CS315

Programming Languages

HW1: Arrays in Dart, Javascript, PHP, Python, and Rust

N. Onur VURAL

21902330

SEC: 01

1-What types are legal for subscripts?

Dart: The subscript can only be integer, subscript operator is []. Legal subscript usage is [integerNumber].

EXAMPLE-> myList[2] shows legal usage

Javascript: The subscript can only be integer, subscript operator is []. Legal subscript usage is [integerNumber].

EXAMPLE-> myList[2] shows legal usage

PHP: The subscript can be integer or string, subscript operator is [] or {}. Legal subscript usage is {integerNumber|stringKey}, [integerNumber|stringKey].

EXAMPLE->myList{2}, myList[2], myList['myKey'], myList{'myKey'} all show legal usage

Python: The subscript can only be integer, subscript operator is []. Legal subscript usage is [integerNumber].

EXAMPLE-> myList[2] shows legal usage

Rust: The subscript can only be integer, subscript operator is []. Legal subscript usage is [integerNumber].

EXAMPLE-> myList[2] shows legal usage

2-Are subscripting expressions in element references range checked?

Dart: Yes, trying to use subscript out of bounds produce RT Error.

EXAMPLE-> Reaching myList[**index-out-of-bound**] produces following error, RangeError (index): Index out of range: index should be less than…

Javascript: Checked, but no index-out-of-bound exception is given. If index out bound is assigned to an item, then the array expands.

EXAMPLE-> When tried to be accessed, myList[**index-out-of-bound**] returns: undefined

EXAMPLE-> after (myList[**index-out-of-bound**] = 7), myList[**index-out-of-bound**] will return 7 after this point.

PHP: Yes, trying to use subscript out of bounds produce null value but does not produce error! It simply gives-> PHP Notice: Undefined offset: … in HelloWorld.php on line …

EXAMPLE-> Reaching out of bounds produces null value, myList[**index-out-of-bound**] is: NULL

Python: Yes, trying to use subscript out of bounds produce RT Error.

EXAMPLE-> Reaching myList[**index-out-of-bound**] produces following error, IndexError: list index out of range

Rust: Yes, trying to use subscript out of bounds produce CT Error.

EXAMPLE-> Reaching my_list[**index-out-of-bound**] produces following error, index out of bounds: the len is … but the index is …

3-When are subscript ranges bound?

Dart: Subscript ranges are bound at run-time/ dynamically bound in heap.

EXAMPLE-> For myList, subscript ranges are bound once again, at run-time, when it is reallocted.

Javascript: Subscript ranges are bound at run-time/ dynamically bound in heap.

EXAMPLE-> For myList, subscript ranges are bound once again, at run-time, when it is reallocted.

PHP: Subscript ranges are bound at run-time/ dynamically bound in heap.

EXAMPLE-> For myList, subscript ranges are bound once again, at run-time, when it is reallocted.

Python: Subscript ranges are bound at run-time/ dynamically bound in heap.

EXAMPLE-> For myList, subscript ranges are bound once again, at run-time, when it is reallocted.

Rust: Subscript ranges are bound at compile-time in stack.

let my_list: [i32; 6] = [6, 5, 5, 3, 2, 1];

my_list = [7, 2, 1, 1, 4]; // ERROR: mismatched types expected an array with a fixed size of 6 elements, found one with 5 elements

EXAMPLE-> For my_list, subscript ranges CANNOT bound once again, since subcscript ranges are bound at COMPILE-TIME

4-When does allocation take place?

Dart: Allocation takes place at run-time, in heap.

EXAMPLE-> Same array, myList can be reallocated (See under Q3)

EXAMPLE-> Trying to reach an array before its creation gives error at RUN-TIME

```
print(myProblematicList); // -> Getter not found: 'myProblematicList" );

var myProblematicList = [1,2,3];
```

Javascript: Allocation takes place at run-time, in heap.

EXAMPLE-> Same array, myList can be reallocated (See under Q3)

EXAMPLE-> Trying to reach an array before its creation returns undefined at RUN-TIME

```
document.writeln(myProblematicList); // undefined

var myProblematicList = [1,2,3];
```

PHP: Allocation takes place at run-time, in heap.

EXAMPLE-> Same array, myList can be reallocated (See under Q3)

EXAMPLE-> Trying to reach an array before its creation returns NULL at RUN-TIME

```
print_r($myProblematicList); // returns NULL

$myProblematicList = array(1, 2, 3);
```

Python: Allocation takes place at run-time, in heap.

EXAMPLE-> Same array, myList can be reallocated (See under Q3)

EXAMPLE-> Trying to reach an array before its creation gives error at RUN-TIME"

```
print(myProblematicList) # NameError: name 'myProblematicList' is not defined

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

Rust: Allocation takes place at compile-time, in stack.

EXAMPLE-> Same array, my_list can't be reallocated

EXAMPLE-> Trying to reach an array before its creation gives error at COMPILE-TIME

```
println!("{:?}",my_problematic_list); // error[E0425]: cannot find value `
my_problematic_list` in this scope

let my_problematic_list: [i32; 3] = [0,0,0];
```

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

Dart: Both are allowed!

EXAMPLE-> var rectArray = [[1,2,3],[4,5,6],[7,8,9]];

EXAMPLE-> var raggedArray = [[1,2,3],[4,5],[6,7,8,9]];

Javascript: Both are allowed!

EXAMPLE-> var rectArray = [[1,2,3],[4,5,6],[7,8,9]];

EXAMPLE-> var raggedArray = [[1,2,3],[4,5],[6,7,8,9]];

PHP: Both are allowed!

EXAMPLE-> $rectArray = array(array(1,2,3), array(4,5,6), array(7,8,9));

EXAMPLE-> $raggedArray = array(array(1,2,3),array(4,5),array(6,7,8,9));

Python: Both are allowed!

EXAMPLE-> rectArray = np.array([[1,2,3],[4,5,6],[7,8,9]])

EXAMPLE->raggedArray = np.array([np.array([1,2,3]), np.array([4,5]), np.array([6,7,8,9])])

Rust:

EXAMPLE-> let rect_array: [[i32; 3]; 3] = [[1,2,3],[4,5,6],[7,8,9]]

Basic array structure of Rust does not function to create jagged array essentially but this can be accomplished via external crates like Rust's jagged_array crate

6-Can array objects be initialized?

Dart:

Array objects can be initialized as following:

EXAMPLE-> var myList = [6, 5, 5, 3, 2, 1];

Javascript:

Array objects can be initialized as following:

EXAMPLE-> var myList = [6, 5, 5, 3, 2, 1];

PHP:

Array objects can be initialized as following:

EXAMPLE-> $myList = array(0,1,2,3,4);

Python:

Array objects can be initialized as following:

EXAMPLE-> myList = np.array([0,1,2,3,4,5,6,7,8,9,10])

Rust:

Array objects can be initialized as following:

EXAMPLE-> let my_list: [i32; 3] = [1, 2, 3];

7-Are any kind of slices supported?

Dart: Slices are possible with sublist(start, end-excluded) or sublist(start)

var myList = [6, 5, 5, 3, 2, 1];

EXAMPLE-> var mySlice = myList.sublist(1,3); //[5,5];

Javascript: Slices are possible with ".slice(start, end-excluded)

var myList = [6, 5, 5, 3, 2, 1];

EXAMPLE-> var mySlice = myList.slice(1,3); //[5,5];

PHP: Slices are possible with array_slice(arrayName, start, end)

$myList = array(0,1,2,3,4);

EXAMPLE-> $mySlice = array_slice($myList, 1, 3); // [1,2,3];

Python: Slices are possible with [start: end-excluded]

myList = np.array([0,1,2,3,4,5,6,7,8,9,10])

EXAMPLE-> mySlice = myList[1 : 3]; #[1,2];

Rust: Slices are possible with &arrayName[start..end-excluded]

let my_list: [i32; 6] = [0, 1, 2, 3, 4, 5];

EXAMPLE-> let my_slice = &my_list[1..3]; //[1,2];

8-Which operators are provided?

Dart: Provided operators are: +, ==, !=, [], []=

EXAMPLE-> var ourList = myList + yourList;

EXAMPLE-> var truth = (myList == yourList);

EXAMPLE-> var truth = (myList != yourList);

EXAMPLE-> myList[3] = 5;

Javascript: Provided operators are: [], =, ==, !=, ===, [...arr]

EXAMPLE-> myList[3] = 5;

EXAMPLE-> var truth = (myList == yourList);

EXAMPLE-> var truth = (myList != yourList);

EXAMPLE-> var truth = (myList === yourList);

EXAMPLE-> const arr12Merged = [...arr1, ...arr2];

PHP: Provided operators are: [], =, +, ==, ===, !=, !==, <>

EXAMPLE-> $ourList = $myList + $yourList;

EXAMPLE-> $myList[2] = 5;

EXAMPLE-> var_dump($myList == $yourList);

EXAMPLE-> var_dump($myList === $yourList);

EXAMPLE-> var_dump($myList != $yourList);

EXAMPLE-> var_dump($myList !== $yourList);

EXAMPLE-> var_dump($myList <> $yourList); // same as not equal

Python: Provided operators are: +, ==, [], []=, +=, !=, *, *=, [:], **, @

EXAMPLE-> ourList = myList + yourList

EXAMPLE-> if(myList == yourList)……

EXAMPLE-> if(myList != yourList)……

EXAMPLE-> myList[2] = 5

EXAMPLE-> theList += yourList

EXAMPLE-> print(myList[1:3])

EXAMPLE-> print(numpyArr * 2)

EXAMPLE-> print(numpyArr ** 2)

EXAMPLE-> matrixMultResult = numpyMatrix1 @ numpyMatrix2

Rust: Provided operators are: [], []=, [T; N], [x;N], &, ==, !=

EXAMPLE-> my_list[2] = 5;

EXAMPLE-> let my_list2: [i32; 2] = [1,2];

EXAMPLE-> let arr_c: [i32;10] = [72; 10];

EXAMPLE-> let my_slice2 = &my_list3[..];

EXAMPLE->let truth_equals = arr1 == arr2;

EXAMPLE->let truth_not_equals = arr1 != arr2;

ANALYSIS AND DETAILS

1.Write a paragraph discussing, in your opinion, which language is the best for array operations, for various criteria. Explain why.

When it comes to array operations, Python turns out to be the most efficienct for many aspects overall. First of all, with NumPy array, Python manages to give wide range of crucial operations such as merging arrays, copying arrays, checking equality of two arrays, obtaininig slices easily, printing all values of it directly (without having a loop), making matrix operaitons and many more arithmatic operations, creating multidimesional arrays that can be either rectangular or even ragged. Apart from this, since one of the key characteristics of Python is it being easy to write code, this applies to arrays as well since their declaration and required operations can be done in very short and human-readible lines of code. For example, to display all elements of a ragged array a simple print(myArr) statement is enough whereas a C group language would require a special for loop structure to print out all elements. Another important issue comes out as memory managment where NumPy arrays once again come forward in the competition since there is no need to specify memory allocation and deallocation which can be quite dangerous as memory leaks can occur. Python garbage collector deals this internally so that the user does not need to worry about these issues. However, it must be noted that there is another side of the medallion as there is a trade-off between safe memory control and flexiblility since such manuel memory managment can bring flexibility. Python allows their array sizes to be changed during run-time which come out as more functional than Rust for example, as it makes allocation at compile time. When it comes to allocation, Python is more efficient once again as it does not require to specify the types as in Rust that results in easier code writing. PHP looks more flexible in terms of subscript as it also allows to access elements with string keys however Python makes up to this by supporting heterogenous arrays. Overall Python comes out as more functional in terms of arrays as it: provides the largest number of operations among all five languages, comes out as the one that is easiest to write and read among all five languages and offers flexibility by dynamic bounding in heap.

2.Write a separate section about your learning strategy in doing this homework assignment. A learning strategy is an individual's approach to complete a task. In this section, discuss, in detail, the material and tools you used, experiments you performed. Also talk about personal communication, if you had.

To complete this task, I used wide range of sources and research strategies. First of all, I started my research from the official documentations of each programming language to grasp the basics of each programming language. Even though understanding to write a complete program takes much time and patience, I tried to understand very simple things about them such as getting input, printing to console, declaring a variable and making arithmatic operations. As a following step, I needed to learn how to execute programs written in specified language where I mostly used my previous knowledge coming from the lectures of Mr. Güvenir in CS315. As a support to that, I watched video tutorials that demonstrate execution of programs in those languages [1, 2, 3, 4, 5]. After this, I wanted to test the Dijkstra server by writing simple example test programs that print “Hello World” to the console. I wanted to do this before even writing desired programs because I wanted to make sure that I can run programs on the server without having any problems. I created files having extensions of .dart, .html, .php, .py, .rs by using a text editor, namely Atom. I entered Dijkstra server using Bilkent VPN, uploaded the files via FileZilla and used the console via PUTTY. For Javascript I did the same test, but on browser. After succesfully running my programs, I researched the fundemental properties of arrays for each programming language from both the offical array (or any similar strcutrue if the language did not have arrays) documentations and other online sources such as informative pages and blogs [6 - 24]. While trying to understand the important points of the documentations, I tried to learn by looking example demonstrations and therefore focused on such sources that show how arrays are created, how arithmatic operations are done etc. While writing the example programs, I paid close attention to answer the given questions and show the grader how the program answers those questions. Therefore, I composed my programs in such a way that the grader can clearly see which question is answered with which examples where I print question itself, the answer provided and the example demonstrating it to the console (outputs are on browser and log in Javascript program). While writing these programs, I needed to test whether they are functioning properly and they do not contain any error. For this, I could have used Dijkstra again but since uploading the file again after each change and checking the correctness from PUTTY console takes a considerable amount of time, I used a rather different appoach. I used online compilers instead and it was much easier to experiment with my programs from there and to spot any errors [25 - 29]. When I completed the programs from those online compilers, I wanted to make sure that they were working on Dijkstra, therefore I tested them once again before uploading the file. On Dijkstra I looked at the outputs to verify that the results were in parallel with my expectancy.

Used Resources

Video Links of Execution Tutorials:

Dart: [1] **https://www.youtube.com/watch?v=XoZ01mY-cUg**

Javascript: [2] **https://www.youtube.com/watch?v=821C5aJ3SLM**

PHP: [3] **https://www.youtube.com/watch?v=Z6JKhoMIjHE**

Python: [4] **https://www.youtube.com/watch?v=pFYcAOsNyvs**

Rust: [5] **https://www.youtube.com/watch?v=pFYcAOsNyvs**

Array Information Sources

Dart

[6] https://api.dart.dev/stable/2.14.2/dart-core/List/sublist.html

[7] https://api.dart.dev/stable/2.14.4/dart-core/List-class.html

[8] https://www.educative.io/edpresso/how-to-create-an-array-in-dart

[9] http://blog.sethladd.com/2011/12/lists-and-arrays-in-dart.html

[10] https://medium.com/flutter/flutter-dont-fear-the-garbage-collector-d69b3ff1ca30

Javascript

[11] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array?retiredLocale=tr

[12] http://www.herongyang.com/JavaScript/Array-Acces-Array-Element-with-Index.html

[13] https://www.freecodecamp.org/news/the-javascript-array-handbook/

PHP

[14] https://www.php.net/manual/tr/language.types.array.php

[15] https://www.w3schools.com/php/php_arrays.asp

[16] https://www.w3schools.com/php/php_arrays_multidimensional.asp

[17] https://www.w3schools.com/php/php_operators.asp

Python

[18] https://docs.python.org/3/

[19] https://numpy.org/doc/stable/reference/generated/numpy.array.html

[20] https://www.w3schools.com/python/python_arrays.asp

[21] https://scipy-lectures.org/intro/numpy/operations.html

[22] https://www.pluralsight.com/guides/overview-basic-numpy-operations

Rust

[23] https://doc.rust-lang.org/rust-by-example/primitives/array.html

[24] https://doc.rust-lang.org/std/primitive.array.html

Online Compilers

Dart: [25] https://dartpad.dev/?null_safety=true

Javascript: [26] https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_write

PHP: [27] https://www.w3schools.com/php/phptryit.asp?filename=tryphp_compiler

Python: [28] https://www.programiz.com/python-programming/online-compiler/

Rust: [29] https://www.tutorialspoint.com/compile_rust_online.php