CS65: Introduction to Computer Science

Dictionary

Md Alimoor Reza
Assistant Professor of Computer Science

Topic

- Dictionary:
  - What is it and why do we need it?
- Dictionary creation
- Dictionary manipulation
  - Adding or updating
  - Removing
- Iterating over a dictionary
- Tuple

Dictionary

- **List**: Access a position using **only numeric index**

```
student_id_list = [1002, 1003, 1004, 1005]
student_id_list[0]
student_id_list[1]
student_id_list[2]
student_id_list[3]
```

- **Dictionary**: Access an element using (eg number, string, character) as index (keys) to associate to something else (values)

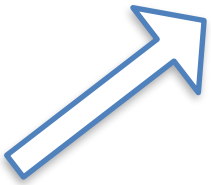
- Dictionary is an object that stores a collection of data
  - collection of key-value pairs
  - variable_name = { key_1 : $value_1$, key_2 : $value_2$, …, key_N : $value_N$ }

id	name
1002	Jack
1003	Daja
1004	Matt
1005	Simran

```
# dictionary with student ids as keys
students = {1002:"Jack", 1003:"Daja", 1004:"Matt", 1005:"Simran"}
```

Dictionary

- Dictionary is collection of key-value pairs or mapping between them
- variable_name = { key_1 : $value_1$, key_2 : $value_2$, ..., key_N : $value_N$ }
- The keys **must be unique**
  - Unlike previous example, keys below are **names**



name	id
Jack	1002
Daja	1003
Matt	1004
Simran	1005

```
# dictionary with student names as keys
students = {"Jack": 1002, "Daja":1003, "Matt":1004, "Simran":1005}
```

Retrieving a Value from a Dictionary

- Given *variable_name* = { key_1 : $value_1$, key_2 : $value_2$, …, key_N : $value_N$ }
- Use the syntax: *variable_name*[key]
- Since keys are **unique**, accessing via a key will return a specific value

```
# dictionary with student ids as keys
students = {1002:"Jack", 1003:"Daja", 1004:"Matt", 1005:"Simran"}
```

```
print("students[1002] --> ", students[1002])
print("students[1003] --> ", students[1003])
print("students[1004] --> ", students[1004])
print("students[1005] --> ", students[1005])
```

```
>>> %Run lec13.py

  students[1002] -->  Jack
  students[1003] -->  Daja
  students[1004] -->  Matt
  students[1005] -->  Simran
```

- Do you find any similarity with List?

Topic

- Dictionary:
  - What is it and why do we need it?
- Dictionary creation
- Dictionary manipulation
  - Adding or updating
  - Removing
- Iterating over a dictionary
- Tuple

Creating a Dictionary

- Several ways to create a dictionary:

- **Approach 1:** an empty dictionary

```
my_dict = {}
```

- **Approach 2:** with predefined entries

```
dict_student_scores = {'Reza':45, 'Chris':50, 'Sigi': 55}

dict_name_parts = {'Papa': 'John', 'Christiano':'Ronaldo', 'LeBron':'James'}

dict_random = {1:'one', (1,2):"two", None:"None keyword"}
```

Creating a Dictionary

- Several ways to create a dictionary:

- **Approach 3:** with **dict()** keyword and arguments <u>unquoted-strings</u>

```
my_dict = dict(Age=29, Tel=3405, Name='Kate')
```

- **Approach 4:** with **dict()** keyword and list of two entries

```
my_dict = dict([[10, '10^1'], [100, '10^2'], [1000, '10^3']])
```

Exercise

- **<u>Exercise 1:</u>** Create a dictionary that can save names of different exams and their scores as key-value pairs. More specifically, your dictionary should store information about the following three tests:
  - '***Quiz 1***' and its ***score***
  - '***Quiz 2***' and its ***score***
  - '***Midterm exam***' and its ***score***

- **<u>Hint:</u>**
  - First, decide — what will be your **keys** and what will be your **values**
  - Second, pick a reasonable variable name for the dictionary
  - Finally, create your new dictionary

Exercise

- **<u>Exercise 2:</u>** Create a dictionary that can save names of 3 different students and a list of <u>5 scores</u> as ***key:value*** pairs.

- More specifically, your dictionary should store information:
  - $Student_1$ name and a list of scores for student 1
  - $Student_2$ name and a list of scores for student 2
  - $Student_3$ name and a list of scores for student 4

Topic

- Dictionary:
  - What is it and why do we need it?
- Dictionary creation
- Dictionary manipulation
  - Adding or updating
  - Removing
- Iterating over a dictionary
- Tuple

Adding or Updating `key:value` Pairs to a Dictionary

- Use the following syntax to add a key-value pair to a dictionary
- *variable_name*[key] = value

```
my_dict = {}

my_dict['Reza']  = 45
my_dict['Chris'] = 50
my_dict['Sigi']  = 55
```

Updating Dictionary Items

- It is also possible to update a dictionary using `.update()` method

```
my_dict = {1:"one", 2:"two", 3:"three"}
print(my_dict)


# update an entry with key index
my_dict[3] = "THREE"
print(my_dict)

# update a dictionary using .update() method
new_dict_entries = {4:"four", 5:"five"}
my_dict.update(new_dict_entries)

print(my_dict)
```

```
>>> %Run lec14_dictionary_modification.py

  {1: 'one', 2: 'two', 3: 'three'}
  {1: 'one', 2: 'two', 3: 'THREE'}
  {1: 'one', 2: 'two', 3: 'THREE', 4: 'four', 5: 'five'}
```

Removing Dictionary Item

- `pop()` method: remove an item by a given key
- `del` keyword: removes item indexed by a given key and [] operator (similar to list removal)
- `popitem()` method: remove an arbitrary item

```
# --------- dictionary remove ---------------
my_dict = {1:"one", 2:"two", 3:"three", 4:"four", 5:"five"}


del my_dict[4]
print(my_dict)

my_dict.pop(3)
print(my_dict)

my_dict.popitem()
print(my_dict)
```

```
{1: 'one', 2: 'two', 3: 'three', 5: 'five'}
{1: 'one', 2: 'two', 5: 'five'}
{1: 'one', 2: 'two'}
```

Topic

- Dictionary:
  - What is it and why do we need it?
- Dictionary creation
- Dictionary manipulation
  - Adding or updating
  - Removing
- Iterating over a dictionary
- Tuple

Iterating through a Dictionary: Approach 1

- Use the following syntax to iterate through a dictionary
  - more like accessing a list using value `for` loop

-
```
for key_var in dictionary_variable_name:
    print(dictionary_variable_name[key_var])
```

- It will process every **`key_var`** in the dictionary, the following line will access corresponding value associated with the **`key_var`**

Example: Iterating through a Dictionary

```
dict_states = {'Iowa': 'IA', 'Indiana':'IN', 'Virginia':'VA', 'Pennsylvania':'PA'}

for key in dict_states:

    print('key=', key, ' and value is ', dict_states[key])
```

```
key= Iowa  and value is  IA
key= Indiana  and value is  IN
key= Virginia  and value is  VA
key= Pennsylvania  and value is  PA
```

Iterating through a Dictionary: Approach 2

- Another way to iterate through a dictionary is to use following syntax

- ```
  for (key, value) in dictionary_variable_name.items():
      print( "key = ", key, " and value is = ", value)
  ```

- It will process every `key` and `value` pair in the dictionary
  ```

Example: Iterating through a Dictionary

```
dict_states = {'Iowa': 'IA', 'Indiana':'IN', 'Virginia':'VA', 'Pennsylvania':'PA'}

for (key, value) in dict_states.items():

    print('key=', key, ' and value is ', value)
```

```
key= Iowa  and value is  IA
key= Indiana  and value is  IN
key= Virginia  and value is  VA
key= Pennsylvania  and value is  PA
```

Example: Iterating through keys

```
dict_states = {'Iowa': 'IA', 'Indiana':'IN', 'Virginia':'VA', 'Pennsylvania':'PA'}

for key in dict_states.keys():

    print('key=', key)
```

```
key= Iowa
key= Indiana
key= Virginia
key= Pennsylvania
```

Example: Iterating through values

```
dict_states = {'Iowa': 'IA', 'Indiana':'IN', 'Virginia':'VA', 'Pennsylvania':'PA'}

for value in dict_states.values():

    print('value =', value)
```

```
value = IA
value = IN
value = VA
value = PA
```

Exercise

- **Exercise 1:** Now you try out these four different ways of iterating through the dictionary

- **Exercise 2:** Iterate through the dictionary you have created in an earlier exercise, e.g., **exam scores**

Dictionary Operations

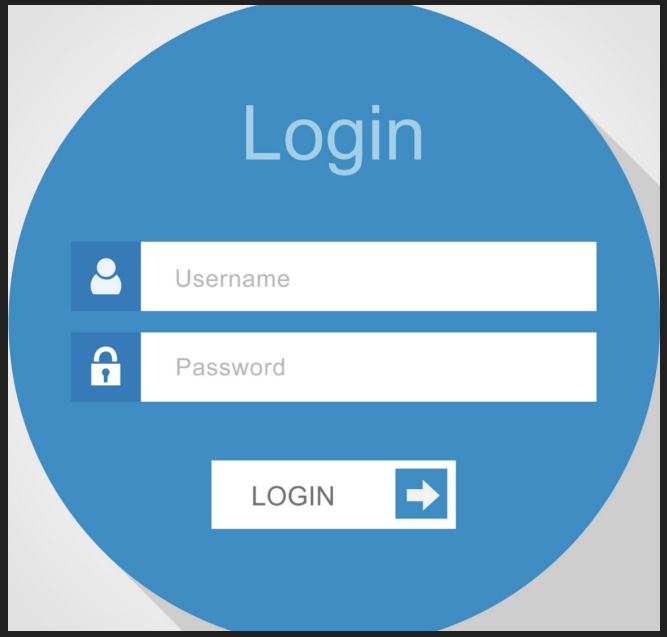
function/method/operation	usage
len: # of key-value pairs.	len(d)
indexing: by key	d[k]
get: (use optional parameter 'default' if not found)	d.get(k) d.get(k, default)
del: remove a key-value pair	del d[k]
in, not in: test key's presence	k in d k not in d
clear: remove all key-value pairs	d.clear()
copy: create a shallow copy	d.copy()
keys, values, items: get the keys, values, or key-val pairs	d.keys() d.values() d.items()
pop: pop value at k (or return default) **popitem()**: pop any value	d.pop(k) d.pop(k,default) d.popitem()
update: insert all of another dict's key-value pairs	d_receiver.update(d_supplier)

Application of Dictionaries

Almost certainly, every authentication system is implemented using a dictionary

key: username

value: password



Exercise #1

1. Set up a dictionary of usernames and passwords

```
passwords = {"urness": "I<3CS", "student19": "Spring2025!"}
passwords["user123"] = "password123"
```

1. Prompt the user for a username and password

```
username = input("What is your username: ")
pw = input("What is your password: ")
```

If the username is NOT in the dictionary, response should be **"username not found"**

If the username is in the dictionary, but the password given does not match the password stored in the dictionary, response should be **"password incorrect"**

If the username and password match, response should be **"access granted"**

Exercise #2

```
input_string = input("please input some words: ")
word_list = input_string.split()

word_count = {}
```

Develop a **word count** dictionary:

The key should be a word entered, the value should be the number of times is occured

```
>>> %Run Exercise1.py
 please input some words: the cat in the hat on the cat
 {'the': 3, 'cat': 2, 'in': 1, 'hat': 1, 'on': 1}
```

Exercise2Starter.py
Available in Xuexitong

Exercise #2 hint

Develop a **word count** dictionary:

The key should be a word entered, the value should be the number of times is occurred

```
# exercise #2

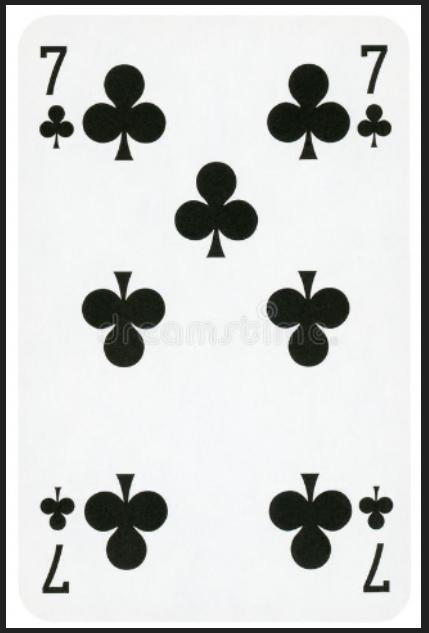
input_string = input("please input some words: ")
word_list = input_string.split()

word_count = {}
for word in word_list:
    print(word)
    ### your code goes here
    ### if the word is in the dictionary....
    ### otherwise if the word is (not yet) in the dictionary...

print(word_count)
```

Dictionaries as Cards

```
my_card = {'value': "7", 'suit': "clubs"}
your_card = {'value': "K", 'suit': "hearts"}
```



Exercise #3

Describe what is happening here in English:

```
deck = []
for suit in ['spades', 'clubs', 'hearts', 'diamonds']:
    for rank in ['A','2','3','4','5','6','7','8','9','T','J','Q','K']:
        deck.append({'value':rank, 'suit':suit})
```

Cards.py

What will be the first card in the deck?

What will be outpu

```
print(deck[0])
print(deck[51])
```

ecute:

Shuffling the deck

The Random module has a simple way of randomizing the elements of a list:

```
import random


deck = []
for suit in ['spades', 'clubs', 'hearts', 'diamonds']:
    for rank in ['A','2','3','4','5','6','7','8','9','T','J','Q','K']:
        deck.append({'value':rank, 'suit':suit})

# The deck is actually a list of 52 dictionaries.

# The shuffle method can be used to shuffle the deck:
random.shuffle(deck)
print("here are 5 cards after shuffling")
for i in range(5):
    print(deck[i])
```

Cards2.py