict = get_dictionary_from_file_words("Story.txt")
    display_results(vowel_words_dict)

def get_dictionary_from_file_words(filename): #complete the code
    def display_results(vowel_words): #complete the code

main()
```

e = ['everything', 'examined', 'eye,']
a = ['a', 'and', 'another', 'at']
i = ['in', 'is', 'it', 'its']
u = []
o = ['of', 'on', 'out']

A small trouble is like a pebble Hold it too close to your eye and it fills the whole world and puts everything out of focus Hold it at the proper distance, and it can be examined and properly classified. Throw it at your feet and it can be seen in its true setting just another tiny bump on the pathway of life.
Exercise

def display_results(vowel_words_dict):

Summary

In Python:

- dictionaries are used to store key:value pairs (also called items)
- an empty dictionary object can be created in two ways
- items can be added to a dictionary
- items can be retrieved from the dictionary
- the keys of a dictionary can be traversed using for ... in

Python features used in this lecture

english_italian = {"yes":"si", "bye":"ciao", "no":"no", "maybe":"forse",
                    "thank you":"grazie"}

english_italian["never"] = "mai"
print(english_italian["bye"])

for word in english_italian:
    print(english_italian[word])

print(len(english_italian))