# CS315 – Homework Assignment 1

Burak Ozturk - Section 3 - 21901841

## Dart Language

1. What types are legal for subscripts?

There are three trivial types that support subscripting in Dart. They are String, List and Map.

Code segment:

String s = "This is a string!";

print("String: $s\n");

print("First char of string: "); print(s[0]);

print("Seventh char of string: "); print(s[6]);

print("\nString is subscriptable\n");

List list = [0,1,2,3,4];

print("List: $list\n");

print("First element of list: "); print(list[0]);

print("Third element of list: "); print(list[2]);

print("\nList is subscriptable");

Map map = {"a": "apple", "b": "banana", "c": "cherry"};

print("Map: $map\n");

print("Key a: "); print(map["a"]);

print("Key c: "); print(map["c"]);

print("\nMap is subscriptable");

1. Are subscripting expressions in element references range checked?

Yes, Dart enforces range checking on arrays since trying to reach to beyond bounds of an array raises a RangeError.

Code segment:

print("List: $list, Length = 5");

print("Fifth element: "); print(list[4]); print("");

print("Trying to reach for sixth");

try {

print(list[5]);

}

on RangeError {

print("RangeError caught, thus Dart utilizes range checking\n");

}

1. When are subscript ranges bound?

Rust arrays can accept more items after they are initialized, thus they have dynamic range bounding.

Code segment:

List list2 = [0,1,2,3]; print("Normal list: $list2");

print("Trying to add a fifth item");

list2.add(4);

print("Fourth item: "); print(list2[4]);

print("It is possible to add after initialization, thus lists have dynamic range bounding\n");

const List cList = [0,1,2,3]; print("Const list: $cList");

print("Trying to add a fifth item");

try {

cList.add(5);

}

on UnsupportedError {

print("UnsupportedError caught, thus const lists have static range bounding\n");

}

1. When does allocation take place?

Dart has dynamically allocated arrays.

Code segment:

void dynamicList() {

List funcList = [4,7,4];

print("List from dynamicList(): $funcList");

}

print("Calling a function that will create a list and print it: ");

dynamicList();

// This is causing compile error:

//print("\nThan printing that list from main function: $cFuncList\n");

print("Trying to access same list from main function causes compile-time error\nthus lists have dynamic allocation");

1. Are ragged or rectangular multidimensional arrays allowed, or both?

Dart supports both ragged and rectangular multi-dimensional arrays.

Code segment:

print("Trying to create rectangular MD array");

List mdArr = [[1,2,3],[4,5,6]];

print("mdArr: $mdArr");

print("\nThus it is possible to have rectangular multidimensional arrays\n");

print("Trying to add another element to the latter array inside mdArr (mdArr[1])");

mdArr[1].add(7);

print("mdArr: $mdArr");

print("\nThus it is possible to have both ragged and rectangular multidimensional arrays\n");

1. Can array objects be initialized?

Arrays can be initialized as they declared or after they declared.

Code segment:

print("Lists can be initialized as they declared or after they declared:\n");

print("afterArr: List afterArr; afterArr = [9,9,9]; print(afterArr);");

List afterArr; afterArr = [9,9,9]; print(afterArr);

print("\nasArr: List asArr = [4,3,6]; print(asArr);");

List asArr = [4,3,6]; print(asArr);

1. Are any kind of slices supported?

Dart has a sublist() function for slicing arrays.

Code segment:

print("\nSlicing with sublist function:\n");

print("list: $list\n");

print("list.sublist(2,4): ");

print(list.sublist(2,4));

1. Which operators are provided?

There are concentation (+), comparing (==) and assignment (=) operators.

Code segment:

List list1 = [1,2,3], list2 = [4,5,6];

print("list1: $list1, list2: $list2");

print("\nlist1 + list2:"); print(list1 + list2);

print("\nlist1 == list2:"); print(list1 == list2);

print("\nlist1[1]:"); print(list1[1]);

print("\nlist1[1] = 56:"); list1[1] = 56; print(list1);

## JavaScript

1. What types are legal for subscripts?

Strings and arrays are legal for subscripts.

Code segment:

var s = "This is a string!";

console.log(`String s: \"${s}\"\n`);

console.log(`s[3]: ${s[3]}, s[11]: ${s[11]}\n`);

console.log("String is subscriptable\n\n");

var arr = [24, 75, 34, 91, 77];

console.log(`Array arr: ${arr}\n`);

console.log(`arr[1]: ${arr[1]}, arr[3]: ${arr[3]}\n`);

console.log("Array is subscriptable\n\n");

1. Are subscripting expressions in element references range checked?

No, JavaScript doesn’t have range checking since following code segment caught no errors.

Code segment:

console.log(`arr: ${arr} - Length = ${arr.length}\n`);

console.log(`Fifth element: ${arr[4]}\n`);

console.log(`Trying to reach for sixth:\n`);

caught = false;

try {

console.log(`Sixth element: ${arr[5]}\n`);

}

catch (e) {

console.log(`${e.message} caught thus JS has range checking\n\n`);

caught = true;

}

if (!caught) {

console.log("No error caught thus JS doesn't have range checking\n\n");

}

1. When are subscript ranges bound?

JavaScript has dynamic range bounding since following code segment raised no errors.

Code segment:

var arr1 = [0,1,2,3,4];

console.log(`Array arr1: ${arr1}\n`);

console.log(`Trying to add another element:\n`);

caught = false;

try {

arr1[5] = 5;

console.log(`arr1[5]: ${arr1[5]}\n`);

}

catch (e) {

console.log(`\"${e.message}\" error caught thus JS has static range bounding\n\n`); caught = true;

}

if (!caught) {

console.log("No error caught thus JS has dynamic range bounding\n\n");

}

1. When does allocation take place?

JavaScript has dynamic array allocation.

Code segment:

function dynamicList() {

var cFuncList = [4,7,4];

console.log(`Array from dynamicList(): ${cFuncList}\n`);

}

console.log("Calling a function that will create a list and print it: ");

dynamicList();

console.log("\nThan printing that list from main function: ");

caught = false;

try {

console.log(`${cFuncList}\n`);

}

catch (e) {

console.log(`Trying to access same list from main function causes \"${e.message}\" error thus arrays have dynamic allocation\n\n`);

caught = true;

}

if (!caught) {

console.log("No error caught thus JS arrays have static allocation\n\n");

}

1. Are ragged or rectangular multidimensional arrays allowed, or both?

JavaScript supports both ragged or rectangular multidimensional arrays.

Code segment:

console.log("Trying to create rectangular MD array\n");

var mdArr = [[1,2,3],[4,5,6]];

console.log(`mdArr: ${mdArr[0]} - ${mdArr[1]}\n`);

console.log("Thus it is possible to have rectangular multidimensional arrays\n");

console.log("Trying to add another element two latter array inside mdArr (mdArr[1][3] = 7;)");

mdArr[1][3] = 7;

console.log(`mdArr: ${mdArr[0]} - ${mdArr[1]}\n`);

console.log("Thus it is possible to have both ragged and rectangular multidimensional arrays\n\n");

1. Can array objects be initialized?

Arrays can be initialized as they declared or after they declared.

Code segment:

console.log("Arrays can be initialized as they declared or after they declared:\n");

console.log("asArr: var asArr; asArr = [9,9,9]; console.log(asArr);\n");

var asArr;

asArr = [9,9,9];

console.log(`${asArr}\n`);

console.log("\nafterArr: var afterArr = [4,3,6]; console.log(afterArr);\n");

var afterArr = [4,3,6];

console.log(`${afterArr}\n\n`);

1. Are any kind of slices supported?

JavaScript has a slice() function for slicing arrays.

Code segment:

console.log("Slicing with slice function:\n");

console.log(`arr: ${arr}\n`);

console.log(`arr.slice(2,4): ${arr.slice(2,4)}\n\n`);

1. Which operators are provided?

There are assignment (=), element referencing ( [] ) and comparing (==) operators provided for array operations.

Code segment:

console.log("There are three operators provided for array operations: =, [] and ==\n");

console.log(`[n] operator accesses nth element from array => arr[3]: ${arr[3]}\n`);

console.log("== operator returns true if both sides of operator have two references to same array object\n");

var arrb = arr; console.log("arrb: var arrb = arr;\n");

console.log(`arr == arrb : ${arr == arrb}\n\n`);

## PHP

1. What types are legal for subscripts?

Strings and arrays are legal for subscripting.

Code segment:

$s = "This is a string!";

echo "String: $s\n";

echo "First char of string: " . $s[0] . "\n";

echo "Seventh char of string: " . $s[6] . "\n";

print("String is subscriptable\n\n");

$arr = [10,20,30,40,50];

echo "Array: \n" . print\_r($arr,1) . "\n";

echo "First element of array: " . $arr[0] . "\n";

echo "Third element of array: " . $arr[2] . "\n";

echo "Array is subscriptable\n\n";

1. Are subscripting expressions in element references range checked?

PHP utilizes range checking since following code segment raised an error.

Code segment:

echo "Array's length = " . count($arr) . "\n";

echo "Fifth element: " . $arr[4] . "\n\n";

echo "Trying to reach for sixth\n";

try {

echo $arr[5];

}

catch (Exception $e) {

echo "\"" . $e->getMessage() . "\" error caught thus PHP utilizes range checking\n\n";

}

1. When are subscript ranges bound?

PHP uses dynamic range bounding.

Code segment:

$arr2 = [0,1,2,3,4];

echo "Original array:\n";

echo print\_r($arr2,1) . "\n";

echo "Trying to add a sixth item\n";

$arr2[] = 5;

echo "Sixth item: " . $arr2[5] . "\n";

echo "It is possible to add after initialization, thus arrays have dynamic range bounding\n\n";

1. When does allocation take place?

PHP have dynamic array allocation.

Code segment:

function dynamicList() {

$funcArr = [54,45,34];

echo "Array from dynamicList(): " . print\_r($funcArr,1) . "\n";

}

echo "Calling a function that will create a list and print it:\n";

dynamicList();

try {

echo "Than trying to print that array from main function:\n";

echo print\_r($funcArr,1);

}

catch (Exception $e) {

echo "\"" . $e->getMessage() . "\" error raised thus arrays have dynamic allocation\n\n";

}

1. Are ragged or rectangular multidimensional arrays allowed, or both?

PHP has support for both ragged and rectangular multidimensional arrays.

Code segment:

echo "Trying to create rectangular MD array\n";

$mdArr = [[1,2,3],[4,5,6]];

echo print\_r($mdArr,1) . "\n";

echo "Thus it is possible to have rectangular multidimensional arrays\n";

echo "Trying to add another element to the latter array inside mdArr (mdArr[1])";

$mdArr[1][] = 7;

echo print\_r($mdArr,1) . "\n";

echo "Thus it is possible to have both ragged and rectangular multidimensional arrays\n\n";

1. Can array objects be initialized?

PHP arrays can be initialized as they declared or after they declared

Code segment:

echo "Arrays can be initialized as they declared or after they declared:\n";

echo "asArr: \$asArr = [9,9,9]; print\_r(\$asArr);\n";

$asArr = [9,9,9];

print\_r($asArr);

print("afterArr: \$afterArr; \$afterArr = [9,9,9]; print\_r(\$afterArr);\n");

$afterArr; $afterArr = [9,9,9]; print\_r($afterArr); echo "\n\n";

1. Are any kind of slices supported?

PHP has a array\_slice() function for slicing arrays.

Code segment:

echo "Slicing with array\_slice(Array arr, int offset, int length = null) function\n";

echo "Array: " . print\_r($arr, 1) . "\n";

echo "array\_slice(\$arr, 2, 2):\n" . print\_r(array\_slice($arr, 2, 2),1) . "\n";

echo "array\_slice(\$arr, 2):\n" . print\_r(array\_slice($arr, 2),1) . "\n\n";

1. Which operators are provided?

PHP has assignment (=) and comparing (==) operators for array operations.

Code segment:

$arr1 = [1,2,3]; unset($arr2); $arr2 = $arr1; $arr3 = [4,5,6];

echo "arr1:\n" . print\_r($arr1, 1) . "\narr3:\n" . print\_r($arr3, 1) . "\n";

echo "arr1 == arr3: ";

echo ($arr1 == $arr3) ? "true\n" : "false\n";

echo "arr2 = arr1\narr2:\n" . print\_r($arr2, 1) . "\n";

echo "arr1 == arr2: ";

echo ($arr1 == $arr2) ? "true\n" : "false\n";

## Python

1. What types are legal for subscripts?

Arrays from numpy library supports subscripting.

Code segment:

arr = np.array([i\*\*2-3 for i in range(10)])

print(f"arr: {arr}")

print(f"arr[2]: {arr[2]}\narr[5]: {arr[5]}")

print("np.array supports subscripting\n")

1. Are subscripting expressions in element references range checked?

numpy arrays have range checking since following code segment raised an error.

Code segment:

print(f"arr's length = {len(arr)}")

print(f"Tenth element: {arr[9]}")

print("Trying to reach for eleventh")

try:

print(arr[10])

except IndexError:

print("IndexError caught thus nd.arrays have range checking\n")

1. When are subscript ranges bound?

Numpy arrays doesn't have any way to put new items. Therefore numpy array ranges are staticly bound.

Code segment: Says prior statement as well.

1. When does allocation take place?

Python has dynamic array allocation.

Code segment:

def dynamicList():

funcArr = np.array([13,2,35,3])

print(f"Array from dynamicList(): {funcArr}")

print("Calling a function that will create a list and print it:")

dynamicList()

try:

print("Than trying to print that array from main function:")

print(funcArr)

except NameError:

print("NameError caught thus arrays have dynamic allocation\n")

1. Are ragged or rectangular multidimensional arrays allowed, or both?

Numpy supports both ragged and rectangular multidimensional arrays but ragged arrays are deprecated.

Code segment:

print("Trying to create rectangular MD array")

mdArr = np.array([[1,2,3],[4,5,6]])

print(mdArr)

print("Thus it is possible to have rectangular multidimensional arrays")

print("Trying to create ragged MD array")

mdArr = np.array([[1,2,3],[4,5,6,7]], dtype=object)

print(mdArr)

print("Thus it is possible to have both ragged and rectangular multidimensional arrays")

print("But ragged arrays are deprecated in numpy\n")

1. Can array objects be initialized?

Numpy arrays need to be initialized as they are declared.

Code segment:

print("Python doesn't have not-initialized variables\nthus arrays need to be initialized as declared")

print("asArr: asArr = np.array([9,9,9]) print($asArr)\n")

asArr = np.array([9,9,9])

print(asArr)

1. Are any kind of slices supported?

Python has a special slice operator as array\_name[start:end:step].

Code segment:

print("Slicing with arr[start:end]")

print(f"arr: {arr}")

print(f"arr[2:8]: {arr[2:8]}")

print(f"arr[8:2:-1]: {arr[7:1:-1]}\n")

1. Which operators are provided?

Numpy arrays have tons of operators such as +, -, \*, /, %, their assigner forms (+=, -=, …), <, >, ==, =.

Code segment:

arr1 = np.array([1,2,3])

print(f"arr1: {arr1}")

arr2 = np.array([4,5,6])

print(f"arr2: {arr2}")

print(f"arr1+arr2: {arr1+arr2}")

print(f"arr1-arr2: {arr1-arr2}")

print(f"arr1\*arr2: {arr1\*arr2}")

print(f"arr1/arr2: {arr1/arr2}\n")

arr1 += arr2

print(f"arr1 += arr2\narr1: {arr1}")

print("...\n")

print(f"arr1 < arr2: {arr1 < arr2}")

print(f"arr1 % arr2: {arr1 % arr2}")

print("...")

## Rust

1. What types are legal for subscripts?

Rust arrays are legal for subscripting.

Code segment:

let arr = [56,435,7,6,45,58,3];

println!("arr: {:?}", arr);

println!("arr[3]: {}\narr[5]: {}", arr[3], arr[5]);

println!("Rust arrays support subscripting\n");

1. Are subscripting expressions in element references range checked?

Rust arrays have range checking since trying to access beyond range raises errors.

Code segment:

println!("arr's length = {}", arr.len());

println!("Tenth element: {}", arr[6]);

println!("Trying to reach for eighth will give a index out of bounds error\nthus Rust arrays have range checking\n");

// println!("{}", arr[7]); <- Does not compile

// 3. When are subscript ranges bound?

println!("Rust array doesn't have any way to put new items in place\nTherefore numpy array ranges are staticly bound\n");

1. When are subscript ranges bound?

Rust arrays doesn't have any way to put new items in-place. Therefore rust array ranges are statically bound.

Code segment: Prints same statement to the console.

1. When does allocation take place?

Following code segment raises and error thus rust arrays have dynamic allocation.

Code segment:

fn dynamicList() {

let funcArr = [4,6,7,4,7];

println!("Array from dynamicList(): {:?}", funcArr);

}

println!("Calling a function that will create a list and print it:");

dynamicList();

println!("Trying to print that array from main function causes error thus arrays have dynamic allocation\n");

// println!("{:?}", funcArr); <- Does not compile

1. Are ragged or rectangular multidimensional arrays allowed, or both?

Rust supports rectangular multidimensional arrays but doesn’t support ragged ones.

Code segment:

println!("Trying to create rectangular MD array");

let mdArr = [[1,2,3],[4,5,6]];

println!("{:?}", mdArr);

println!("Thus it is possible to have rectangular multidimensional arrays");

println!("Trying to create ragged MD array causes error\nthus ragged multidimensional arrays are not available in Rust\n");

// let mdArr2 = [[1,2,3],[4,5,6,7]]; <- Does not compile

1. Can array objects be initialized?

Arrays can be initialized after they declared or as they are getting declared.

Code segment:

println!("Arrays can be initialized after they declared");

println!("let afterArr: [i32; 5]; afterArr = [45; 5]; println!(\"{{:?}}\", afterArr);");

let afterArr: [i32; 5];

afterArr = [45; 5];

println!("{:?}\n", afterArr);

println!("Arrays can be initialized as they declared");

println!("let asArr: [i32; 5] = [45; 5]; println!(\"{{:?}}\", asArr);");

let asArr: [i32; 5] = [45; 5];

println!("{:?}\n", asArr);

1. Are any kind of slices supported?

Rust has a special operator as &array[start..end] for slicing arrays.

Code segment:

println!("Slicing with &arr[start..end]");

println!("arr: {:?}", arr);

println!("arr[1..4]: {:?}\n", &arr[1..4]);

1. Which operators are provided?

Rust provides assignment (=) and comparing (==) operators for array operations.

let mut arr1 = [1,2,3];

println!("arr1: {:?}", arr1);

let mut arr2 = [4,5,6];

println!("arr2: {:?}", arr2);

println!("arr1 == arr2: {}", arr1 == arr2);

arr2 = arr1;

println!("arr2 = arr1 => arr2: {:?}", arr2);

println!("arr1 == arr2: {}", arr1 == arr2);

println!("== operator returns true if both arrays have the same values");