ITD62-123 COMPUTER PROGRAMMING

IMI62-122 FUNDAMENTAL OF COMPUTER PROGRAMMING

Theerat Saichoo & Yunyong Punsawad
School of Informatics, Walailak University

Chapter 6
Structure

python™

Python Dictionaries

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is unordered, changeable and does not allow duplicates.
- Dictionaries are written with curly brackets, and have keys and values.
- The items does not have a defined order, We cannot refer to an item by using an index.
- Dictionaries are changeable, meaning that we can change, add or remove items.

Python Dictionaries

- **Example**

```
thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
}
```

Duplicates Not Allowed

- Dictionaries cannot have two items with the same key.
- Duplicate values will overwrite existing values.
- **Example**

```
thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        ~~"year": 1964,~~
        "year": 2020
}
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

Dictionary Items - Data Types

- The values in dictionary items can be of any data type.
- **Example** - String, int, boolean, and list data types:

```
thisdict = {
        "brand": "Ford",
        "electric": False,
        "year": 1964,
        "colors": ["red", "white", "blue"]
}
```

Access Dictionary Items

- We can access the items of a dictionary by calling the method get().

```
thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
}

x = thisdict.get("model")

print(x)
```

```
Mustang
```

Access Dictionary Items

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

```
thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
}

x = thisdict.keys()

print(x)
```

```
dict_keys(['brand', 'model', 'year'])
```

Access Dictionary Items

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

```
thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
}

x = thisdict.values()

print(x)
```

```
dict_values(['Ford', 'Mustang', 1964])
```

Change Dictionary Items

- The update() method will update the dictionary with the items from the given argument.
- The argument must be a dictionary, or an iterable object with key:value pairs.

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

```
thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
}

thisdict.update({"year": 2020})
```

Change Dictionary Items

- The update() method will update the dictionary with the items from a given argument. If the item does not exist, the item will be added.
- The argument must be a dictionary, or an iterable object with key:value pairs.

```
thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
}

thisdict.update({"color": "red"})
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

Remove Dictionary Items

- The pop() method removes the item with the specified key name.

```
thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
}

thisdict.pop("model")
```

Remove Dictionary Items

- The del keyword can also delete the dictionary completely.

```
thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
}

del thisdict
```

Remove Dictionary Items

- The clear() method empties the dictionary.

```
thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
}

thisdict.clear()
```

Loop Dictionaries

- We can loop through a dictionary by using a for loop.
- When looping through a dictionary, the return value are the *keys* of the dictionary, but there are methods to return the *values* as well.
- Example – return keys

```
thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
}

for x in thisdict.keys():
        print(x)
```

```
brand
model
year
```

Loop Dictionaries

- Example – return values

```
thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
}

for x in thisdict.values():
        print(x)
```

```
Ford
Mustang
1964
```

Loop Dictionaries

- Example – return items

```
thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
}

for x, y in thisdict.items():

        print(x, y)
```

```
brand Ford
model Mustang
year 1964
```

Copy Dictionaries

- There are ways to make a copy, one way is to use the built-in Dictionary method copy().

```
thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
}
mydict = thisdict.copy()

print(mydict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Nested Dictionaries

- A dictionary can contain dictionaries, this is called nested dictionaries
- Create three dictionaries, then create one dictionary that will contain the other three dictionaries:

```
child1 = {
  "name" : "Emil",
  "year" : 2004
}
child2 = {
  "name" : "Tobias",
  "year" : 2007
}
```

```
child3 = {
  "name" : "Linus",
  "year" : 2011
}
myfamily = {
  "child1" : child1,
  "child2" : child2,
  "child3" : child3
}
```

Python JSON

- JSON is a syntax for storing and exchanging data.
- JSON is text, written with JavaScript object notation.
- Python has a built-in package called json, which can be used to work with JSON data.

```
>> import json
```

Convert from Python to JSON

- If you have a Python object, you can convert it into a JSON string by using the json.dumps() method.

```
import json

# a Python object (dict):
x = {
        "name": "John",
        "age": 30,
        "city": "New York"
}

# convert into JSON:
y = json.dumps(x)

# the result is a JSON string:
print(y)
```

```
{"name": "John", "age": 30, "city": "New York"}
```

Convert from JSON to Python

- If you have a JSON string, you can parse it by using the json.loads() method.

```
import json

# some JSON:
x =  '{ "name":"John", "age":30, "city":"New York"}'

# parse x:
y = json.loads(x)

# the result is a Python dictionary:
print(y)
```

```
{'name': 'John', 'age': 30, 'city': 'New York'}
```

Special thanks to W3Schools

Class activity 8

1. จงออกแบบ Python Dictionary สำหรับเก็บข้อมูลของนักศึกษา ซึ่งประกอบด้วย ชื่อ นามสกุล รหัสนักศึกษา อายุ และเกรดเฉลี่ยสะสม
2. จงออกแบบ Python Dictionary สำหรับเก็บข้อมูลที่อยู่ของนักศึกษา ซึ่งประกอบด้วย บ้านเลขที่ หมู่ที่ ตำบล อำเภอ จังหวัด และรหัสไปรษณีย์
3. จงออกแบบ Nested- Python Dictionary เก็บข้อมูลนักศึกษาและที่อยู่