Dictionaries and Sets

陳建良

Dictionaries

- A *dictionary* is an object that stores a collection of data.
- Each element in a dictionary has two parts: *a key* and *a value*.
- In fact, dictionary elements are commonly referred to as *key-value pairs*.
- You use a key to locate a specific value.

Creating a Dictionary

- You can create a dictionary by enclosing the elements inside a set of curly braces ({}).
- An element consists of a key, followed by a colon, followed by a value.

phonebook = {'Chris':'555−1111', 'Katie':'555−2222’, 'Joanne':'555−3333'}

Retrieving a Value from a Dictionary

```
>>> phonebook = {'Chris':'555-1111', 'Katie':'555-2222',
'Joanne':'555-3333'} Enter
>>> phonebook Enter
{'Chris': '555-1111', 'Joanne': '555-3333', 'Katie': '555-2222'}
>>>
```

- To retrieve a value from a dictionary, you simply write an expression in the following general format:

  dictionary_name[*key*]

```
>>> phonebook = {'Chris':'555-1111', 'Katie':'555-2222',
'Joanne':'555-3333'} Enter
>>> phonebook['Chris'] Enter
'555-1111'
>>> phonebook['Joanne'] Enter
'555-3333'
>>> phonebook['Katie'] Enter
'555-2222'
```

Aletheia University
資訊工程學系

Using the in and not in Operators to Test for a Value in a Dictionary

```
>>> phonebook = {'Chris':'555-1111', 'Katie':'555-2222',
'Joanne':'555-3333'} Enter
>>> if 'Chris' in phonebook: Enter
        print(phonebook['Chris']) Enter Enter

555-1111
>>>

>>> phonebook = {'Chris':'555-1111', 'Katie':'555-2222'} Enter
>>> if 'Joanne' not in phonebook: Enter
        print('Joanne is not found.') Enter Enter

Joanne is not found.
>>>
```

Adding Elements to an Existing Dictionary

- You can add new key-value pairs to a dictionary with an assignment statement in the following general format:

  dictionary_name[*key*] = *value*

```
>>> phonebook = {'Chris':'555-1111', 'Katie':'555-2222',
'Joanne':'555-3333'} Enter
>>> phonebook['Joe'] = '555-0123' Enter
>>> phonebook['Chris'] = '555-4444' Enter
>>> phonebook Enter
{'Chris': '555-4444', 'Joanne': '555-3333', 'Joe': '555-0123',
 'Katie': '555-2222'}
>>>
```

Deleting Elements

- You can delete an existing key-value pair from a dictionary with the del statement. Here is the general format:

  del *dictionary_name*[*key*]

```
>>> phonebook = {'Chris':'555-1111', 'Katie':'555-2222',
'Joanne':'555-3333'} Enter
>>> phonebook Enter
{'Chris': '555-1111', 'Joanne': '555-3333', 'Katie': '555-2222'}
>>> del phonebook['Chris'] Enter
>>> phonebook Enter
{'Joanne': '555-3333', 'Katie': '555-2222'}
```

```
>>> del phonebook['Chris'] Enter
Traceback (most recent call last):
    File "<pyshell#5>", line 1, in <module>
        del phonebook['Chris']
KeyError: 'Chris'
>>>
```

- To prevent a KeyError exception from being raised, you should use the in operator to determine whether a key exists before you try to delete it and its associated value.

```
>>> phonebook = {'Chris':'555-1111', 'Katie':'555-2222',
'Joanne':'555-3333'} Enter
>>> if 'Chris' in phonebook: Enter
        del phonebook['Chris'] Enter Enter

>>> phonebook Enter
{'Joanne': '555-3333', 'Katie': '555-2222'}
>>>
```

Getting the Number of Elements in a Dictionary

- You can use the built-in len function to get the number of elements in a dictionary.

```
>>> phonebook = {'Chris':'555-1111', 'Katie':'555-2222'} Enter
>>> num_items = len(phonebook) Enter
>>> print(num_items) Enter
2
>>>
```

Mixing Data Types in a Dictionary

- The keys in a dictionary must be immutable objects, but their associated values can be any type of object.
- For example, the values can be lists, as demonstrated in the following

```
>>> test_scores = { 'Kayla' : [88, 92, 100], Enter
                    'Luis' : [95, 74, 81], Enter
                    'Sophie' : [72, 88, 91], Enter
                    'Ethan' : [70, 75, 78] } Enter
>>> test_scores Enter
{'Kayla': [88, 92, 100], 'Sophie': [72, 88, 91], 'Ethan': [70, 75, 78],
'Luis': [95, 74, 81]}
>>> test_scores['Sophie'] Enter
[72, 88, 91]
>>> kayla_scores = test_scores['Kayla'] Enter
>>> print(kayla_scores) Enter
[88, 92, 100]
>>>
```

- The values that are stored in a single dictionary can be of different types. For example, one element’s value might be a string, another element’s value might be a list, and yet another element’s value might be an integer.

```
>>> mixed_up = {'abc':1, 999:'yada yada', (3, 6, 9):[3, 6, 9]} Enter
>>> mixed_up Enter
{(3, 6, 9): [3, 6, 9], 'abc': 1, 999: 'yada yada'}
>>>
```

- The following interactive session gives a more practical example.

```
>>> employee = {'name' : 'Kevin Smith', 'id' : 12345, 'payrate' :
25.75 } Enter
>>> employee Enter
{'payrate': 25.75, 'name': 'Kevin Smith', 'id': 12345}
>>>
```

真理大學 Aletheia University 資訊工程學系

Creating an Empty Dictionary

```
>>> phonebook = {} Enter
>>> phonebook['Chris'] = '555-1111' Enter
>>> phonebook['Katie'] = '555-2222' Enter
>>> phonebook['Joanne'] = '555-3333' Enter
>>> phonebook Enter
{'Chris': '555-1111', 'Joanne': '555-3333', 'Katie': '555-2222'}
>>>
```

- You can also use the built-in dict() method to create an empty dictionary, as shown in the following statement:

  phonebook = dict()

真理大學 Aletheia University 資訊工程學系

Using the for Loop to Iterate over a Dictionary

- You can use the for loop in the following general format to iterate over all the keys in a dictionary:

  for *var* in *dictionary*:

  statement

  statement

  etc.

```
>>> phonebook = {'Chris':'555-1111', Enter
                 'Katie':'555-2222', Enter
                 'Joanne':'555-3333'} Enter
>>> for key in phonebook: Enter
        print(key) Enter Enter


Chris
Joanne
Katie
>>> for key in phonebook: Enter
        print(key, phonebook[key]) Enter Enter


Chris 555-1111
Joanne 555-3333
Katie 555-2222
>>>
```

Aletheia University
真理大學
資訊工程學系

Some Dictionary Methods

Method	Description
`clear`	Clears the contents of a dictionary.
`get`	Gets the value associated with a specified key. If the key is not found, the method does not raise an exception. Instead, it returns a default value.
`items`	Returns all the keys in a dictionary and their associated values as a sequence of tuples.
`keys`	Returns all the keys in a dictionary as a sequence of tuples.
`pop`	Returns the value associated with a specified key and removes that key-value pair from the dictionary. If the key is not found, the method returns a default value.
`popitem`	Returns a randomly selected key-value pair as a tuple from the dictionary and removes that key-value pair from the dictionary.
`values`	Returns all the values in the dictionary as a sequence of tuples.

The clear Method

- The clear method deletes all the elements in a dictionary, leaving the dictionary empty. The method's general format is

  dictionary.clear()

```
>>> phonebook = {'Chris':'555-1111', 'Katie':'555-2222'} Enter
>>> phonebook Enter
{'Chris': '555-1111', 'Katie': '555-2222'}
>>> phonebook.clear() Enter
>>> phonebook Enter
{}
>>>
```

The get Method

- You can use the get method as an alternative to the [] operator for getting a value from a dictionary.
- Here is the method's general format:

  dictionary.get(*key*, *default*)

```
>>> phonebook = {'Chris':'555-1111', 'Katie':'555-2222'} Enter
>>> value = phonebook.get('Katie', 'Entry not found') Enter
>>> print(value) Enter
555-2222
>>> value = phonebook.get('Andy', 'Entry not found') Enter
>>> print(value) Enter
Entry not found
>>>
```

The items Method

- The items method returns all of a dictionary’s keys and their associated values.
- They are returned as a special type of sequence known as a *dictionary view*.

```
phonebook = {'Chris':'555−1111', 'Katie':'555−2222', 'Joanne':'555−3333’}

phonebook.items()

[('Chris', '555−1111'), ('Joanne', '555−3333'), ('Katie', '555−2222')]
```

Aletheia University
資訊工程學系

- You can use the for loop to iterate over the tuples in the sequence.

```
>>> phonebook = {'Chris':'555-1111', Enter
                 'Katie':'555-2222', Enter
                 'Joanne':'555-3333'} Enter
>>> for key, value in phonebook.items(): Enter
        print(key, value) Enter Enter


Chris 555-1111
Joanne 555-3333
Katie 555-2222
>>>
```

Aletheia University 真理大學 資訊工程學系

The keys Method

- The keys method returns all of a dictionary's keys as a dictionary view, which is a type of sequence.

```
>>> phonebook = {'Chris':'555-1111', Enter
                 'Katie':'555-2222', Enter
                 'Joanne':'555-3333'} Enter
>>> for key in phonebook.keys(): Enter
        print(key) Enter Enter


Chris
Joanne
Katie
>>>
```

真理大學 Aletheia University 資訊工程學系

The pop Method

- The pop method returns the value associated with a specified key and removes that keyvalue pair from the dictionary.
- If the key is not found, the method returns a default value.

  dictionary.pop(*key*, *default*)

- In the general format, ***dictionary*** is the name of a dictionary, ***key*** is a key to search for in the dictionary, and ***default*** is a default value to return if the *key* is not found.

```
>>> phonebook = {'Chris':'555-1111', Enter
                 'Katie':'555-2222', Enter
                 'Joanne':'555-3333'} Enter
>>> phone_num = phonebook.pop('Chris', 'Entry not found') Enter
>>> phone_num Enter
'555-1111'
>>> phonebook Enter
{'Joanne': '555-3333', 'Katie': '555-2222'}
>>> phone_num = phonebook.pop('Andy', 'Element not found') Enter
>>> phone_num Enter
'Element not found'
>>> phonebook Enter
{'Joanne': '555-3333', 'Katie': '555-2222'}
>>>
```

Aletheia University 資訊工程學系

The popitem Method

- The popitem method returns a randomly selected key-value pair, and it removes that keyvalue pair from the dictionary.
- The key-value pair is returned as a tuple.

  dictionary.popitem()

  k, *v* = *dictionary*.popitem()

- This type of assignment is known as a ***multiple assignment*** because multiple variables are being assigned at once.
- In the general format, *k* and *v* are variables.

```
>>> phonebook = {'Chris':'555-1111', Enter
                 'Katie':'555-2222', Enter
                 'Joanne':'555-3333'} Enter
>>> phonebook Enter
{'Chris': '555-1111', 'Joanne': '555-3333', 'Katie': '555-2222'}
>>> key, value = phonebook.popitem() Enter
>>> print(key, value) Enter
Chris 555-1111
>>> phonebook Enter
{'Joanne': '555-3333', 'Katie': '555-2222'}
>>>
```

- The popitem method raises a KeyError exception if it is called on an empty dictionary.

Aletheia University 資訊工程學系

The values Method

- The values method returns all a dictionary’s values (without their keys) as a dictionary view, which is a type of sequence.
- Each element in the dictionary view is a value from the dictionary.

```
phonebook = {'Chris':'555−1111', 'Katie':'555−2222', 'Joanne':'555−3333’}

phonebook.values()

['555−1111', '555−2222', '555−3333']
```

- You can use a for loop to iterate over the sequence that is returned from the values method:

```
>>> phonebook = {'Chris':'555-1111', Enter
                 'Katie':'555-2222', Enter
                 'Joanne':'555-3333'} Enter
>>> for val in phonebook.values(): Enter
        print(val) Enter Enter


555-1111
555-3333
555-2222
>>>
```

Using a Dictionary to Simulate a Deck of Cards

- In some games involving poker cards, the cards are assigned numeric values. For example, in the game of Blackjack, the cards are given the following numeric values:
  - Numeric cards are assigned the value they have printed on them. For example, the value of the 2 of spades is 2, and the value of the 5 of diamonds is 5.
  - Jacks, queens, and kings are valued at 10.
  - Aces are valued at either 1 or 11, depending on the player's choice.

- In this section, we look at a program that uses a dictionary to simulate a standard deck of poker cards, where the cards are assigned numeric values similar to those used in Blackjack. (In the program, we assign the value 1 to all aces.)
- The key-value pairs use the name of the card as the key, and the card's numeric value as the value. For example, the key-value pair for the queen of hearts is

  'Queen of Hearts':10
- And the key-value pair for the 8 of diamonds is

  '8 of Diamonds':8

Program Output (with input shown in bold)

```
How many cards should I deal? 5 Enter
8 of Hearts
5 of Diamonds
5 of Hearts
Queen of Clubs
10 of Spades
Value of this hand: 38
```

Storing Names and Birthdays in a Dictionary

- we look at a program that keeps your friends' names and birthdays in a dictionary. Each entry in the dictionary uses a friend's name as the key, and that friend's birthday as the value. You can use the program to look up your friends' birthdays by entering their names.
- The program displays a menu that allows the user to make one of the following choices:
    1. Look up a birthday
    2. Add a new birthday
    3. Change a birthday
    4. Delete a birthday
    5. Quit the program
- The program initially starts with an empty dictionary

Program Output (with input shown in bold)

```
Friends and Their Birthdays
---------------------------
1. Look up a birthday
2. Add a new birthday
3. Change a birthday
4. Delete a birthday
5. Quit the program

Enter your choice: 2 Enter
Enter a name: Cameron Enter
Enter a birthday: 10/12/1990 Enter

Friends and Their Birthdays
---------------------------
1. Look up a birthday
2. Add a new birthday
3. Change a birthday
4. Delete a birthday
5. Quit the program

Enter your choice: 2 Enter
Enter a name: Kathryn Enter
Enter a birthday: 5/7/1989 Enter
```

```
Friends and Their Birthdays
---------------------------
1. Look up a birthday
2. Add a new birthday
3. Change a birthday
4. Delete a birthday
5. Quit the program

Enter your choice: 1 Enter
Enter a name: Cameron Enter
10/12/1990

Friends and Their Birthdays
---------------------------
1. Look up a birthday
2. Add a new birthday
3. Change a birthday
4. Delete a birthday
5. Quit the program

Enter your choice: 1 Enter
Enter a name: Kathryn Enter
5/7/1989

Friends and Their Birthdays
---------------------------
1. Look up a birthday
2. Add a new birthday
3. Change a birthday
```

```
4. Delete a birthday
5. Quit the program

Enter your choice: 3 Enter
Enter a name: Kathryn Enter
Enter the new birthday: 5/7/1988 Enter

Friends and Their Birthdays
---------------------------
1. Look up a birthday
2. Add a new birthday
3. Change a birthday
4. Delete a birthday
5. Quit the program

Enter your choice: 1 Enter
Enter a name: Kathryn Enter
5/7/1988

Friends and Their Birthdays
---------------------------
1. Look up a birthday
2. Add a new birthday
3. Change a birthday
4. Delete a birthday
5. Quit the program

Enter your choice: 4 Enter
Enter a name: Cameron Enter
```

```
Friends and Their Birthdays
---------------------------
1. Look up a birthday
2. Add a new birthday
3. Change a birthday
4. Delete a birthday
5. Quit the program

Enter your choice: 1 Enter
Enter a name: Cameron Enter
Not found.

Friends and Their Birthdays
---------------------------
1. Look up a birthday
2. Add a new birthday
3. Change a birthday
4. Delete a birthday
5. Quit the program

Enter your choice: 5 Enter
```

Aletheia University
資訊工程學系

Sets

- A *set* is an object that stores a collection of data in the same way as mathematical sets.
- Here are some important things to know about sets:
  - All the elements in a set must be unique. No two elements can have the same value.
  - Sets are unordered.
  - The elements that are stored in a set can be of different data types.

Creating a Set

- To create a set, you have to call the built-in **set** function.

```
myset = set()
```

- The individual elements of the object that you pass as an argument become elements of the set. Here is an example:

```
myset = set(['a', 'b', 'c’])
```

- the myset variable references a set containing the elements 'a', 'b', and 'c’.
- If you pass a string as an argument to the set function, each individual character in the string becomes a member of the set.

```
myset = set('abc')
```

Getting the Number of Elements In a Set

```
>>> myset = set([1, 2, 3, 4, 5])
>>> len(myset)
5
>>>
```

Adding and Removing Elements

- Sets are mutable objects, so you can add items to them and remove items from them. You use the add method to add an element to a set.

```
>>> myset = set()
>>> myset.add(1)
>>> myset.add(2)
>>> myset.add(3)
>>> myset
{1, 2, 3}
```

Aletheia University
資訊工程學系

- You can add a group of elements to a set all at one time with the update method.

```
>>> myset = set([1, 2, 3])
>>> myset.update([4, 5, 6])
>>> myset
{1, 2, 3, 4, 5, 6}
>>> set1 = set([1, 2, 3])
>>> set2 = set([8, 9, 10])
>>> set1.update(set2)
```

- You can remove an item from a set with either the **remove** method or the **discard** method.

```
myset = set([1, 2, 3, 4, 5])
>>> myset
{1, 2, 3, 4, 5}
>>> myset.remove(1)
>>> myset
{2, 3, 4, 5}
>>> myset.discard(5)
>>> myset
{2, 3, 4}
```

- You can clear all the elements of a set by calling the clear method.

```
>>> myset = set([1, 2, 3, 4, 5]) Enter
>>> myset Enter
{1, 2, 3, 4, 5}
>>> myset.clear() Enter
>>> myset Enter
set()
```

Using the for Loop to Iterate over a Set

- You can use the for loop in the following general format to iterate over all the elements in a set:

  for *var* in *set:*

  statement

  statement

  etc.

```
>>> myset = set(['a', 'b', 'c']) Enter
>>> for val in myset: Enter
        print(val) Enter Enter

a
c
b
>>>
```

- *var* is the name of a variable and *set* is the name of a set.

Using the in and not in Operators to Test for a Value in a Set

- You can use the in operator to determine whether a value exists in a set.

```
>>> myset = set([1, 2, 3]) Enter
>>> if 1 in myset: Enter
        print('The value 1 is in the set.') Enter Enter

The value 1 is in the set.
>>>
```

- You can also use the not in operator to determine if a value does not exist in a set.

```
>>> myset = set([1, 2, 3]) Enter
>>> if 99 not in myset: Enter
        print('The value 99 is not in the set.') Enter Enter

The value 99 is not in the set.
>>>
```

Finding the Union of Sets

- The union of two sets is a set that contains all the elements of both sets. In Python, you can call the **union** method to get the union of two sets.

  set1.union(*set2*)

```
>>> set1 = set([1, 2, 3, 4]) Enter
>>> set2 = set([3, 4, 5, 6]) Enter
>>> set3 = set1.union(set2) Enter
>>> set3 Enter
{1, 2, 3, 4, 5, 6}
>>>
```

Aletheia University 資訊工程學系

- You can also use the | operator to find the union of two sets. Here is the general format of an expression using the | operator with two sets:

  set1 | set2

```
>>> set1 = set([1, 2, 3, 4]) Enter
>>> set2 = set([3, 4, 5, 6]) Enter
>>> set3 = set1 | set2 Enter
>>> set3 Enter
{1, 2, 3, 4, 5, 6}
>>>
```

Finding the Intersection of Sets

- The intersection of two sets is a set that contains only the elements that are found in both sets. In Python, you can call the intersection method to get the intersection of two sets.

  set1.intersection(*set2*)

```
>>> set1 = set([1, 2, 3, 4]) Enter
>>> set2 = set([3, 4, 5, 6]) Enter
>>> set3 = set1.intersection(set2) Enter
>>> set3 Enter
{3, 4}
>>>
```

Aletheia University 資訊工程學系

- You can also use the & operator to find the intersection of two sets.

  set1 & set2

```
>>> set1 = set([1, 2, 3, 4]) Enter
>>> set2 = set([3, 4, 5, 6]) Enter
>>> set3 = set1 & set2 Enter
>>> set3 Enter
{3, 4}
>>>
```

Finding the Difference of Sets

- The difference of set1 and set2 is the elements that appear in set1 but do not appear in set2. In Python, you can call the difference method to get the **difference** of two sets.

  set1.difference(*set2*)

```
>>> set1 = set([1, 2, 3, 4]) Enter
>>> set2 = set([3, 4, 5, 6]) Enter
>>> set3 = set1.difference(set2) Enter
>>> set3 Enter
{1, 2}
>>>
```

Aletheia University 資訊工程學系

- You can also use the − operator to find the difference of two sets.

  set1 − set2

```
>>> set1 = set([1, 2, 3, 4]) Enter
>>> set2 = set([3, 4, 5, 6]) Enter
>>> set3 = set1 - set2 Enter
>>> set3 Enter
{1, 2}
>>>
```

Finding the Symmetric Difference of Sets

- The symmetric difference of two sets is a set that contains the elements that are not shared by the sets.
- In other words, it is the elements that are in one set but not in both. In Python, you can call the **symmetric_difference** method to get the symmetric difference of two sets.

  set1.symmetric_difference(*set2*)

```
>>> set1 = set([1, 2, 3, 4]) Enter
>>> set2 = set([3, 4, 5, 6]) Enter
>>> set3 = set1.symmetric_difference(set2) Enter
>>> set3 Enter
{1, 2, 5, 6}
>>>
```

Aletheia University
資訊工程學系

- You can also use the ^ operator to find the symmetric difference of two sets.

  set1 ^ set2

```
>>> set1 = set([1, 2, 3, 4]) Enter
>>> set2 = set([3, 4, 5, 6]) Enter
>>> set3 = set1 ^ set2 Enter
>>> set3 Enter
{1, 2, 5, 6}
>>>
```

Aletheia University 資訊工程學系

Finding Subsets and Supersets

- Suppose you have two sets, and one of those sets contains all of the elements of the other set.

  set1 = set([1, 2, 3, 4])

  set2 = set([2, 3])

- set1 contains all the elements of set2, which means that set2 is a ***subset*** of set1. It also means that set1 is a ***superset*** of set2.
- In Python, you can call the **issubset** method to determine whether one set is a subset of another.

  set2.issubset(*set1*)

- You can call the issuperset method to determine whether one set is a **superset** of another.

  set1.issuperset(*set2*)

Aletheia University 資訊工程學系

```
>>> set1 = set([1, 2, 3, 4]) Enter
>>> set2 = set([2, 3]) Enter
>>> set2.issubset(set1) Enter
True
>>> set1.issuperset(set2) Enter
True
>>>
```

- You can also use the <= operator to determine whether one set is a subset of another and the >= operator to determine whether one set is a superset of another.

set2 <= set1

```
>>> set1 = set([1, 2, 3, 4]) Enter
>>> set2 = set([2, 3]) Enter
>>> set2 <= set1 Enter
True
>>> set1 >= set2 Enter
True
>>> set1 <= set2 Enter
False
```

Aletheia University 資訊工程學系

- The program creates two sets: one that holds the names of students on the baseball team, and another that holds the names of students on the basketball team. The program then performs the following operations:
  1. It finds the intersection of the sets to display the names of students who play both sports.
  2. It finds the union of the sets to display the names of students who play either sport.
  3. It finds the difference of the baseball and basketball sets to display the names of students who play baseball but not basketball.
  4. It finds the difference of the basketball and baseball (*basketball – baseball*) sets to display the names of students who play basketball but not baseball. It also finds the difference of the baseball and basketball (*baseball – basketball*) sets to display the names of students who play baseball but not basketball.
  5. It finds the symmetric difference of the basketball and baseball sets to display the names of students who play one sport but not both.

Aletheia University
資訊工程學系

Program Output

```
The following students are on the baseball team:
Jodi
Aida
Carmen
Alicia

The following students are on the basketball team:
Sarah
Eva
Alicia
Carmen

The following students play both baseball and basketball:
Alicia
Carmen

The following students play either baseball or basketball:
Sarah
Alicia
Jodi
Eva
Aida
Carmen

The following students play baseball but not basketball:
Jodi
Aida
The following students play basketball but not baseball:
Sarah
Eva

The following students play one sport but not both:
Sarah
Aida
Jodi
Eva
```

Serializing Objects

- Sometimes you need to store the contents of a complex object, such as a dictionary or a set, to a file.
- The easiest way to save an object to a file is to serialize the object.
- When an object is ***serialized***, it is converted to a stream of bytes that can be easily stored in a file for later retrieval.
- In Python, the process of serializing an object is referred to as ***pickling***.
- The Python standard library provides a module named **pickle** that has various functions for serializing, or pickling, objects.

真理大學 Aletheia University 資訊工程學系

- Once you import the pickle module, you perform the following steps to pickle an object:
  1. You open a file for binary writing.
  2. You call the pickle module's dump method to pickle the object and write it to the specified file.
  3. After you have pickled all the objects that you want to save to the file, you close the file.
- Let's take a more detailed look at these steps. To open a file for binary writing, you use 'wb' as the mode when you call the open function.

  outputfile = open('mydata.dat', 'wb')

- Once you have opened a file for binary writing, you call the pickle module's dump function.

  pickle.dump(*object*, *file*)

- *object* is a variable that references the object you want to pickle, and *file* is a variable that references a file object.

```
>>> import pickle Enter
>>> phonebook = {'Chris' : '555-1111', Enter
                 'Katie' : '555-2222', Enter
                 'Joanne' : '555-3333'} Enter
>>> output_file = open('phonebook.dat', 'wb') Enter
>>> pickle.dump(phonebook, output_file) Enter
>>> output_file.close() Enter
>>>
```

- At some point, you will need to retrieve, or unpickle, the objects that you have pickled. Here are the steps that you perform:
  1. You open a file for binary reading.
  2. You call the pickle module's load function to retrieve an object from the file and unpickle it.
  3. After you have unpickled all the objects that you want from the file, you close the file.

- To open a file for binary reading, you use 'rb' as the mode when you call the open function.

  inputfile = open('mydata.dat', 'rb’)

- Once you have opened a file for binary reading, you call the pickle module’s load function.

  object = pickle.load(*file*)

```
>>> import pickle Enter
>>> input_file = open('phonebook.dat', 'rb') Enter
>>> pb = pickle.load(inputfile) Enter
>>> pb Enter
{'Chris': '555-1111', 'Joanne': '555-3333', 'Katie': '555-2222'}
>>> input_file.close() Enter
>>>
```

Program Output (with input shown in bold)

```
Name: Angie Enter
Age: 25 Enter
Weight: 122 Enter
Enter more data? (y/n): y Enter
Name: Carl Enter
Age: 28 Enter
Weight: 175 Enter
Enter more data? (y/n): n Enter
```

Aletheia University 資訊工程學系

```
# This program demonstrates object pickling.
import pickle

def main():
    again = 'y' # To control loop repetition

    # Open a file for binary writing.
    output_file = open('info.dat', 'wb')

    # Get data until the user wants to stop.
    while again.lower() == 'y':
        # Get data about a person and save it.
        save_data(output_file)

        # Does the user want to enter more data?
        again = input('Enter more data? (y/n): ')

    # Close the file.
    output_file.close()

# The save_data function gets data about a person,
# stores it in a dictionary, and then pickles the
# dictionary to the specified file.
def save_data(file):
    # Create an empty dictionary.
    person = {}

    # Get data for a person and store
    # it in the dictionary.
    person['name'] = input('Name: ')
    person['age'] = int(input('Age: '))
    person['weight'] = float(input('Weight: '))

    # Pickle the dictionary.
    pickle.dump(person, file)

# Call the main function.
main()
```