

Dictionary ??

dictionary (key) Tuple
tuple tuple list list
(index assignment) (slice assignment) append() extend() method

dictionary (key: value pair) set dictionary
dictionary { }

Key type:

- Numbers
- Booleans
- Strings
- Tuples

???

```
dictionary1 = {"keyA":valuea, "keyB":value2, "keyC":value3, "keyD":value4}

dictionary2 = {"keyA":["value1", "value2"], "keyB":["value3", "value4"]}
```

???

“ NOTE: Dictionary

```
file_counts = {"jpg":10, "txt":14, "csv":2, "py":23}
file_counts["txt"]
# Output: 14

# 
file_counts = {"jpg":10, "txt":14, "csv":2, "py":23, "txt":99}
file_counts["txt"]
# Output: 99
```

????

```
file_counts = {"jpg":10, "txt":14, "csv":2, "py":23}
"jpg" in file_counts
# Output: True
```

????: dictionary[key] = value

```
file_counts = {"jpg":10, "txt":14, "csv":2, "py":23}
file_counts["cfg"] = 8
print(file_counts)
# Output {'jpg': 10, 'txt': 14, 'csv': 2, 'py': 23, 'cfg': 8}
```

?????????: dictionary[key] = value

```
file_counts = {"jpg":10, "txt":14, "csv":2, "py":23}
file_counts["csv"] = 17
print(file_counts)
# Output {'jpg': 10, 'txt': 14, 'csv': 17, 'py': 23}
```

?????????

```
file_counts = {"jpg":10, "txt":14, "csv":2, "py":23, 'cfg':8}
del file_counts["cfg"]
print(file_counts)
# Output {'jpg': 10, 'txt': 14, 'csv': 2, 'py': 23}
```

Operations

- **len(dictionary)** - Returns the number of items in a dictionary.
- **for key, in dictionary** - Iterates over each key in a dictionary.
- **for key, value in dictionary.items()** - Iterates over each key,value pair in a dictionary.
- **if key in dictionary** - Checks whether a key is in a dictionary.
- **dictionary[key]** - Accesses a value using the associated key from a dictionary.
- **dictionary[key] = value** - Sets a value associated with a key.
- **del dictionary[key]** - Removes a value using the associated key from a dictionary.

???? for loop ????????? key ??

```
file_counts = {"jpg":10, "txt":14, "csv":2, "py":23}
for extension in file_counts:
    print(extension)
```

Output

```
jpg
txt
csv
py
```

Methods

- **dictionary.get(key, default)** - Returns the value corresponding to a key, or the default value if the specified key is not present.
- **dictionary.keys()** - Returns a sequence containing the keys in a dictionary.
- **dictionary.values()** - Returns a sequence containing the values in a dictionary.
- **dictionary[key].append(value)** - Appends a new value for an existing key.
- **dictionary.update(other_dictionary)** - Updates a dictionary with the items from another dictionary. Existing entries are updated; new entries are added.
- **dictionary.clear()** - Deletes all items from a dictionary.
- **dictionary.copy()** - Makes a copy of a dictionary.

.item()

`.items()` ?? dictionary ??????? key ? value?

```
file_counts = {"jpg":10, "txt":14, "csv":2, "py":23}
for ext, amount in file_counts.items():
    print("There are {} files with the .{} extension".format(amount, ext))
```

Output

```
There are 10 files with the .jpg extension
There are 14 files with the .txt extension
There are 2 files with the .csv extension
There are 23 files with the .py extension
```

```
# This function returns the total time, with minutes represented as
# decimals (example: 1 hour 30 minutes = 1.5), for all end user time
# spent accessing a server in a given day.
```

```
def sum_server_use_time(Server):

    # Initialize the variable as a float data type, which will be used
    # to hold the sum of the total hours and minutes of server usage by
    # end users in a day.
    total_use_time = 0.0

    # Iterate through the "Server" dictionary's key and value items
    # using a for loop.
    for key,value in Server.items():

        # For each end user key, add the associated time value to the
        # total sum of all end user use time.
        total_use_time += Server[key]

    # Round the return value and limit to 2 decimal places.
    return round(total_use_time, 2)

FileServer = {"EndUser1": 2.25, "EndUser2": 4.5, "EndUser3": 1, "EndUser4": 3.75, "EndUser5": 0.6, "EndUser6":
8}

print(sum_server_use_time(FileServer)) # Should print 20.1
```

```
# This function receives a dictionary, which contains common employee
# last names as keys, and a list of employee first names as values.
# The function generates a new list that contains each employees' full
# name (First_name Last_Name). For example, the key "Garcia" with the
# values ["Maria", "Hugo", "Lucia"] should be converted to a list
# that contains ["Maria Garcia", "Hugo Garcia", "Lucia Garcia"].
```

```
def list_full_names(employee_dictionary):

    # Initialize the "full_names" variable as a list data type using
    # empty [] square brackets.
    full_names = []

    # The outer for loop iterates through each "last_name" key and
    # associated "first_name" values, in the "employee_dictionary" items.
    for last_name, first_names in employee_dictionary.items():
```

```
# The inner for loop iterates over each "first_name" value in
# the list of "first_names" for one "last_name" key at a time.
for first_name in first_names:
```

```
    # Append the new "full_names" list with the "first_name" value
    # concatenated with a space " ", and the key "last_name".
    full_names.append(first_name+" "+last_name)
```

```
# Return the new "full_names" list once the outer for loop has
# completed all iterations.
return(full_names)
```

```
print(list_full_names({"Ali": ["Muhammad", "Amir", "Malik"], "Devi": ["Ram", "Amaira"], "Chen": ["Feng", "Li"]})))
# Should print ['Muhammad Ali', 'Amir Ali', 'Malik Ali', 'Ram Devi', 'Amaira Devi', 'Feng Chen', 'Li Chen']
```

.keys() .values()

`.keys()` , `.values()`

```
file_counts = {"jpg":10, "txt":14, "csv":2, "py":23}
file_counts.keys() # Return dict_keys(['jpg', 'txt', 'csv', 'py'])
file_counts.values() # Return dict_values([10, 14, 2, 23])
```

```
file_counts = {"jpg":10, "txt":14, "csv":2, "py":23}
for value in file_counts.values():
    print(value)
```

Output

```
10
14
2
23
```

- Use the **dictionary[key] = value** operation to associate a value with a key in a dictionary.
- Iterate over keys with multiple values from a dictionary, using nested **for** loops and an **if**-statement, and the **dictionary.items()** method.
- Use the **dictionary[key].append(value)** method to add the key, a string, and the key for each item in the dictionary.

```
def groups_per_user(group_dictionary):
    user_groups = {}

    # Go through group_dictionary
    for group, users in group_dictionary.items():
        # Now go through the users in the group
        for user in users:
            # Now add the group to the the list of
            if user in user_groups:
                user_groups[user].append(group)
            else:
                user_groups[user] = [group]

    # groups for this user, creating the entry
    # in the dictionary if necessary

    return(user_groups)

print(groups_per_user({"local": ["admin", "userA"],
    ["public": ["admin", "userB"],
    ["administrator": ["admin"] ]}))

# Should print {'admin': ['local', 'public', 'administrator'], 'userA': ['local'], 'userB': ['public']}
```

.update()

- **dictionary.update(other_dictionary)** - Updates a dictionary with the items from another dictionary. Existing entries are updated; new entries are added.

```
wardrobe = {'shirt': ['red', 'blue', 'white'], 'jeans': ['blue', 'black']}
new_items = {'jeans': ['white'], 'scarf': ['yellow'], 'socks': ['black', 'brown']}
wardrobe.update(new_items)

# wardrobe: {'shirt': ['red', 'blue', 'white'], 'jeans': ['white'], 'scarf': ['yellow'], 'socks': ['black', 'brown']}
```

.copy()

```
# The scores() function accepts a dictionary "game_scores" as a parameter.
def reset_scores(game_scores):

    # The .copy() dictionary method is used to create a new copy of the "game_scores".
```

```

new_game_scores = game_scores.copy()

# The for loop iterates over new_game_scores items, with the player as the key
# and the score as the value.
for player, score in new_game_scores.items():

    # The dictionary operation to assign a new value to a key is used
    # to reset the grade values to 0.
    new_game_scores[player] = 0

return new_game_scores

# The dictionary is defined.
game1_scores = {"Arshi": 3, "Catalina": 7, "Diego": 6}

# Call the "reset_scores" function with the "game1_scores" dictionary.
print(reset_scores(game1_scores))
# Should print {'Arshi': 0, 'Catalina': 0, 'Diego': 0}

```

Functions

sorted()

- `sorted(dict.items())` : ??? Dictionary ? key ???
- `sorted(, key=operator.itemgetter(0))` : `.itemgetter(0)` ? Dictionary ? key?????
- `sorted(, key=operator.itemgetter(1))` : `.itemgetter(1)` ? Dictionary ? value?????
- `sorted(, reverse=True)` : ????

```

fruit = {"oranges": 3, "apples": 5, "bananas": 7, "pears": 2}

sorted(fruit.items())
# [('apples', 5), ('bananas', 7), ('oranges', 3), ('pears', 2)]

import operator
sorted(fruit.items(), key=operator.itemgetter(0))
# [('apples', 5), ('bananas', 7), ('oranges', 3), ('pears', 2)]

sorted(fruit.items(), key=operator.itemgetter(1))
# [('pears', 2), ('oranges', 3), ('apples', 5), ('bananas', 7)]

```

```
sorted(fruit.items(), key = operator.itemgetter(1), reverse=True)
# [('bananas', 7), ('apples', 5), ('oranges', 3), ('pears', 2)]
```

Revision #40

Created 15 November 2024 14:06:57 by Admin

Updated 9 December 2024 14:00:10 by Admin