Business Programming (using Python)

Zhaohu (Jonathan) Fan

Information Technology Management

Scheller College of Business

Georgia Institute of Technology

October 3, 2023

Main topics

- Data Structures
  - Dictionaries
  - Exercises

In-class presentation

- **Question (HW#5|Problem #4)?**
    - Objective:
        - Prompt the user to enter **numbers** .
        - Store numbers in a list.
        - Compute and display the **maximum** and **minimum numbers** .
    - Example:
        - Entered numbers: 4, 10, 15, 2, 7
        - **Maximum** : 15
        - **Minimum** : 2

In-class presentation

- How do you input numbers into a Python list?
    - How do you finalize your entries to find the **maximum** and **minimum** values?
        - Example:
            - Entered numbers: 4, 10, 15, 2, 7
            - **Maximum** : 15
            - **Minimum** : 2
- **Brianna Abreu** will present her answer to this question on Thursday.

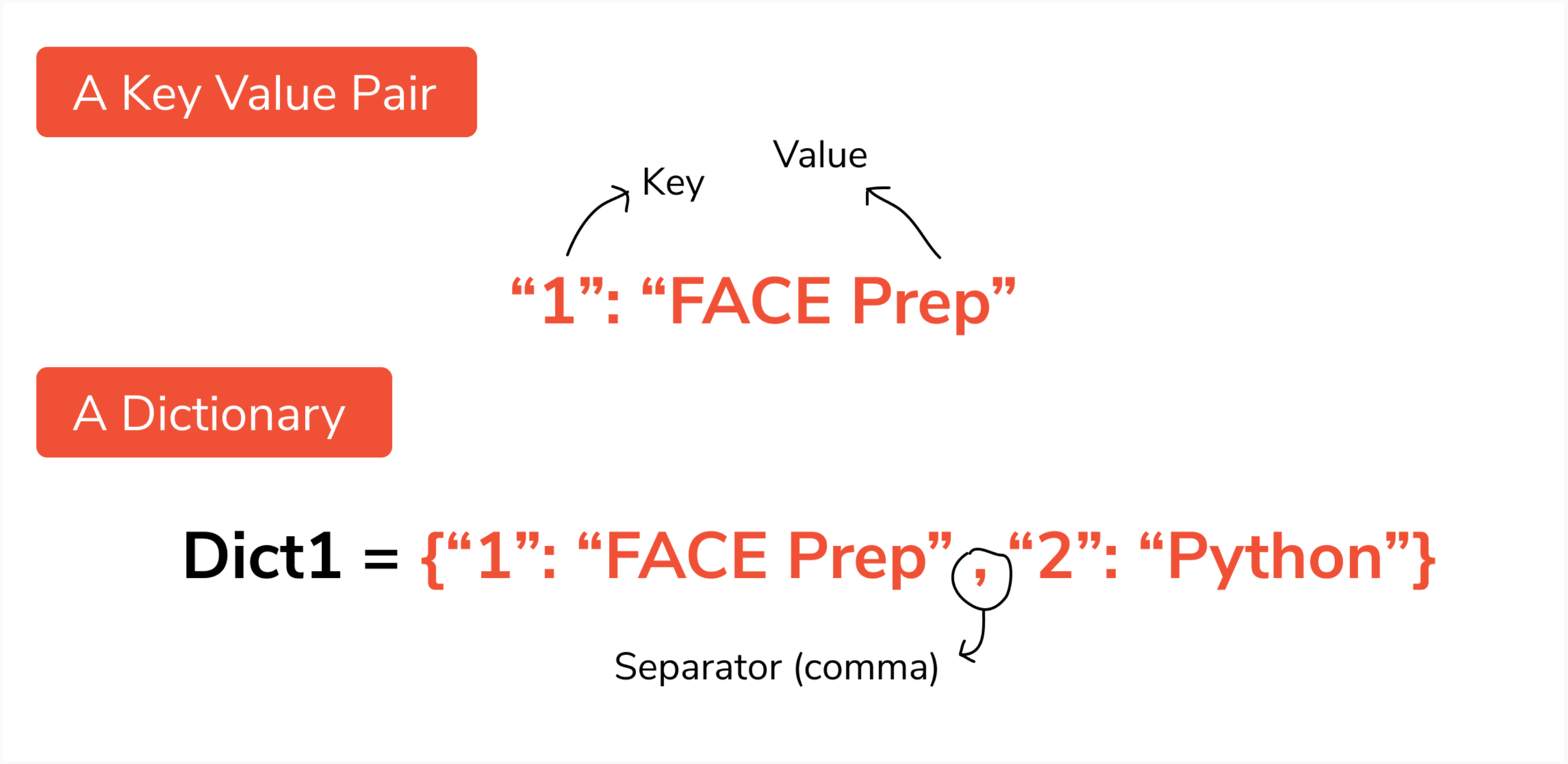
Data Structures - Dictionaries (II)

Dictionaries

- Please click on the link provided below.
    - Built-In Data Structures:Dictionaries

What is dictionary in Python?

- Python dictionary is an unordered collection of items. Each item of a dictionary has a `key/value` pair.
    - Dictionaries are optimized to retrieve **values** when the **key** is known.



Creating Python dictionary

- Creating a dictionary is as simple as placing items inside curly braces `{}` separated by commas or the **dict()** built-in function.
  - An item has a `key` and a corresponding `value` that is expressed as a pair {**key: value**}.
  - The `key` and the `value` is separated by a colon `:`. Items are separated from each other by a comma `,`.

Python

```
>>> dict = { }  #empty dictionary
>>> dict = {1:'Python',2:'Java',3:'C++'}
```

Example

- **Dictionary is mutable** i.e., **value can be updated**.
    - **`Key`** must be unique and immutable. **`Value`** is accessed by key. **`Value`** can be updated while **`key`** cannot be changed.
    - Dictionary is known as Associative array since the Key works as Index and they are decided by the user.

Python

```
>>> dict = { }  #empty dictionary
>>> dict = {1:'Python',2:'Java',3:'C++'}
```

Accessing elements from Dictionary

- While indexing is used with other data types to access values, a dictionary uses `keys`. Keys can be used either inside square brackets `[]` or with the `get()` method.
  - If we use the square brackets `[]`, `KeyError` is raised in case a key is not found in the dictionary. On the other hand, the `get()` method returns `None` if the key is not found.
  - iterate all elemnet using for loop for `keys()` method, `keys()` method return list of all keys in dictionary.

Changing and adding Dictionary

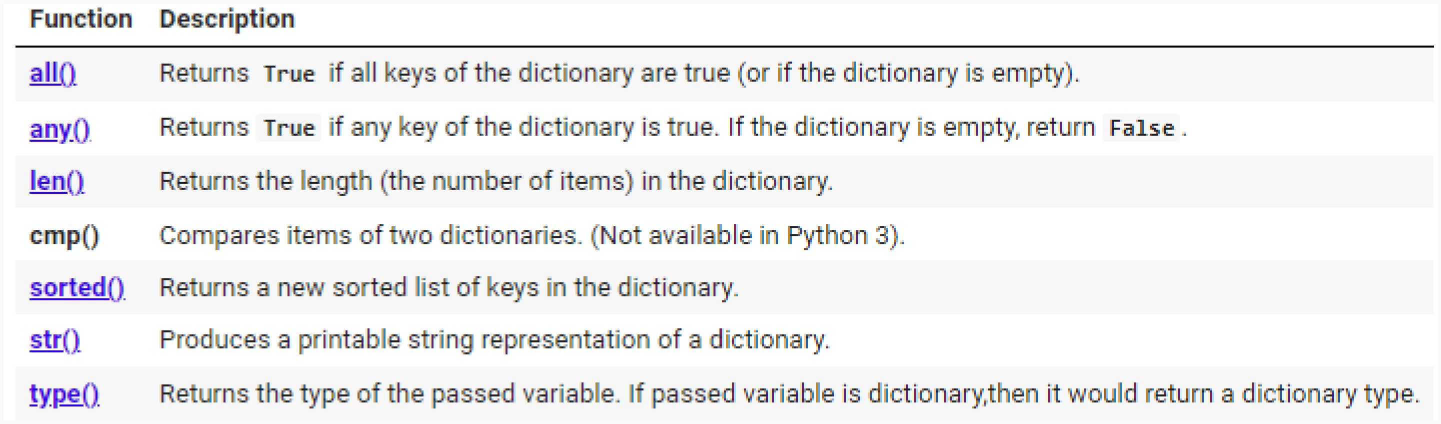
- Dictionaries are mutable. We can add new items or change the value of existing items using an assignment operator.
    - If the key is already present, then the existing value gets updated. In case the key is not present, a new (key: value) pair is added to the dictionary.

Removing elements from Dictionary

- We can remove a particular item in a dictionary by using the `pop()` method. This method removes an item with the provided `key` and returns the `value`.
    - The `popitem()` method can be used to remove and return an arbitrary `(key, value)` item pair from the dictionary. All the items can be removed at once, using the `clear()` method.
    - We can also use the `del` keyword to remove individual items or the entire dictionary itself.

Dictionary Built-in Dictionary functions

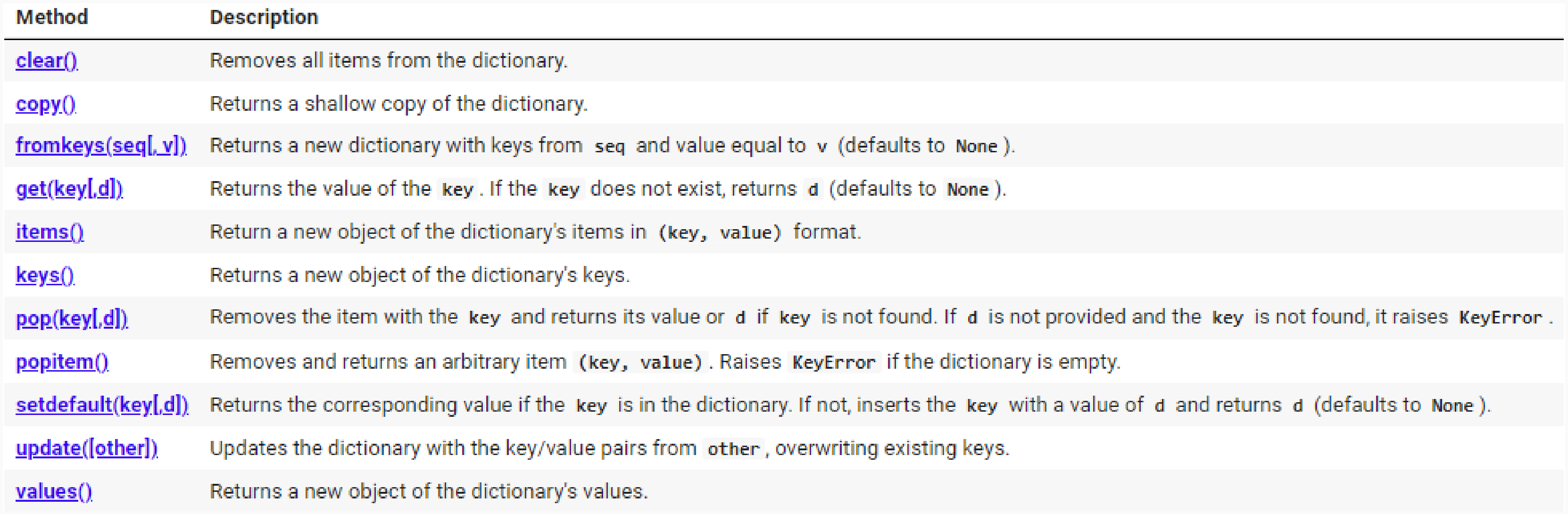
- Built-in functions like `all()`, `any()`, `len()`, `cmp()`, `sorted()`, `str()`, `typ()`, etc. are commonly used with dictionaries to perform different tasks.



Function	Description
all()	Returns `True` if all keys of the dictionary are true (or if the dictionary is empty).
any()	Returns `True` if any key of the dictionary is true. If the dictionary is empty, return `False`.
len()	Returns the length (the number of items) in the dictionary.
cmp()	Compares items of two dictionaries. (Not available in Python 3).
sorted()	Returns a new sorted list of keys in the dictionary.
str()	Produces a printable string representation of a dictionary.
type()	Returns the type of the passed variable. If passed variable is dictionary,then it would return a dictionary type.

Python Dictionary methods

- Methods that are available with a dictionary are tabulated below. Some of them have already been used in the above examples.



Method	Description
clear()	Removes all items from the dictionary.
copy()	Returns a shallow copy of the dictionary.
fromkeys(seq[, v])	Returns a new dictionary with keys from `seq` and value equal to `v` (defaults to `None`).
get(key[,d])	Returns the value of the `key`. If the `key` does not exist, returns `d` (defaults to `None`).
items()	Return a new object of the dictionary's items in `(key, value)` format.
keys()	Returns a new object of the dictionary's keys.
pop(key[,d])	Removes the item with the `key` and returns its value or `d` if `key` is not found. If `d` is not provided and the `key` is not found, it raises `KeyError`.
popitem()	Removes and returns an arbitrary item `(key, value)`. Raises `KeyError` if the dictionary is empty.
setdefault(key[,d])	Returns the corresponding value if the `key` is in the dictionary. If not, inserts the `key` with a value of `d` and returns `d` (defaults to `None`).
update([other])	Updates the dictionary with the key/value pairs from `other`, overwriting existing keys.
values()	Returns a new object of the dictionary's values.

Python dictionary comprehension

- Dictionary comprehension is an elegant and concise way to create a new dictionary from an iterable in Python.
  - Dictionary comprehension consists of an expression pair `(key: value)` followed by a `for` statement inside curly braces `{}`.
  - Here is an example to make a dictionary with each item being a pair of a number and its square.

Python

```
#Example: Dictionary Comprehension
>>> squares = {x: x*x for x in range(6)}
>>> print(squares)  # ▶ {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Exercises

- Please click on the link provided below.
  - Built-In Data Structures:Dictionaries
  - In-Class Exercise