
# CS 315

# Programming Languages

## Section 3

## Ömer Oktay Gültekin

## 21901413

## 2022-2023 / Fall Semester

# Python:

## General Info about Python:

In Python, the associative array is called dictionary and it doesn't require to use any package of the language. While the value of the dictionary can be any type, the key must be immutable. This feature makes the language orthogonal for that part while not making it less reliable. Also, lines need not to be terminated which eases the writing programs. Variable naming convention is snake_case.

### 1) Initialize Associative Array

```
print("-------------------------- 1 --------------------------")
cs315_instructors = {
    "Section 1": "Altay Güvenir",
    "Section 2": "Aynur Dayanik",
    "Section 3": "Karani Kardas"
}

print("Initial CS 315 Instructors:", cs315_instructors)
print("-------------------------- 1 --------------------------")
print()
```

```
-------------------------- 1 --------------------------
Initial CS 315 Instructors: {'Section 1': 'Altay Güvenir', 'Section 2': 'Aynur Dayanik'
, 'Section 3': 'Karani Kardas'}
-------------------------- 1 --------------------------
```

The above code segment initializes associative array from given values and prints them in the console. The writability of initializing associative array is good for the following reasons:

1) Dictionary data type not needs to be explicitly stated
2) Type of the keys and values do not need to be explicitly stated
3) Print statement automatically prints newline (although one can override this using end option of the print statement)
4) Print statement of the language is short (i.e Java has System.out.println statement)
5) The Printable representation of the dictionary is not needing an additional formatting (otherwise the programmer should define additional methods)

The readability of initializing associative array is good for the following reasons:

1) Dictionary notation is intuitional that is in a commonly used format (Same as JSON format).
2) The name of the print statement is self descriptive.
3) The printable representation of the dictionary is in a nice format.

## 2) Get the Value for a Given Key

```
print("-------------------------- 2 --------------------------")
print("Instructor of Section 3 is {}".format(cs315_instructors["Section 3"]))
print("-------------------------- 2 --------------------------")
print()
```

```
-------------------------- 2 --------------------------
Instructor of Section 3 is Karani Kardas
-------------------------- 2 --------------------------
```

The above code segment prints the value of a given key “Section 3”. The writability of getting the value for a given key of the associative array is good for the following reasons:

1) Dictionary Value can be reached by the bracket notation (intuitive approach)
2) One can use the string formatting by simply inserting {} in the place that needs to be formatted and giving the value by using the format() method. The format method is easy to remember, and nothing needs to be inserted between {} to indicate the data type unlike some other programming languages.

The readability of getting the value for a given key of the associative array is good for the following reasons:

1) Dictionary Value can be reached by the bracket notation (intuitive approach).
2) Format method self descriptive.
3) The programmer that does not know Python can easily understand this operation.

## 3) Add a New Element

```
print("-------------------------- 3 --------------------------")
cs315_instructors["Section 4"] = "Not Available"
print("CS 315 Instructors After Addition: {}".format(cs315_instructors))
print("-------------------------- 3 --------------------------")
print()
```

```
-------------------------- 3 --------------------------
CS 315 Instructors After Addition: {'Section 1': 'Altay Güvenir', 'Section 2': 'Aynur D
ayanik', 'Section 3': 'Karani Kardas', 'Section 4': 'Not Available'}
-------------------------- 3 --------------------------
```

The above code segment adds a new element and prints the added dictionary. The writability of adding a new element to the associative array is good for the following reasons:

1) The syntax of adding a new element is the same as changing the value of existing element, no need to memorize new ways to do it.
2) Adding a new element is short.

Positive aspects in terms of the readability:

1) Short syntax
2) The programmer understands that now there is a key with the value of “Not Available” in the dictionary

Negative aspects in terms of the readability:

1) The previous code segments need to be read to understand whether the element is updated or newly added.

## 4) Remove an Element

```
print("-------------------------- 4 --------------------------")
# print(del cs315_instructors["Section 4"])
del cs315_instructors["Section 4"]
print("CS 315 Instructors After Deletion:", cs315_instructors)
print("-------------------------- 4 --------------------------")
print()
```

```
-------------------------- 4 --------------------------
CS 315 Instructors After Deletion: {'Section 1': 'Altay Güvenir', 'Section 2': 'Aynur D
ayanik', 'Section 3': 'Karani Kardas'}
-------------------------- 4 --------------------------
```

The above code segment removes a given key and prints the resulting dictionary.

Positive aspects in terms of the writability:

1) Intuitive approach for deleting key (del keyword)
2) Short syntax

Negative aspects in terms of the writability:

1) del keyword is different from python syntax (Python methods generally use parantheses like print() or len())

Positive aspects in terms of the readability:

1) Easy to understand
2) Intuitive

Negative aspects in terms of the readability:

1) Delete statement do not return what value was deleted; therefore, programmer needs to see the previous sections to see the deleted value

## 5) Modify the Value of an Existing Element

```
print("-------------------------- 5 --------------------------")
cs315_instructors["Section 1"] = "Halil Altay Güvenir"
print("CS 315 Instructors After Modification:", cs315_instructors)
print("-------------------------- 5 --------------------------")
print()
```

```
------------------------- 5 -------------------------
CS 315 Instructors After Modification: {'Section 1': 'Halil Altay Güvenir', 'Section 2'
: 'Aynur Dayanik', 'Section 3': 'Karani Kardas'}
------------------------- 5 -------------------------
```

The above code segment modifies the value using the given key and prints the resulting dictionary. Positive aspects in terms of the writability:

1) No need to memorize new syntax, it is the same as adding new element

Positive aspects in terms of the readability:

1) Intuitive approach, one with no coding experience in Python can understand

Negative aspects in terms of the readability:

1) Programmer needs to see the previous sections to see if this operation is adding new element or modifying existing element

### 6) Search for the Existence of a Key

```
print("------------------------- 6 -------------------------")
print("Is Section 2 in CS 315 Sections:", "Section 2"
      in cs315_instructors.keys())
print("Is Section 1 in Cs 315 Sections:", "Section 1"
      in cs315_instructors)  # Also Valid
print("------------------------- 6 -------------------------")
```

```
------------------------- 6 -------------------------
Is Section 2 in CS 315 Sections: True
Is Section 1 in Cs 315 Sections: True
------------------------- 6 -------------------------
```

The above code segment searches the existence of given key using 2 ways that returns Python list data type. Positive aspects in terms of the writability:

1) 2 different ways to implement
2) First way is very intuitive (in and keys())

Negative aspects in terms of the writability:

1) Second choice is confusing (cs315_instructors can also return (key, value) pair, one cannot know by looking at the code)

Positive aspects in terms of the readability:

1) Intuitive

Negative aspects in terms of the readability:

1) Same problem with writability

### 7) Search for the Existence of a Value

```
print("------------------------- 7 -------------------------")
print("Is Karani Kardas one of instructors:", "Karani Kardas"
      in cs315_instructors.values())
print("------------------------- 7 -------------------------")
```

```
-------------------------- 7 --------------------------
Is Karani Kardas one of instructors: True
-------------------------- 7 --------------------------
```

The above code segment searches the existence of a given value in values() list.
Positive aspects in terms of the writability and readability:

1) Intuitive and short syntax

## 8) Loop through an Associative Array, apply a Function, called foo, which Simply Prints the Key-Value Pair

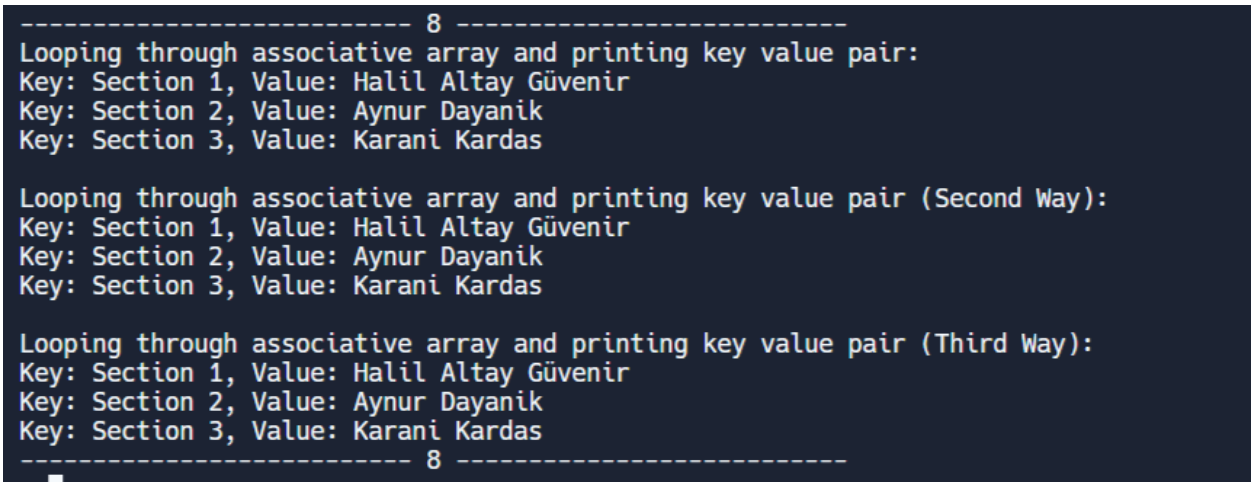
```
print("-------------------------- 8 --------------------------")
def foo(key, value):
    print(f"Key: {key}, Value: {value}")


print("Looping through associative array and printing key value pair:")
for item in cs315_instructors.items():
    foo(*item)

print()
print("Looping through associative array and printing key value pair (Second
Way):")
for item in cs315_instructors.items():
    foo(item[0], item[1])

print()
print("Looping through associative array and printing key value pair (Third
Way):")
for key, value in cs315_instructors.items():
    foo(key, value)

# Not work as expected, cs315_instructors returns only keys
# for key, value in cs315_instructors:
#     foo(key, value)
print("-------------------------- 8 --------------------------")
```




The above code segment defines a function using def keyword and uses it to print key value pairs in 3 different ways. The code uses packing feature of the Python and depack it by using “*” operator. Positive aspects in terms of the writability:

1) Different ways to iterate over dictionary
2) Short syntax using depacking operator
3) Intuitive .items() list which returns tuple data type
4) Easy method definition
5) Uses indentation to indicate part of the program
6) Third way to formatting string (via f-strings)

Negative aspects in terms of the writability:

1) Most intuitive way does not return elements together (cs315_instructor without using .items())
2) No special for each loop like in other languages

Positive aspects in terms of the readability:

1) Intuitive syntax (method def, indentation usage, .item(), for loop)

Negative aspects in terms of the readability:

1) Depacking operator may be hard to understand withour knowing it. Also, it uses same operator which is used to referencing in C-based languages

### Learning Strategy for Python:

Since I know Python, I did not need to look at any source on the internet. Also, I did not have any personal communication for any language.

## Dart:

### General Info about Dart:

In Dart, the associative array is called Map which is a dynamic collection, and it doesn't require to use any external package of the language. Keys and values can be of any type which means dart is very orthogonal. Also, lines need to be terminated. Variable naming convention is camelCase.

## 1) Initialize Associative Array

```
void main() {
  print("--------------------------- 1 ---------------------------");
  // Map<String, String> ListFinalAllInfos = {'stackoverflow': 'one',
'google': 'two'};
  var cs315Instructors = {
    "Section 1": "Altay Güvenir",
    "Section 2": "Aynur Dayanik",
    "Section 3": "Karani Kardas"
  };
  print("Initial CS 315 Instructors: ${cs315Instructors}");
  print("--------------------------- 1 ---------------------------");
```

```
--------------------------- 1 ---------------------------
Initial CS 315 Instructors: {Section 1: Altay Güvenir, Section 2: Aynur Dayanik, Se
ction 3: Karani Kardas}
--------------------------- 1 ---------------------------
```

The above code segment initializes associative array from given values and prints them in the console. Positive aspects in terms of the writability:

1) Type inferencing using var keyword. (Normal data type is shown in comments).
2) Intuitive map creation.
3) Intuitive Print statement.

Positive aspects in terms of the readability:

1) Intuitive code

Negative aspects in terms of the readability:

1) Type inferencing may be hinder readability

## 2) Get the Value for a Given Key

```
  print("--------------------------- 2 ---------------------------");
  print(cs315Instructors["Section 3"]);
  print("--------------------------- 2 ---------------------------");
```

```
--------------------------- 2 ---------------------------
Karani Kardas
--------------------------- 2 ---------------------------
```

The above code segment prints the value using the given key. Positive aspects in terms of the writability:

1) Intuitive
2) Short syntax

Positive aspects in terms of the readability:

1) Same aspects with writability

### 3) Add a New Element

```
print("--------------------------- 3 ---------------------------");
  cs315Instructors["Section 4"] = "Not Available";
  print(cs315Instructors);
  print("--------------------------- 3 ---------------------------");
```

```
--------------------------- 3 ---------------------------
{Section 1: Altay Güvenir, Section 2: Aynur Dayanik, Section 3: Karani Kardas, Sect
ion 4: Not Available}
--------------------------- 3 ---------------------------
```

The above code segment adds a new element and prints the added map. The writability of adding a new element to the associative array is good for the following reasons:

3) The syntax of adding a new element is the same as changing the value of existing element, no need to memorize new ways to do it.
4) Adding a new element is short.

Positive aspects in terms of the readability:

3) Short syntax
4) The programmer understands that now there is a key with the value of “Not Available” in the dictionary

Negative aspects in terms of the readability:

2) The previous code segments need to be read to understand whether the element is updated or newly added.

### 4) Remove an Element

```
print("--------------------------- 4 ---------------------------");
  print(cs315Instructors.remove("Section 4"));
  print(cs315Instructors);
  print("--------------------------- 4 ---------------------------");
```

```
------------------------- 4 -------------------------
Not Available
{Section 1: Altay Güvenir, Section 2: Aynur Dayanik, Section 3: Karani Kardas}
------------------------- 4 -------------------------
```

The above code segment removes a given key and prints the resulting map. Positive aspects in terms of the writability:

1) Intuitive and short syntax (remove entry with remove() method)

Positive aspects in terms of the readability:

1) Intuitive and short syntax
2) Returns removed value (Which Python does not)

### 5) Modify the Value of an Existing Element

```
print("------------------------- 5 -------------------------");
  cs315Instructors["Section 1"] = "Halil Altay Güvenir";
  print(cs315Instructors);
  print("------------------------- 5 -------------------------");
```

```
------------------------- 5 -------------------------
{Section 1: Halil Altay Güvenir, Section 2: Aynur Dayanik, Section 3: Karani Kardas
}
------------------------- 5 -------------------------
```

The above code segment modifies the value using the given key and prints the resulting map. Positive aspects in terms of the writability:

1) No need to memorize new syntax, it is the same as adding new element

Positive aspects in terms of the readability:

1) Intuitive approach, one with no coding experience in Dart can understand

Negative aspects in terms of the readability:

1) Programmer needs to see the previous sections to see if this operation is adding new element or modifying existing element

### 6) Search for the Existence of a Key

```
  print("------------------------- 6 -------------------------");
  print(cs315Instructors.containsKey("Section 2"));
  print(cs315Instructors.containsKey("Section 4"));
  print("------------------------- 6 -------------------------");
```

```
-------------------------- 6 --------------------------
true
false
-------------------------- 6 --------------------------
```

The above code segment searches for the existence of  a given key. Positive aspects in terms of the writability:

1) Intuitive and short

Positive aspects in terms of the readability:

1) Same as writability advantages

### 7) Search for the Existence of a Value

```
  print("-------------------------- 7 --------------------------");
  print(cs315Instructors.containsValue("Karani Kardas"));
  print(cs315Instructors.containsValue("Karani"));
  print("-------------------------- 7 --------------------------");
```

```
-------------------------- 7 --------------------------
true
false
-------------------------- 7 --------------------------
```

The above code segment searches for the existence of  a given value. Positive aspects in terms of the writability:

1) Intuitive and short

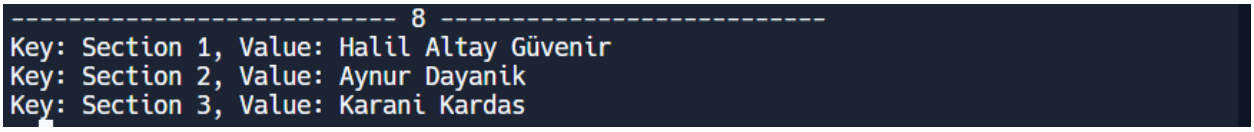
Positive aspects in terms of the readability:

1) Same as writability advantages

### 8) Loop through an Associative Array, apply a Function, called foo, which Simply Prints the Key-Value Pair

```
foo(var key, var value) {
  print("Key: ${key}, Value: ${value}");
}

print("-------------------------- 8 --------------------------");
cs315Instructors.forEach((k, v) => foo(k, v));
print("-------------------------- 8 --------------------------");
```




The above code segment defines a function using no keyword and uses it to print key value pairs by using type inference. The code uses special forEach method of the dart and uses arrow functions which is allowed if the inner function body is 1 line. Positive aspects in terms of the writability:

1) Type inference
2) Easy formatting (${} anything in between works)
3) forEach method
4) Arrow function
5) Intuitive and short code

Positive aspects in terms of the readability:

1) Intuitive, short understandable code (like forEach())

Negative aspects in terms of the readability:

1) No method definition keyword
2) Arrow function may be hard to understand

### Learning Strategy for Dart:

I learned a little about Dart before since I was interested in App Dev using Flutter. For what I didn't remember I mostly use the documentation of the Dart. Other than that I specifically search the problem I have encountered using following links:
https://www.javatpoint.com/dart-methods
https://zetcode.com/dart/map/
https://www.educative.io/answers/what-is-the-mapremove-method-in-dart
https://stackoverflow.com/questions/64484985/check-if-the-map-has-a-specific-string-in-its-keys-then-give-me-its-value-th
https://www.tutorialspoint.com/dart_programming/dart_programming_map_function_foreach.htm

## Javascript:

### General Info about Javascript:

In Javascript, the associative array is an Object which is an implementation of OOP paradigm in JS, and it doesn't require to use any external package of the language. Keys must be strings, numbers or sembols but values can be of any type which means JS is very orthogonal. Also, lines can be terminated or not. Variable naming convention is camelCase.

Note: I omit the html tags for simplicity of the report (it is already very long sorry about that ☹)

## 1) Initialize Associative Array

```
    console.log("-------------------------- 1 ----------------------
----")
    const cs315Instructors = {
      "Section 1": "Altay Güvenir",
      "Section 2": "Aynur Dayanik",
      "Section 3": "Karani Kardas",
      Section: "Different way to create entry"
    }
    console.log(cs315Instructors)
    console.log("-------------------------- 1 ----------------------
----")
```

```
-------------------------- 1 --------------------------
Object {Section 1: "Halil Altay Güvenir", Section 2: "Aynur Dayanik", Section
3: "Karani Kardas", Section: "Different way to create entry"}
-------------------------- 1 --------------------------
```

The above code segment initializes the Object and prints the resulting Object. When key is one word, The quotes are not necessary. Positive aspects in terms of the writability:

1) Intuitive and short
2) No need of quotes in certain condition

Positive aspects in terms of the readability:

1) Intuitive and short

Negative aspects in terms of the readability:

1) Key without quotes can be confusing

## 2) Get the Value for a Given Key

```
    console.log("-------------------------- 2 ----------------------
----")
    console.log(cs315Instructors["Section 3"])
```

```
    console.log("--------------------------- 2 ------------------------
----")
```

```
--------------------------- 2 ---------------------------
Karani Kardas
--------------------------- 2 ---------------------------
```

The above code segment prints the value using the given key. Positive aspects in terms of the writability:

1) Intuitive
2) Short syntax

Positive aspects in terms of the readability:

1) Same aspects with writability

### 3) Add a New Element

```
    console.log("--------------------------- 3 ------------------------
----")
    cs315Instructors["Section 4"] = "Not Available"
    console.log(cs315Instructors)
    console.log("--------------------------- 3 ------------------------
----")
```

```
--------------------------- 3 ---------------------------
Object {Section 1: "Halil Altay Güvenir", Section 2: "Aynur Dayanik", Section
3: "Karani Kardas", Section: "Different way to create entry"}
--------------------------- 3 ---------------------------
```

The above code segment adds a new element and prints the added Object. The writability of adding a new element to the associative array is good for the following reasons:

1) The syntax of adding a new element is the same as changing the value of existing element, no need to memorize new ways to do it.
2) Adding a new element is short.

Positive aspects in terms of the readability:

1) Short syntax

2) The programmer understands that now there is a key with the value of “Not Available” in the dictionary

Negative aspects in terms of the readability:

1) The previous code segments need to be read to understand whether the element is updated or newly added.

## 4) Remove an Element

```
    console.log("--------------------------- 4 ------------------------
----")
    delete cs315Instructors["Section 4"]
    console.log(cs315Instructors)
    console.log("--------------------------- 4 ------------------------
----")
```

```
--------------------------- 4 ---------------------------
Object {Section 1: "Halil Altay Güvenir", Section 2: "Aynur Dayanik", Section
3: "Karani Kardas", Section: "Different way to create entry"}
--------------------------- 4 ---------------------------
```

The above code segment removes a given key and prints the resulting object.

Positive aspects in terms of the writability:

1) Intuitive approach for deleting key (delete keyword)
2) Short syntax

Negative aspects in terms of the writability:

1) delete keyword is different from JS syntax (JS methods generally use parantheses like print())

Positive aspects in terms of the readability:

1) Easy to understand
2) Intuitive

Negative aspects in terms of the readability:

1) Delete statement do not return what value was deleted; therefore, programmer needs to see the previous sections to see the deleted value

## 5) Modify the Value of an Existing Element

```
    console.log("-------------------------- 5 ----------------------
----")
    cs315Instructors["Section 1"] = "Halil Altay Güvenir"
    console.log(cs315Instructors)
    console.log("-------------------------- 5 ----------------------
----")
```

```
-------------------------- 5 --------------------------
Object {Section 1: "Halil Altay Güvenir", Section 2: "Aynur Dayanik", Section
3: "Karani Kardas", Section: "Different way to create entry"}
-------------------------- 5 --------------------------
```

The above code segment modifies the value using the given key and prints the resulting Object. Positive aspects in terms of the writability:

1) No need to memorize new syntax, it is the same as adding new element

Positive aspects in terms of the readability:

1) Intuitive approach, one with no coding experience in JS can understand

Negative aspects in terms of the readability:

1) Programmer needs to see the previous sections to see if this operation is adding new element or modifying existing element

## 6) Search for the Existence of a Key

```
    console.log("-------------------------- 6 ----------------------
----")
    console.log(cs315Instructors.hasOwnProperty("Section 2"))
    console.log("-------------------------- 6 ----------------------
----")
```

```
-------------------------- 6 --------------------------
true
-------------------------- 6 --------------------------
```

The above code segment searches for the existence of the given key. Positive aspects in terms of the writability:

1) Direct method to do
2) Short syntax

Negative aspects in terms of the writability:

1) hasOwnProperty hard to guess since it is OOP method

Positive aspects in terms of the readability:

1) Same with writability

Negative aspects in terms of the readability:

1) Same with writability

## 7) Search for the Existence of a Value

```
    console.log("-------------------------- 7 ------------------------
----")
    console.log(Object.values(cs315Instructors).includes("Karani
Kardas"))
    console.log("-------------------------- 7 ------------------------
----")
```

```
-------------------------- 7 --------------------------
true
-------------------------- 7 --------------------------
```

The above code segment searches the existence of a value by using given key. Positive aspects in terms of the writability:

1) Direct method
2) Semanticly meaningful method

Positive aspects in terms of the readability:

1) Intuitive and short syntax

8) **Loop through an Associative Array, apply a Function, called foo, which Simply Prints the Key-Value Pair**

```
function foo(k, v) {
      console.log(`Key: ${k}, Value: ${v}`)
    }


.
.
.
    console.log("--------------------------- 8 ------------------------
----")
    Object.entries(cs315Instructors).forEach((k, v) => foo(k,v))
    console.log("--------------------------- 8 ------------------------
----")
```

```
--------------------------- 8 ---------------------------
Key: Section 1,Halil Altay Güvenir, Value: 0
Key: Section 2,Aynur Dayanik, Value: 1
Key: Section 3,Karani Kardas, Value: 2
Key: Section,Different way to create entry, Value: 3
--------------------------- 8 ---------------------------
```

The above code segment defines a function using function keyword and uses it to print key value pairs by using no type decleration (implicit type inference). The code uses special forEach method of the JS and uses arrow functions which is allowed if the inner function body is 1 line. Positive aspects in terms of the writability:

1) Type inference
2) Easy formatting (${} anything in between works but use “`” for printing)
3) forEach method
4) Arrow function
5) Intuitive and short code

Positive aspects in terms of the readability:

1) Intuitive, short understandable code (like forEach(), function keyword)

Negative aspects in terms of the readability:

1) Arrow function may be hard to understand

**Learning Strategy for Javascript:**

I learned a little about JS before for CS 319 project. For what I didn't know, I mostly use the forums for the JS. Other than that, I specifically search the problem I have encountered using following links:
https://stackoverflow.com/questions/921789/how-to-loop-through-a-plain-javascript-object-with-the-objects-as-members
https://www.w3schools.com/js/js_function_definition.asp
https://stackoverflow.com/questions/35948669/how-to-check-if-a-value-exists-in-an-object-using-javascript
https://stackoverflow.com/questions/346021/how-do-i-remove-objects-from-a-javascript-associative-array
https://pietschsoft.com/post/2015/09/05/javascript-basics-how-to-create-a-dictionary-with-keyvalue-pairs
https://stackoverflow.com/questions/40291766/naming-convention-for-const-object-keys-in-es6

# Lua:

## General Info about Lua:

In Lua, the associative array is a Table, and it doesn't require to use any external package of the language. Keys should be between brackets if there is more than 1 word otherwise no bracket and quotes are used. Also, lines are not terminated. There is no variable naming convention.

### 1) Initialize Associative Array

```
function printDict(dict)
  print()
  for k, v in pairs(dict) do
    foo(k, v)
  end
  print()
end

.
.
.

print("--------------------------- 1 ----------------------------");
cs315_instructors = {
    ["Section 1"] = "Altay Güvenir",
    ["Section 2"] = "Aynur Dayanik",
```

```
    ["Section 3"] = "Karani Kardas",
    Section       = "Different way to create key"
}

print(cs315_instructors)
printDict(cs315_instructors)
print("-------------------------- 1 --------------------------");
```

```
-------------------------- 1 --------------------------
table: 0x125c8d0

Section 1   Altay Güvenir
Section 2   Aynur Dayanik
Section 3   Karani Kardas
Section Different way to create key

-------------------------- 1 --------------------------
```

The above code segment initializes a table with given values. Positive aspects in terms of the writability:

1) Intuitive print method and table definition

Negative aspects in terms of the writability:

1) Normal print method does not print meaningful output for Tables
2) Brackets are odd

Positive aspects in terms of the readability:

1) Intuitive and short code

Negative aspects in terms of the readability:

2) Same problems with writability

### 2) Get the Value for a Given Key

```
print("-------------------------- 2 --------------------------");
print(cs315_instructors["Section 1"])
print("-------------------------- 2 --------------------------");
```

```
-------------------------- 2 --------------------------
Altay Güvenir
-------------------------- 2 --------------------------
```

The above code segment prints the value using the given key. Positive aspects in terms of the writability:

1) Intuitive
2) Short syntax

Positive aspects in terms of the readability:

1) Same aspects with writability

### 3) Add a New Element

```
print("--------------------------- 3 ---------------------------");
cs315_instructors["Section 4"] = "Not Available"
printDict(cs315_instructors)
print("--------------------------- 3 ---------------------------");
```

```
--------------------------- 3 ---------------------------

Section 3   Karani Kardas
Section Different way to create key
Section 2   Aynur Dayanik
Section 1   Altay Güvenir
Section 4   Not Available

--------------------------- 3 ---------------------------
```

The above code segment adds a new element and prints the added Table. The writability of adding a new element to the associative array is good for the following reasons:

1) The syntax of adding a new element is the same as changing the value of existing element, no need to memorize new ways to do it.
2) Adding a new element is short.

Positive aspects in terms of the readability:

3) Short syntax

### 4) Remove an Element

```
print("--------------------------- 4 ---------------------------");
cs315_instructors["Section 4"] = nil
printDict(cs315_instructors)
print("--------------------------- 4 ---------------------------");
```

```
--------------------------- 4 ---------------------------

Section 3   Karani Kardas
Section Different way to create key
Section 2   Aynur Dayanik
Section 1   Altay Güvenir

--------------------------- 4 ---------------------------
```

The above code segment removes a given key and prints the resulting dictionary.
Positive aspects in terms of the writability:

1) Intuitive and short code

Negative aspects in terms of the writability:

1) Assigning nil is not a common practice in programming

Positive aspects in terms of the readability:

1) Intuitive and short code

### 5) Modify the Value of an Existing Element

```
print("--------------------------- 5 ---------------------------");
cs315_instructors["Section 1"] = "Halil Altay Güvenir"
printDict(cs315_instructors)
print("--------------------------- 5 ---------------------------");
```

```
--------------------------- 5 ---------------------------

Section 3   Karani Kardas
Section Different way to create key
Section 2   Aynur Dayanik
Section 1   Halil Altay Güvenir

--------------------------- 5 ---------------------------
```

The above code segment modifies the value using the given key and prints the resulting Table. Positive aspects in terms of the writability:

2) No need to memorize new syntax, it is the same as adding new element

Positive aspects in terms of the readability:

2) Intuitive approach, one with no coding experience in Lua can understand

Negative aspects in terms of the readability:

2) Programmer needs to see the previous sections to see if this operation is adding new element or modifying existing element

### 6) Search for the Existence of a Key

```
print("--------------------------- 6 ---------------------------");
--[[
~= is negation operator in lua
--]]
```

```
print(cs315_instructors["Section 1"] ~= nil)
print("-------------------------- 6 --------------------------");
```

```
-------------------------- 6 --------------------------
true
-------------------------- 6 --------------------------
```

The above code segment searches for the existence of a given key. Positive aspects in terms of the writability:

1) Short Code

Negative aspects in terms of the writability:

1) Not using != operator is not very intuitive

Positive aspects in terms of the readability:

1) The code is very understandable

Negative aspects in terms of the readability:

1) Nil is not used in many languages (Null is more common) and ~= is not intuitive

### 7) Search for the Existence of a Value

```
function hasValue(dict, val)
    for k, v in pairs(dict) do
      if v == val then
            return true
      end
    end
    return false
end
print("-------------------------- 7 --------------------------");
print(hasValue(cs315_instructors, "Karani Kardas"));
print("-------------------------- 7 --------------------------");
```

```
-------------------------- 7 --------------------------
true
-------------------------- 7 --------------------------
```

The above code segment searches for existence of the value by using given key

Negative aspects in terms of the writability:

1) No built-in function for searching

Aspects in terms of the readability:

1) Above solution is intuitive and understandable but not standart of the language

### 8) Loop through an Associative Array, apply a Function, called foo, which Simply Prints the Key-Value Pair

```
function foo(key, value)
    print(key, value)
end
print("-------------------------- 8 --------------------------");
print()
for k, v in pairs(cs315_instructors) do
    foo(k, v)
end
print()
print("--------------------------  8 --------------------------");
```

```
-------------------------- 8 --------------------------

Section 3   Karani Kardas
Section Different way to create key
Section 2   Aynur Dayanik
Section 1   Halil Altay Güvenir

--------------------------  8 --------------------------
```

The above code segment defines a function using function keyword and uses it to print key value pairs by using no type declaration (implicit type inference). There is no special forEach method of the Lua. Positive aspects in terms of the writability:

1) Type inference
2) Pairs method to reach key value pair
3) Intuitive and short code

Negative aspects in terms of the writability:

1) The function definition does not use { or : which is not intuitive for me

Positive aspects in terms of the readability:

1) Intuitive, short understandable code (like for method and function keyword)

## Learning Strategy for Lua:

The documentation for Lua was not very helpful. Therefore,I mostly search the problem I have encountered using following links:
https://www.quora.com/What-coding-language-is-Lua-based-from
https://stackoverflow.com/questions/51051431/in-lua-is-there-a-difference-between-local-functions-declared-with-and-without
https://stackoverflow.com/questions/2282444/how-to-check-if-a-table-contains-an-element-in-lua
https://snippets.bentasker.co.uk/page-1705231140-Check-if-table-has-element-LUA.html

https://stackoverflow.com/questions/513239/how-to-quickly-initialise-an-associative-table-in-lua
https://www.reddit.com/r/lua/comments/1shtsg/is_camelcase_or_snake_case_more_common_in_lua/

# PHP:

## General Info about PHP:

In PHP, the associative array is normal array, and it doesn't require to use any external package of the language. The => operator is used for key value pairs. Variables are declared with $ sign. Also, lines are terminated. The naming convention is camelCase.

```
<?php
```

Program starting point

## 1) Initialize Associative Array

```
print "--------------------------- 1 ---------------------------" .
"\n";
// as of PHP 5.4
$cs315Instructors = [
    "Section 1" => "Altay Güvenir",
    "Section 2" => "Aynur Dayanik",
    "Section 3" => "Karani Kardas"
];


// $search_array = array('first' => 1, 'second' => 4);
print_r($cs315Instructors);
print "--------------------------- 1 ---------------------------" .
"\n";
```

```
------------------------- 1 -------------------------
Array
(
    [Section 1] => Altay Güvenir
    [Section 2] => Aynur Dayanik
    [Section 3] => Karani Kardas
)
------------------------- 1 -------------------------
```

The above code segment initializes associative array.Positive aspects in terms of the writability:

1) Intuitive and short code

Negative aspects in terms of the writability:

1) Different sign for value binding “=>”
2) There is 3 print function (print, echo and print_r, the last of which is specifically printing associative array. (I believe it is print recursively))

Positive aspects in terms of the readability:

1) Intuitive and short code

Negative aspects in terms of the readability:

1) Some differences from conventions

## 2) Get the Value for a Given Key

```
print "------------------------- 2 -------------------------" .
"\n";
print "Instructor of " . "Section 3 is ". $cs315Instructors["Section
3"] . "\n";
print "------------------------- 2 -------------------------" .
"\n";
```

```
------------------------- 2 -------------------------
Instructor of Section 3 is Karani Kardas
------------------------- 2 -------------------------
```

The above code segment prints the value of given key. Positive aspects in terms of the writability:

1) “.” Notation for concaneting strings is easy

2) Common access notation

Positive aspects in terms of the readability:

1) Intuitive code

Negative aspects in terms of the readability:

1) “.” Notation is different from other languages

### 3) Add a New Element

```
print "--------------------------- 3 ---------------------------" .
"\n";
$cs315Instructors["Section 4"] = "Not Available";
print_r($cs315Instructors);
print "--------------------------- 3 ---------------------------" .
"\n";
```

```
--------------------------- 3 ---------------------------
Array
(
    [Section 1] => Altay Güvenir
    [Section 2] => Aynur Dayanik
    [Section 3] => Karani Kardas
    [Section 4] => Not Available
)
--------------------------- 3 ---------------------------
```

The above code segment addes given element to the associative array. Positive aspects in terms of the writability:

1) Intuitive and short code

Positive aspects in terms of the readability:

1) Intuitive and short code

### 4) Remove an Element

```
print "--------------------------- 4 ---------------------------" .
"\n";
```

```
unset($cs315Instructors["Section 4"]);
print_r($cs315Instructors);
print "-------------------------- 4 --------------------------" .
"\n";
```

```
-------------------------- 4 --------------------------
Array
(
    [Section 1] => Altay Güvenir
    [Section 2] => Aynur Dayanik
    [Section 3] => Karani Kardas
)
-------------------------- 4 --------------------------
```

The above code segment removes a given element and prints the resulting dictionary.
Positive aspects in terms of the writability:

1) One method for given task
2) Short code

Negative aspects in terms of the writability:

1) Unset() is hard to remember

Positive aspects in terms of the readability:

1) Short code

Negative aspects in terms of the readability:

1) Unset() is not common

### 5) Modify the Value of an Existing Element

```
print "-------------------------- 5 --------------------------" .
"\n";
$cs315Instructors["Section 1"] = "Halil Altay Güvenir";
print_r($cs315Instructors);
print "-------------------------- 5 --------------------------" .
"\n";
```

```
-------------------------- 5 --------------------------
Array
(
    [Section 1] => Halil Altay Güvenir
    [Section 2] => Aynur Dayanik
    [Section 3] => Karani Kardas
)
-------------------------- 5 --------------------------
```

The above code segment modifies a given element and prints the resulting dictionary. Positive aspects in terms of the writability:

1) Intuitive and short

Positive aspects in terms of the readability:

1) Intuitive and short

Negative aspects in terms of the readability:

1) Not clear whether it modifies the existence key or the new element

## 6) Search for the Existence of a Key

```
print "-------------------------- 6 --------------------------" .
"\n";
print array_key_exists("Section", $cs315Instructors) . "\n";
print array_key_exists("Section 2", $cs315Instructors) . "\n";
print isset($cs315Instructors["Section 2"]) . "\n";
print "-------------------------- 6 --------------------------" .
"\n";
```

```
-------------------------- 6 --------------------------

1
1
-------------------------- 6 --------------------------
```

The above code segment searches an existence of a given key. Positive aspects in terms of the writability:

1) One method for given task
2) 2 different ways to do

Negative aspects in terms of the writability:

1) Not common methods with other languages

Positive aspects in terms of the readability:

1) Short code, one method usage

Negative aspects in terms of the readability:

1) When the key is not found, there is no false value printed
2) Not common methods

### 7) Search for the Existence of a Value

```
print "-------------------------- 7 --------------------------" .
"\n";
print (array_search("Karani Kardas", $cs315Instructors) == True) .
"\n";
print "-------------------------- 7 --------------------------" .
"\n";
```

```
-------------------------- 7 --------------------------
1
-------------------------- 7 --------------------------
```

The above code segment searchs for an existence of given value. Positive aspects in terms of the writability:

1) Intuitive and short

Negative aspects in terms of the writability:

1) Array search needs to be remembered

Positive aspects in terms of the readability:

1) Intuitive and short code

### 8) Loop through an Associative Array, apply a Function, called foo, which Simply Prints the Key-Value Pair

```
function foo($k, $v) {
  print "Key: $k Value: $v \n";
}

.
```

```
.
.


print "-------------------------- 8 --------------------------" .
"\n";
foreach($cs315Instructors as $k => $v) {
  foo($k, $v);
}
print "-------------------------- 8 --------------------------" .
"\n";
```

```
-------------------------- 8 --------------------------
Key: Section 1 Value: Halil Altay Güvenir
Key: Section 2 Value: Aynur Dayanik
Key: Section 3 Value: Karani Kardas
-------------------------- 8 --------------------------
```

The above code segment defines a function using function keyword and uses it to print key value pairs by using no type declaration (implicit type inference). There is special forEach method of the PHP. Positive aspects in terms of the writability:

1) Type inference
2) Easy access to key value pair
3) Intuitive and short code
4) Easy string formatting since variables are accessed via $ sign

Positive aspects in terms of the readability:

1) Intuitive, short understandable code (like foreach() and function keyword)

## Learning Strategy for PHP:

I mostly uses online forums to search for the problems, but I rarely used the official doc of the PHP since it is not that bad. I specifically search the problems I have encountered using following links:

https://www.w3schools.com/php/php_functions.asp
https://www.quora.com/How-do-you-check-if-a-value-exists-in-an-associative-array-in-PHP
https://www.php.net/manual/en/function.array-key-exists.php
https://www.geeksforgeeks.org/removing-array-element-and-re-indexing-in-php/
https://www.techiedelight.com/print-array-php/
https://www.educative.io/answers/what-are-the-naming-conventions-of-variables-in-php
https://www.w3schools.com/php/php_echo_print.asp

https://stackoverflow.com/questions/6490482/are-there-dictionaries-in-php

# Ruby:

## General Info about Ruby:

In Ruby, the associative array is called Hash, and it doesn't require to use any external package of the language. The => operator is used for key value pairs. Also, lines are not terminated. The naming convention is snake_case.

### 1) Initialize Associative Array

```
print "--------------------------- 1 ---------------------------",
"\n"
cs315_instructors = {
  'Section 1' => 'Altay Güvenir',
  'Section 2' => 'Aynur Dayanik',
  'Section 3' => 'Karani Kardas'
}

print cs315_instructors, "\n"
print "--------------------------- 1 ---------------------------",
"\n"
```

```
--------------------------- 1 ---------------------------
{"Section 1"=>"Altay Güvenir", "Section 2"=>"Aynur Dayanik", "Section 3"=>"Karani K
ardas"}
--------------------------- 1 ---------------------------
```

The above code segment initializes the given Hash and prints the result. Positive aspects in terms of the writability:

1) Intuitive and short code

Negative aspects in terms of the writability:

1) No parenthesis in usage of print
2) Print require manual new line insertion

Positive aspects in terms of the readability:

1) Intuitive and short code:

### 2) Get the Value for a Given Key

```
print "--------------------------- 2 ----------------------------",
"\n"
print cs315_instructors['Section 3'], "\n"
print "--------------------------- 2 ----------------------------",
"\n"
```

```
--------------------------- 2 ---------------------------
Karani Kardas
--------------------------- 2 ---------------------------
```

The above code segment gets the value using given key. Positive aspects in terms of the writability:

1) Intuitive and short code

Positive aspects in terms of the readability:

1) Intuitive and short code

### 3) Add a New Element

```
print "--------------------------- 3 ----------------------------",
"\n"
cs315_instructors['Section 4'] = 'Not Available'
print cs315_instructors, "\n"
print "--------------------------- 3 ----------------------------",
"\n"
```

```
--------------------------- 3 ---------------------------
{"Section 1"=>"Altay Güvenir", "Section 2"=>"Aynur Dayanik", "Section 3"=>"Karani K
ardas", "Section 4"=>"Not Available"}
--------------------------- 3 ---------------------------
```

The above code segment adds a given element. Positive aspects in terms of the writability:

1) Intuitive and short code:

Positive aspects in terms of the readability:

1) Intuitive and short code

Negative aspects in terms of the readability:

1) Is it modification or adding statement is not clear

## 4) Remove an Element

```
print "--------------------------- 4 ---------------------------",
"\n"
puts cs315_instructors.delete('Section 4')
print "#{cs315_instructors}\n"
print "--------------------------- 4 ---------------------------",
"\n"
```

```
--------------------------- 4 ---------------------------
Not Available
{"Section 1"=>"Altay Güvenir", "Section 2"=>"Aynur Dayanik", "Section 3"=>"Karani K
ardas"}
--------------------------- 4 ---------------------------
```

The above code segment removes a given key and prints the resulting hash. Positive aspects in terms of the writability:

1) Intuitive and 1 method deletion

Negative aspects in terms of the writability:

1) String formatting is not common

Positive aspects in terms of the readability:

1) Intuitive and short code

## 5) Modify the Value of an Existing Element

```
print "--------------------------- 5 ---------------------------",
"\n"
cs315_instructors['Section 1'] = 'Halil Altay Güvenir'
print "#{cs315_instructors}\n"
print "--------------------------- 5 ---------------------------",
"\n"
```

```
--------------------------- 5 ---------------------------
{"Section 1"=>"Halil Altay Güvenir", "Section 2"=>"Aynur Dayanik", "Section 3"=>"Ka
rani Kardas"}
--------------------------- 5 ---------------------------
```

The above code segment modifies a given element and prints the resulting hash. Positive aspects in terms of the writability:

1) Intuitive and short code

Positive aspects in terms of the readability:

1) Intuitive and short code

## 6) Search for the Existence of a Key

```
print "-------------------------- 6 --------------------------",
"\n"
print cs315_instructors.has_key?('Section 1'), "\n"
print "-------------------------- 6 --------------------------",
"\n"
```

```
-------------------------- 6 --------------------------
true
-------------------------- 6 --------------------------
```

The above code segment searches for the existence of a given key. Positive aspects in terms of the writability:

1) Intuitive and short syntax

Negative aspects in terms of the writability:

1) ? may likely to be forgotten

Positive aspects in terms of the readability:

1) Intuitive and short syntax

## 7) Search for the Existence of a Value

```
print "-------------------------- 7 --------------------------",
"\n"
print cs315_instructors.has_value?('Karani Kardas'), "\n"
print "-------------------------- 7 --------------------------",
"\n"
```

```
-------------------------- 7 --------------------------
true
-------------------------- 7 --------------------------
```

The above code segment searches for the existence of a given value. Positive aspects in terms of the writability:

2) Intuitive, short syntax, 1 method for the given task

Negative aspects in terms of the writability:

2) ? may likely to be forgotten

Positive aspects in terms of the readability:

2) Intuitive and short syntax

## 8) Loop through an Associative Array, apply a Function, called foo, which Simply Prints the Key-Value Pair

```
def foo(k, v)
  puts format('%s : %s', k, v)
end

.
.
.

print "--------------------------- 8 ---------------------------",
"\n"
cs315_instructors.each_pair { |k, v| foo(k, v) }
print "--------------------------- 8 ---------------------------",
"\n"
```

```
--------------------------- 8 ---------------------------
Section 1 : Halil Altay Güvenir
Section 2 : Aynur Dayanik
Section 3 : Karani Kardas
--------------------------- 8 ---------------------------
```

The above code segment defines a function using def keyword and uses it to print key value pairs by using no type declaration (implicit type inference). There is special forEach method of the Ruby called each_pair. Positive aspects in terms of the writability:

1) Special method for for each method

Negative aspects in terms of the writability:

1) |k, v| is confusing
2) String formatting uses modifiers “%s”
3) Each_pair is not common method

Positive aspects in terms of the readability:

1) The code is understandable

Negative aspects in terms of the readability:

1) There is so much difference from the standarts

## Learning Strategy for Ruby:

I mostly uses online forums to search for the problems, but I rarely used the official doc of the Ruby since it is not that bad. I specifically search the problems I have encountered using following links:
https://www.geeksforgeeks.org/ruby-methods/
https://www.geeksforgeeks.org/ruby-hash-has_value-function/
https://www.geeksforgeeks.org/ruby-hash-delete-function/
http://sandbox.mc.edu/~bennet/ruby/code/hash1_rb.html
https://www.rubyguides.com/2012/01/ruby-string-formatting/
https://mixandgo.com/learn/ruby/hash-vs-dictionary

# Rust:

## General Info about Rust:

In Rust, the associative array is called HashMap, and it requires to use any external package of the language. Also, lines are terminated. The naming convention is snake_case.

### 1) Initialize Associative Array

```
use std::collections::HashMap;

fn main() {
    println!("--------------------------- 1 ---------------------------
-");
    let mut cs315_instructors = HashMap::from([
        ("Section 1", "Altay Güvenir"),
        ("Section 2", "Aynur Dayanik"),
        ("Section 3", "Karani Kardas"),
    ]);

    // Error println!("{}", cs315_instructors);
    println!("{:?}", cs315_instructors);
    println!("--------------------------- 1 ---------------------------
-");
```

```
 --------------------------- 1 ---------------------------
{"Section 3": "Karani Kardas", "Section 1": "Altay Güvenir", "Section 2": "Aynur Da
yanik"}
 --------------------------- 1 ---------------------------
```

The above code segment initializes HashMap from key value pairs and prints the resulting dictionary.
Negative aspects in terms of the writability:

1) mut keyword which makes hashmap to be editable can be forgotten
2) Very different syntax from other languages
3) Strict formatting for printing an array

Positive aspects in terms of the readability:

1) mut keyword is useful
2) Code is understandable

Negative aspects in terms of the readability:

1) The HashMap is not printed in order

### 2) Get the Value for a Given Key

```
    println!("--------------------------- 2 --------------------------
-");
    println!("{}", cs315_instructors["Section 3"]);
    println!("--------------------------- 2 --------------------------
-");
```

```
--------------------------- 2 ---------------------------
Karani Kardas
--------------------------- 2 ---------------------------
```

The above code segment gets the value of a given key. Positive aspects in terms of the writability:

1) Intuitive and short syntax

Positive aspects in terms of the readability:

1) Intuitive and short syntax

### 3) Add a New Element

```
    println!("--------------------------- 3 --------------------------
-");
    cs315_instructors.insert("Section 4", "Not Available");

    println!("{:?}", cs315_instructors);
    println!("--------------------------- 3 --------------------------
-");
```

```
-------------------------- 3 --------------------------
{"Section 4": "Not Available", "Section 1": "Altay Güvenir", "Section 3": "Karani K
ardas", "Section 2": "Aynur Dayanik"}
-------------------------- 3 --------------------------
```

The above code segment adds a given key and prints the resulting HashMap. Positive aspects in terms of the writability:

1) Intuitive and short syntax

Negative aspects in terms of the writability:

1) One can use bracket notation

Positive aspects in terms of the readability:

1) Intuitive and short syntax

### 4) Remove an Element

```
    println!("-------------------------- 4 --------------------------
-");
    cs315_instructors.remove("Section 4");

    println!("{:?}", cs315_instructors);
    println!("-------------------------- 4 --------------------------
-");
```

```
-------------------------- 4 --------------------------
{"Section 1": "Altay Güvenir", "Section 3": "Karani Kardas", "Section 2": "Aynur Da
yanik"}
-------------------------- 4 --------------------------
```

The above code segment removes a given key and prints the resulting HashMap.
Positive aspects in terms of the writability:

1) Intuitive and short syntax

Negative aspects in terms of the writability:

1) One can use bracket notation

Positive aspects in terms of the readability:

1) Intuitive and short syntax

### 5) Modify the Value of an Existing Element

```
    println!("--------------------------- 5 --------------------------
-");
    cs315_instructors.insert("Section 1", "Halil Altay Güvenir");
    println!("{:?}", cs315_instructors);
    println!("--------------------------- 5 --------------------------
-");
```

```
--------------------------- 5 --------------------------
{"Section 1": "Halil Altay Güvenir", "Section 3": "Karani Kardas", "Section 2": "Ay
nur Dayanik"}
--------------------------- 5 --------------------------
```

The above code segment modifies a given key and prints the resulting HashMap.
Positive aspects in terms of the writability:

1) Intuitive and short syntax

Negative aspects in terms of the writability:

1) One can use bracket notation

Positive aspects in terms of the readability:

1) Intuitive and short syntax

Negative aspects in terms of the readability:

1) Modifying or adding new element is not clear

## 6) Search for the Existence of a Key

```
    println!("--------------------------- 6 --------------------------
-");
    println!("{}", cs315_instructors.contains_key("Section 1"));
    println!("--------------------------- 6 --------------------------
-");
```

```
--------------------------- 6 --------------------------
true
--------------------------- 6 --------------------------
```

The above code segment searches for the existence of a given key. Positive aspects in terms of the writability:

1) Intuitive and short syntax

Positive aspects in terms of the readability:

1) Intuitive and short syntax

## 7) Search for the Existence of a Value

```
    println!("--------------------------- 7 ---------------------------
-");
    println!( "{}", cs315_instructors.values().any(|&val| val ==
"Karani Kardas"));
    println!("--------------------------- 7 ---------------------------
-");
```

```
--------------------------- 7 ---------------------------
true
--------------------------- 7 ---------------------------
```

The above code segment searches for the existence of a given key. Positive aspects in terms of the writability:

1) Values() list is intuitive

Negative aspects in terms of the writability:

3) Any method is hard to remember

Positive aspects in terms of the readability:

1) Intuitive and short syntax

## 8) Loop through an Associative Array, apply a Function, called foo, which Simply Prints the Key-Value Pair

```
fn foo(k: &str, v: &str) {
    println!("{k}: {v}")
}

.
.
.

    println!("--------------------------- 8 ---------------------------
-");
    for (key, value) in &cs315_instructors {
        foo(key, value);
```

```
    }
    println!("-------------------------- 8 -------------------------
-");
```

```
-------------------------- 8 -------------------------
Section 1: Halil Altay Güvenir
Section 3: Karani Kardas
Section 2: Aynur Dayanik
-------------------------- 8 -------------------------
```

The above code segment defines a function using fn keyword and uses it to print key value pairs by using type declaration. There is no special forEach method of the Rust.
Positive aspects in terms of the writability:

1) Short syntax
2) In keyword

Negative aspects in terms of the writability:

1) &cs315_instructors hard to remember
2) &str parameters hard to remember and preventing the foo function to be generic
3) No built-in foreach method

Positive aspects in terms of the readability:

1) The code is understandable

## Learning Strategy for Rust:

I mostly used the doc of the Rust since it is amazing and every knowledge I needed was there. I specifically search the problems I have encountered using following links:
https://doc.rust-lang.org/book/ch03-03-how-functions-work.html
https://programming-idioms.org/idiom/52/check-if-map-contains-value/455/rust
https://doc.rust-lang.org/std/collections/struct.HashMap.html#method.clear
https://doc.rust-lang.org/std/collections/struct.HashMap.html

## Opinion for the best language to use in Associative Array Operations:

Based on what I discussed above, I think the best language for associative array operations is Dart language since every method has semantic meaning and every task has specific methods to use. Also, it is very readable and writable language.

## Online Compilers I have use (I made workspaces public so you can see the code via link):

I have used replit code environment to test all my codes.

Python: https://replit.com/@MrKty/gultekinomeroktaypython#main.py
Dart: https://replit.com/@MrKty/gultekinomeroktaydart#main.dart
Javascript: https://replit.com/@MrKty/gultekinomeroktayjavascript#index.html
Lua: https://replit.com/@MrKty/gultekinomeroktaylua#main.lua
PHP: https://replit.com/@MrKty/gultekinomeroktayphp#main.php
Ruby: https://replit.com/@MrKty/gultekinomeroktayruby#main.rb
Rust: https://replit.com/@MrKty/gultekinomeroktayrust#src/main.rs