CS 315

PROGRAMMING LANGUAGES

HW #1 REPORT

Berk Yıldız

21502040

Section 1

Table of Contents

1. JavaScript

*Details are given in example program.

- **What types are legal for subscripts?**

  All types are legal for subscripts and any type of subscript does not produce an error. However types except integer give the 'undefined' output.

- **Are subscripting expressions in element references range checked?**

  No. There is no range check mechanism in JavaScript. The script continues to run without any error, but the array returns 'undefined' as the result for out-of-ranged subscript.

- **When are subscript ranges bound?**

  JavaScript is an interpreted language and it has a dynamic structure. So the subscript ranges bound at runtime.

- **When does allocation take place?**

  JavaScript is an interpreted language and it has a dynamic structure. So allocation takes place at runtime.

- **Are ragged or rectangular multidimensional arrays allowed, or both?**

  Both are allowed but multidimensional array is also an array of arrays and it acts like kind of a ragged array.

- **What is the maximum number of subscripts?**

  There is not a defined maximum number of subscripts

- **Can array objects be initialized?**

  Yes array objects can be initialized.

- **Are any kinds of slices supported?**

  Yes, array slice is supported.

- **Which operators are provided?**

  Table is taken from: *https://www.w3schools.com/jsref/jsref_obj_array.asp*

Method	**Description**
concat()	Joins two or more arrays, and returns a copy of the joined arrays

copyWithin()	Copies array elements within the array, to and from specified positions
entries()	Returns a key/value pair Array Iteration Object
every()	Checks if every element in an array pass a test
fill()	Fill the elements in an array with a static value
filter()	Creates a new array with every element in an array that pass a test
find()	Returns the value of the first element in an array that pass a test
findIndex()	Returns the index of the first element in an array that pass a test
forEach()	Calls a function for each array element
from()	Creates an array from an object
includes()	Check if an array contains the specified element
indexOf()	Search the array for an element and returns its position
isArray()	Checks whether an object is an array

join()	Joins all elements of an array into a string
keys()	Returns a Array Iteration Object, containing the keys of the original array
lastIndexOf()	Search the array for an element, starting at the end, and returns its position
map()	Creates a new array with the result of calling a function for each array element
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
reduce()	Reduce the values of an array to a single value (going left-to-right)
reduceRight()	Reduce the values of an array to a single value (going right-to-left)
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element
slice()	Selects a part of an array, and returns the new array
some()	Checks if any of the elements in an array pass a test

sort()	Sorts the elements of an array
splice()	Adds/Removes elements from an array
toString()	Converts an array to a string, and returns the result
unshift()	Adds new elements to the beginning of an array, and returns the new length
valueOf()	Returns the primitive value of an array

- **Are associative arrays available?**

  Yes, associative array is available.

- **JavaScript Code and Outputs**

```
<script>
document.write("----------Types For Legal Subscripts----------<br>");
var books = ["book1", "book2", "book3", "book4"];
document.write(books[0]); //output: book1
document.write("<br>");
a=1;
document.write(books[a]); //output: book2
document.write("<br>");
c = "a"
document.write(books[c]); //output: undefined (not an error)
document.write("<br><br>");
document.write("----------Range Check----------<br>");
document.write(books[5]); //output: undefined (not an error)
document.write("<br>");
document.write(books[-1]); //output: undefined (not an error)
document.write("<br><br>");
document.write("----------Ragged & Rectangular Multidimensioned Arrays----------<br><br
    >");
document.write("Example for ragged array:<br>");
var ragged = [[10, 20, 30], [10, 20]]
document.write(ragged); //output: 10,20,30,10,20
document.write("<br>");
document.write("<br>Example for artificial multidimensional array: <br>");
var mDimension = [
   ["a", 10, 1000],
   ["b", 20, 2000],
   ["c", 30, 3000],
   ["d", 40, 4000],
];
document.write("mDimension[1][1] = " + mDimension[1][1]); //output: 20
document.write("<br><br>");
document.write("----------Array Initialization----------<br>");
var init = ["Arrays ", "can ", "be ", "initialized ", "in ", "JavaScript."];
document.write(init); //output: Arrays ,can ,be ,initialized ,in ,JavaScript.
```

```
document.write("<br><br>----------Array Slice----------<br>");
var sliceExample = ["Slice ", "is ", "supported ", "in ", "JavaScript"];
var sliced = sliceExample.slice(2, 5);
document.write(sliced); //output: supported ,in ,JavaScript
document.write("<br><br>----------Array Operations----------<br><br>");
document.write("Concatenation: <br>");
var first = [1,2];
var second = [3,4];
var final = first.concat(second);
document.write("Concatenation of " + first + " and " + second + ": " + final); //output: 1,2,3,4
document.write("<br><br>Copying array elements within the array, to and from specified positions: <br>");
first = [1,2,3,4,5,6];
document.write("Before calling function: " + first + "<br>");
document.write("After calling fcuntion: " + first.copyWithin(3, 0, 2) + "<br><br>"); //output: 1,2,3,1,2,6
document.write("Checking if an element includes in an array: <br>");
document.write(first.includes(3) + "<br>"); //output: true
document.write(first.includes(10 + "<br>")); //output: false
document.write("<br><br>Sorting: <br>");
document.write(first.sort()); //output: 1,1,2,2,3,6
document.write("<br><br>----------Associative Array----------<br>");
var capitals = {"Turkey": "Ankara", "UK": "London", "France": "Paris"};
document.write(capitals["UK"]); //output: London

</script>
```

----------Types For Legal Subscripts----------
book1
book2
undefined

----------Range Check----------
undefined
undefined

----------Ragged & Rectangular Multidimensioned Arrays----------

Example for ragged array:
10,20,30,10,20

Example for artificial multidimensional array:
mDimension[1][1] = 20

----------Array Initialization----------
Arrays ,can ,be ,initialized ,in ,JavaScript.

----------Array Slice----------
supported ,in ,JavaScript

----------Array Operations----------

Concatenation:
Concatenation of 1,2 and 3,4: 1,2,3,4

Copying array elements within the array, to and from specified positions:
Before calling function: 1,2,3,4,5,6
After calling fcuntion: 1,2,3,1,2,6

```
Checking if an element includes in an array:
true
false

Sorting:
1,1,2,2,3,6

----------Associative Array----------
London
```

2. Python

*Details are given in example program.

- **What types are legal for subscripts?**

  Only integer types are legal for the subscripts of lists. However dictionaries accept all types as subscript according to its key elements.

- **Are subscripting expressions in element references range checked?**

  Yes, there is range check in Python.

- **When are subscript ranges bound?**

  Python is an interpreted language and it has a dynamic structure. So the subscript ranges bound at runtime.

- **When does allocation take place?**

  Python is an interpreted language and it has a dynamic structure. So allocation takes place at runtime

- **Are ragged or rectangular multidimensional arrays allowed, or both?**

  Ragged lists are allowed but multidimensional is not allowed natively. Some operations have to be done for having a multidimensional array.

- **What is the maximum number of subscripts?**

  There is not a defined limit.

- **Can array objects be initialized?**

  Yes, lists can be initialized.

- **Are any kind of slices supported?**

  Yes, slices are supported.

- **Which operators are provided?**

  Table is taken from: *https://www.geeksforgeeks.org/python-list/*

Function	Description
Append()	Add an element to the end of the list
Extend()	Add all elements of a list to the another list
Insert()	Insert an item at the defined index
Remove()	Removes an item from the list
Pop()	Removes and returns an element at the given index
Clear()	Removes all items from the list
Index()	Returns the index of the first matched item
Count()	Returns the count of number of items passed as an argument
Sort()	Sort items in a list in ascending order
Reverse()	Reverse the order of items in the list
copy()	Returns a copy of the list
reduce()	apply a particular function passed in its argument to all of the list elements stores the intermediate result and only returns the final summation value
sum()	Sums up the numbers in the list
ord()	Returns an integer representing the Unicode code point of the given Unicode character
cmp()	This function returns 1, if first list is "greater" than second list
max()	return maximum element of given list
min()	return minimum element of given list
all()	Returns true if all element are true or if list is empty
any()	return true if any element of the list is true. if list is empty, return false
len()	Returns length of the list or size of the list
enumerate()	Returns enumerate object of list
accumulate()	apply a particular function passed in its argument to all of the list elements returns a list containing the intermediate results
filter()	tests if each element of a list true or not
map()	returns a list of the results after applying the given function to each item of a given iterable
lambda()	This function can have any number of arguments but only one expression, which is evaluated and returned.

- **Are associative arrays available?**

  Yes, associative arrays are available by dictionary data type.

```
print("----------Types For Legal Subscripts----------\n")
books = ["book1", "book2", "book3", "book4"]
print(books[0]) #output: book1
print("\n")
a=1
print(books[a]) #output: book2
print("\n")
c = "a"
#print(books[c]) #output: TypeError: list indices must be integers or slices, not str
print("\n")
print("----------Range Check----------\n")
#print(books[5]) #output: IndexError: list index out of range
#print("\n")
print("----------Ragged & Rectangular Multidimensioned Arrays----------\n\n")
print("Example for ragged array:\n")
ragged = [[10, 20, 30], [10, 20]]
print(ragged) #output: [[10, 20, 30], [10, 20]]
print("\n")
print("\nExample for artificial multidimensional array: \n")
mDimension = [
    ["a", 10, 1000],
    ["b", 20, 2000],
    ["c", 30, 3000],
    ["d", 40, 4000],
]
print("mDimension[1][1] = " , mDimension[1][1]) #output: 20
print("----------Array Initialization----------\n")
init = ["Arrays ", "can ", "be ", "initialized ", "in ", "JavaScript."]
print(init) #output: Arrays ,can ,be ,initialized ,in ,JavaScript.
print("\n\n----------Array Slice----------\n")
print("First way by using slice():\n")
sliceExample = ["Slice ", "is ", "supported ", "in ", "JavaScript"]
sliced = slice(2, 5)
print(sliceExample[sliced], "\n") #output: ['supported ', 'in ', 'JavaScript']
print("Second way:\n")
sliced = sliceExample[2:5] #output: ['supported ', 'in ', 'JavaScript']
print(sliced)
```

```
print("\n\n----------Array Operations----------\n\n")
print("Append: ")
first = [1, 2, 3]
print("Before calling function: ", first) #output: [1, 2, 3]
first.append(4)
print("After calling function: ", first) #output: [1, 2, 3, 4]
print("Insert: ")
print("Before calling function: ", first) #output: [1, 2, 3, 4]
first.insert(2, 5)
print("After calling function: ", first) #output: [1, 2, 5, 3, 4]
print("Sort: ")
second = [23, 10, 9, 87, 1]
print("Before calling function: ", second) #output: [23, 10, 9, 87, 1]
second.sort()
print("After calling function: ", second) #output: [1, 9, 10, 23, 87]
print("Sum: ")
print("Sum of the elements in the list is", sum(second))
print("\n\n----------Associative Array----------\n")
#actually dictionary
capitals = {'Turkey': 'Ankara', 'UK': 'London', 'France': 'Paris'}
print(capitals) #output: {'Russia': 'Moscow', 'Ukraine': 'Kiev', 'USA': 'Washington'}
```

```
$python main.py
----------Types For Legal Subscripts----------

book1


book2



----------Range Check----------

----------Ragged & Rectangular Multidimensioned Arrays----------


Example for ragged array:

[[10, 20, 30], [10, 20]]



Example for artificial multidimensional array:

('mDimension[1][1] = ', 20)
----------Array Initialization----------

['Arrays ', 'can ', 'be ', 'initialized ', 'in ', 'JavaScript.']


----------Array Slice----------

First way by using slice():

(['supported ', 'in ', 'JavaScript'], '\n')
Second way:

['supported ', 'in ', 'JavaScript']



----------Array Operations----------


Append:
('Before calling function: ', [1, 2, 3])
('After calling function: ', [1, 2, 3, 4])
Insert:
('Before calling function: ', [1, 2, 3, 4])
('After calling function: ', [1, 2, 5, 3, 4])
Sort:
('Before calling function: ', [23, 10, 9, 87, 1])
('After calling function: ', [1, 9, 10, 23, 87])
Sum:
('Sum of the elements in the list is', 130)


----------Associative Array----------

{'Turkey': 'Ankara', 'UK': 'London', 'France': 'Paris'}
```

3. PHP

*Details are given in example program.

- **What types are legal for subscripts?**

  Integer are legal for regular arrays, however all types are legal for the arrays which are in map structure.

- **Are subscripting expressions in element references range checked?**

  Yes, there is range check in PHP.

- **When are subscript ranges bound?**

  PHP is an interpreted and dynamic language. So, subscript ranges bounding occurs at runtime.

- **When does allocation take place?**

  PHP is an interpreted and dynamic language. So, allocation takes place at runtime.

- **Are ragged or rectangular multidimensional arrays allowed, or both?**

  Both ragged and rectangular multidimensional arrays are allowed.

- **What is the maximum number of subscripts?**

  There is not a defined limit.

- **Can array objects be initialized?**

  Yes, array objects can be initialized.

- **Are any kind of slices supported?**

  Yes, slices are supported in PHP.

- **Which operators are provided?**

  Table is taken from: *https://www.w3schools.com/php/php_ref_array.asp*.

Operator & Function	**Description**
Union (+)	Union of $x and $y
Equality (==)	Returns true if $x and $y have the same key/value pairs
Identity (===)	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
Inequality (!=)	Returns true if $x is not equal to $y

Inequality (<>)	Returns true if $x is not equal to $y
Non-identity (!==)	Returns true if $x is not identical to $y
array()	Creates an array
array_change_key_case()	Changes all keys in an array to lowercase or uppercase
array_chunk()	Splits an array into chunks of arrays
array_column()	Returns the values from a single column in the input array
array_combine()	Creates an array by using the elements from one "keys" array and one "values" array
array_count_values()	Counts all the values of an array
array_diff()	Compare arrays, and returns the differences (compare values only)
array_diff_assoc()	Compare arrays, and returns the differences (compare keys and values)
array_diff_key()	Compare arrays, and returns the differences (compare keys only)
array_diff_uassoc()	Compare arrays, and returns the differences (compare keys and values, using a user-defined key comparison function)

array_diff_ukey()	Compare arrays, and returns the differences (compare keys only, using a user-defined key comparison function)
array_fill()	Fills an array with values
array_fill_keys()	Fills an array with values, specifying keys
array_filter()	Filters the values of an array using a callback function
array_flip()	Flips/Exchanges all keys with their associated values in an array
array_intersect()	Compare arrays, and returns the matches (compare values only)
array_intersect_assoc()	Compare arrays and returns the matches (compare keys and values)
array_intersect_key()	Compare arrays, and returns the matches (compare keys only)
array_intersect_uassoc()	Compare arrays, and returns the matches (compare keys and values, using a user-defined key comparison function)
array_intersect_ukey()	Compare arrays, and returns the matches (compare keys only, using a user-defined key comparison function)
array_key_exists()	Checks if the specified key exists in the array
array_keys()	Returns all the keys of an array

array_map()	Sends each value of an array to a user-made function, which returns new values
array_merge()	Merges one or more arrays into one array
array_merge_recursive()	Merges one or more arrays into one array recursively
array_multisort()	Sorts multiple or multi-dimensional arrays
array_pad()	Inserts a specified number of items, with a specified value, to an array
array_pop()	Deletes the last element of an array
array_product()	Calculates the product of the values in an array
array_push()	Inserts one or more elements to the end of an array
array_rand()	Returns one or more random keys from an array
array_reduce()	Returns an array as a string, using a user-defined function
array_replace()	Replaces the values of the first array with the values from following arrays
array_replace_recursive()	Replaces the values of the first array with the values from following arrays recursively

array_reverse()	Returns an array in the reverse order
array_search()	Searches an array for a given value and returns the key
array_shift()	Removes the first element from an array, and returns the value of the removed element
array_slice()	Returns selected parts of an array
array_splice()	Removes and replaces specified elements of an array
array_sum()	Returns the sum of the values in an array
array_udiff()	Compare arrays, and returns the differences (compare values only, using a user-defined key comparison function)
array_udiff_assoc()	Compare arrays, and returns the differences (compare keys and values, using a built-in function to compare the keys and a user-defined function to compare the values)
array_udiff_uassoc()	Compare arrays, and returns the differences (compare keys and values, using two user-defined key comparison functions)
array_uintersect()	Compare arrays, and returns the matches (compare values only, using a user-defined key comparison function)
array_uintersect_assoc()	Compare arrays, and returns the matches (compare keys and values, using a built-in

	function to compare the keys and a user-defined function to compare the values)
array_uintersect_uassoc()	Compare arrays, and returns the matches (compare keys and values, using two user-defined key comparison functions)
array_unique()	Removes duplicate values from an array
array_unshift()	Adds one or more elements to the beginning of an array
array_values()	Returns all the values of an array
array_walk()	Applies a user function to every member of an array
array_walk_recursive()	Applies a user function recursively to every member of an array
arsort()	Sorts an associative array in descending order, according to the value
asort()	Sorts an associative array in ascending order, according to the value
compact()	Create array containing variables and their values
count()	Returns the number of elements in an array
current()	Returns the current element in an array
each()	Deprecated from PHP 7.2. Returns the

	current key and value pair from an array
end()	Sets the internal pointer of an array to its last element
extract()	Imports variables into the current symbol table from an array
in_array()	Checks if a specified value exists in an array
key()	Fetches a key from an array
krsort()	Sorts an associative array in descending order, according to the key
ksort()	Sorts an associative array in ascending order, according to the key
list()	Assigns variables as if they were an array
natcasesort()	Sorts an array using a case insensitive "natural order" algorithm
natsort()	Sorts an array using a "natural order" algorithm
next()	Advance the internal array pointer of an array
pos()	Alias of current()
prev()	Rewinds the internal array pointer

range()	Creates an array containing a range of elements
reset()	Sets the internal pointer of an array to its first element
rsort()	Sorts an indexed array in descending order
shuffle()	Shuffles an array
sizeof()	Alias of count()
sort()	Sorts an indexed array in ascending order
uasort()	Sorts an array by values using a user-defined comparison function
uksort()	Sorts an array by keys using a user-defined comparison function
usort()	Sorts an array using a user-defined comparison function

- **Are associative arrays available?**

  Yes, associative arrays are available.

```php
<?php
    //Some lines are commented in sake of execution.
   echo "----------Types For Legal Subscripts----------\n";
   $books = array("book1", "book2", "book3", "book4");
   echo $books[0] , "\n"; //output: book1
   $a=1;
   echo $books[$a] , "\n"; //output: book2
   $b="c";
   //echo $books[$b] , "\n"; //output: PHP Notice:  Undefined index: c
   echo "----------Range Check----------\n";
   //echo ($books[5]); //output: Undefined offset: 5
   echo "There is range check!\n";
   //echo($books[-1]); //output: Undefined offset: -1
   echo("----------Ragged & Rectangular Multidimensioned Arrays----------\n");
   //It is like array of arrays.
   $mDimension = array
  (
  array("A",10,100),
  array("B",20,200),
  array("C",30,300),
  array("D",40,400)
  );

  echo $mDimension[2][2], "\n";
  echo ("----------Array Initialization----------\n");
  $init = array("Arrays ", "can ", "be ", "initialized ", "in ", "PHP.");
  print_r($init); //output: Arrays ,can ,be ,initialized ,in ,PHP.
  echo ("----------Array Slice----------\n");
  $sliceExample = array("Slice ", "is ", "supported ", "in ", "PHP");
  print_r(array_slice($sliceExample,2,5));// output: supported, in, PHP
```

```php
  echo ("----------Array Operations----------\n");
  echo ("Union: \n");
  $first = array(1 => "a", 2 => "b");
  $second = array(3 => "c", 4 => "d");
  print_r($first + $second); //output is the union of two arrays
  echo ("Equality: \n");
  var_dump($first == $second);//output is false because arrays have different pairs
  echo ("Identity: \n");
  $test1 = array(1 => "a", 2 => "b");
  $test2 = array(1 => "a", 2 => "b");
  var_dump($test1 === $test2); //output is true because arrays have same paris in same orders
  echo ("Inequality: \n");
  var_dump($first != $second); //output is treu because arrays are not same
  echo ("Non-identity: \n");
  var_dump($test1 !== $test2); //output is false because arrays are identical
  echo "----------Associative Array----------\n";
  $test = array("a"=>"10", "b"=>"20", "c"=>"30");
  print_r($test);
```

```
----------Types For Legal Subscripts----------
book1
book2
----------Range Check----------
There is range check!
----------Ragged & Rectangular Multidimensioned Arrays----------
300
```

```
----------Array Initialization----------
Array
(
    [0] => Arrays
    [1] => can
    [2] => be
    [3] => initialized
    [4] => in
    [5] => PHP.
)
----------Array Slice----------
Array
(
    [0] => supported
    [1] => in
    [2] => PHP
)
----------Array Operations----------
Union:
Array
(
    [1] => a
    [2] => b
    [3] => c
    [4] => d
)
Equality:
bool(false)
Identity:
bool(true)
Inequality:
bool(true)
Non-identity:
bool(false)
----------Associative Array----------
Array
(
    [a] => 10
    [b] => 20
    [c] => 30
)
```

4. Dart

*Details are given in example program.

- **What types are legal for subscripts?**

  Like Python, Dart has lists instead of arrays. Only integers are legal for subscripts.

- **Are subscripting expressions in element references range checked?**

  Yes, there is range check in Dart.

- **When are subscript ranges bound?**

  There is not clear information about this topic but there are two types of lists: fixed size and growable. Probably fixed size lists are acting like static arrays and subscript ranges bound at compile time for static lists. Growable lists can act like dynamic arrays and subscript ranges can be bound runtime.

- **When does allocation take place?**

  Like I mentioned in the question above, if fixed size lists are acting like static arrays, allocation takes place at compile time and if growable lists are acting like dynamic arrays, allocation takes place at runtime.

- **Are ragged or rectangular multidimensional arrays allowed, or both?**

  Both of them are allowed. Multidimensional lists can be created by generate method.

- **What is the maximum number of subscripts?**

  There is not a defined limit.

- **Can array objects be initialized?**

  Yes, list objects can be initialized.

- **Are any kind of slices supported?**

  Yes, list slices are supported.

- **Which operators are provided?**

  List is taken from: *https://medium.com/flutter-community/useful-list-methods-in-dart-6e173cac803d*

Methods	**Description**
sublist()	This method returns a new list containing elements from index between start and end.
shuffle()	This method re-arranges order of the elements in the given list randomly.
reversed	reversed is a getter which reverses iteration of the list depending upon given list order
asMap()	This method returns a map representation of the given list.
whereType()	This method returns iterable with all elements of specific data type
getRange()	This method returns elements from specified range [start] to [end] in same order as in the given list.
replaceRange()	This method helps to replace / update some elements of the given list with the new ones.
firstWhere()	This method returns the first element from the list when the given condition is satisfied.
singleWhere()	This method returns the first matching element from the list when there's an *exact* match.

fold()	This method returns a single value by iterating all elements of given list.
reduce()	This method is very similar to fold and returns a single value by iterating all elements of given list.
followedBy()	This method appends new iterables to the given list.
any()	This method returns a boolean depending upon whether *any* element satisfies the condition or not.
every()	This method returns a boolean depending upon whether *all* elements satisfies the condition or not.
take()	This method returns iterable starting from index 0 till the count provided from given list.
skip()	This method ignores the elements starting from index 0 till count and returns remaining iterable from given list.
add()	adds new element in given list.
length()	returns total number of elements in given list.
isEmpty()	returns boolean if given list is empty or not.
isNotEmpty()	checks and returns boolean if given list has elements.
first()	returns first element from given list.
last()	returns last element from given list.
clear()	clears the given list.

- **Are associative arrays available?**

  There is map data type which acts like associative arrays.

```dart
void main(){
  print("----------Types For Legal Subscripts----------\n");
  var books = ["book1", "book2", "book3"];
  print(books[0]); //output: book1
  var a = "b";
  //print(books[a]); //outputs: Invalid argument(s): b
  print("----------Range Check----------\n");
  //print(books[5]); //output: RangeError: index (5) must be in the range [0..3)
  print("There is range check.");
  print("----------Ragged & Rectangular Multidimensioned Arrays----------\n");
  print("Ragged: ");
  var raggedTest = [1 ,2 ,3, [4, 5]];
  print(raggedTest); //output: [1, 2, 3, [4, 5]]
  print("Multidimensional: ");

  //var mdTest = List.generate(4, (i) => List.generate(4, (j) => i + j));
   //print(mdTest);
  print("----------Array Initialization----------\n");
  var init = ["Lists ", "can ", "be ", "initialized ", "in ", "Dart."];
  print(init);
  print("----------Array Slice----------\n");
  var sliceExample = ["Slice ", "is ", "supported ", "in ", "Dart."];
  print(sliceExample.sublist(2,5)); //output: [supported , in , Dart.]
  print ("----------Array Operations----------\n");
  var first = [1, 2, 3, 4, 5, 6, 7];
  print("Reverse: ");
  print(first.reversed.toList()); //reverse the elements of the list
  print("Replace range");
  first.replaceRange(1,4, [5]); //replace the list elements with given number in the given range
  print('$first'); //output: [1, 5, 5, 6, 7]
  print("Fold: ");
  //print (first.fold(1, (i, j) => i * j)); //output: 1050 (multiplies all elements of the list)
  print("Shuffle: ");
  first.shuffle(); //shuffles the elements of the list
  print('$first');
  print ("----------Associative Array----------\n");
  //there is map type for associative arrays.
  var test = {'a': 10,'b': 20, 'c': 30};
  print(test); //output: {a: 10, b: 20, c: 30}
```

```
----------Types For Legal Subscripts----------

book1
----------Range Check----------

There is range check.
----------Ragged & Rectangular Multidimensioned Arrays-

Ragged:
[1, 2, 3, [4, 5]]
Multidimensional:
----------Array Initialization----------

[Lists , can , be , initialized , in , Dart.]
----------Array Slice----------

[supported , in , Dart.]
----------Array Operations----------

Reverse:
[7, 6, 5, 4, 3, 2, 1]
Replace range
[1, 5, 5, 6, 7]
Fold:
Shuffle:
[6, 1, 7, 5, 5]
----------Associative Array----------

{a: 10, b: 20, c: 30}
```

5. Rust

*Details are given in example program.

- **What types are legal for subscripts?**

  Type called ‘usize’ is legal for subscripts. Usize is a kind of representation of integer.

- **Are subscripting expressions in element references range checked?**

  Yes, there is range check in Rust.

- **When are subscript ranges bound?**

  In the arrays section of official website of Rust, it is stated that arrays are static. So subscript ranges bound at compile time for static arrays.

- **When does allocation take place?**

  If the arrays are static, also allocation takes place in compile time.

- **Are ragged or rectangular multidimensional arrays allowed, or both?**

  Ragged arrays are allowed but some packages should be imported to create ragged arrays. Multidimensional arrays are allowed but also a helper function can be used to create multidimensional arrays.

- **What is the maximum number of subscripts?**

  There is not a defined limit.

- **Can array objects be initialized?**

  Yes, array objects can be initialized.

- **Are any kind of slices supported?**

  Yes, array slices are supported.

- **Which operators are provided?**

  I could not find a source to answer this question. I just encounter with a len() function which returns the length of an array in the examples I’ve seen.

- **Are associative arrays available?**

  Yes, there are hashmaps which are in same functionality with associative arrays.

```rust
use std::collections::HashMap;
//Some lines are commented in sake of execution.

//This function is taken from official Rust documentation.
fn analyze_slice(slice: &[i32]) {
    println!("first element of the slice: {}", slice[0]);
    println!("the slice has {} elements", slice.len());
}

fn main() {

    println!("----------Types For Legal Subscripts----------\n");
    let books = ["book1", "book2", "book3"];
    println!("{}", books[0]); //output: book1
    //let a = "c";
    //println!("{}", books[a]); //output: error[E0277]: the trait bound `&str: std::slice::SliceIndex<[&str]>` is not satisfied
    println!("----------Range Check----------\n");
    //println!("{}", books[5]);//give both error and warning
    println!("There is range check");
    println!("----------Ragged & Rectangular Multidimensioned Arrays----------\n");
    println!("Ragged: natively not allowed, but by importing some libraries or packages, ragged arrays can be built");
    //let ragged = [1, 2, 3, [4, 5]]; natively not allowed. gives error error[E0308]: mismatched types, expected integral
        variable, found array
    println!("Multidimensional: ");
    let mut state = [[0u8; 4]; 6];
    state[0][1] = 42;
    println!("{}", state[0][1]); //output: 42
    println!("----------Array Initialization----------\n");
    let init = ["Lists ", "can ", "be ", "initialized ", "in ", "Rust."];
    println!("{:?}", init); //output: ["Lists ", "can ", "be ", "initialized ", "in ", "Rust."]
```

```rust
    println!("----------Array Slices----------\n");
    //Array slice example is taken from official Rust documentation.
    let ys: [i32; 500] = [0; 500];
    analyze_slice(&ys[1 .. 4]);
    //output: first element of the slice: 0
    //the slice has 3 elements

    println!("----------Array Operations----------\n");
    let test = [1, 2, 3, 4, 5];
    println!("Length :{}",test.len()); //output:5 (returns size of the array)
    println!("----------Associative Array----------\n");
    //There is hashmap type for pretending associative arrays.
    let mut test1 = HashMap::new();
    test1.insert(String::from("a"), 10);
    test1.insert(String::from("b"), 20);

    for (key, value) in &test1 {
    println!("{}: {}", key, value);
    }
```

```
$rustc -o main *.rs
$main
----------Types For Legal Subscripts----------

book1
----------Range Check----------

There is range check
----------Ragged & Rectangular Multidimensioned Arrays----------

Ragged: natively not allowed, but by importing some libraries or packages, ragged arrays can be built
Multidimensional:
42
----------Array Initialization----------

["Lists ", "can ", "be ", "initialized ", "in ", "Rust."]
----------Array Slices----------

first element of the slice: 0
the slice has 3 elements
----------Array Operations----------

Length :5
----------Associative Array----------

b: 20
a: 10
```

6. Best Language for Arrays

Arrays are the most used data type in application development and offer very convenient opportunities for data management. Even in different structures or names, there is an array structure in almost all modern programming languages. While doing homework, I had the opportunity to research and analyze the sequence structures of five different languages. All of them have both advantages and disadvantages for array handling. If I consider my own programming habits, my expectations from an array structure is that it should be have good writability and I should perform my operations with as few lines as possible, using fewer loops. From my perspective, I think Python is the best for array operations. Because of its dynamic structure, it is very easy to define lists and assign elements to it. Also it has various functions which avoid thinking of array algorithms and there is usually no need to write a loop. Beside of the default functions, also it provides thousands of available functions via libraries. It can have reliability and performance issues but these are programming related issues and should not be evaluated in the scope of array structure. So I can say that the best language is Python when I only consider the issue of array.

7. Learning Strategies

There were both theoretical and experimental sections in the homework that needed to be studied. Before that, it was necessary to learn the syntax of languages. So, different learning practice strategies are required for these three different concepts.

First of all, the all sources that I used are on the internet. Internet offers the fastest and most efficient learning opportunity in programming-related subjects. But even though all the sources are on the internet, I used different approaches for the theoretical & experimental sections and understanding the syntax.

Five different languages have their own unique syntax. The important parts for the homework were declaring an array, setting elements to an array, accessing the array elements and function calls if necessary. There are some websites called *tutorialspoint.com*, *w3schools.com* and *geekforgeeks.com* which clearly denotes the syntax of JavaScript, Python, PHP and Dart. I used these websites for faster

understanding of the syntax of these four languages. However there is not a third party source which present deep information about Rust. There are some information in *tutorailspoint.com* about Rust but it is not sufficient for the homework. So I used official documentation of Rust for learning its syntax which is published on official website.

The theoretical parts of the homework consists of questions like *'When are subscript ranges bound?'* or *'When does allocation take place?'*. It is not possible to find answer these questions by conducting some experiments because as I know compilers do not provide information for this kind of issues. So a research is needed to provide answer to these questions. The course book includes the information about binding time of static arrays, fixed stack-dynamic arrays, fixed-heap dynamic array and heap-dynamic array. There is information for the array types of JavaScript, Python and PHP on *stackoverflow.com.* Rust gives information on its official document and there is some information about the list structure of Dart in *tutorialspoint.com.* After having the information of array type, course book gives the information about is it statically bound or something else.

The experimental part of the homework consists of questions about, type of subscripts, range check, ragged & multidimensional arrays, array initialization, array slices, array operations and associative array. Range check and array initialization can be understood with a simple experiment without conducting any research. For deciding the acceptable subscripts, I tried all primitive data types of the languages as subscript and followed the outputs of code snippets. For range check issue, I provided a subscript which is out of range in all languages and followed the output. For array initialization, I simply initialize array with objects in all languages and followed to understand if it causing an error or not. For the other issues like array operations, I had to search the internet to see if these components exist in the languages. After that, I applied the existing components to see how they work in which conditions. I took the operations tables directly from the websites because I think there is no need to discover them from scratch.