CS 315
Homework 1

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

Zeynep Cankara
ID: 21703381
Section 1

1. What types are legal for subscripts?

- **Dart:** Dart uses [] **(square brackets)** as legal subscript.

```
list1[0];
```

- **Javascript:** [] **(square brackets)** used as legal subscript.

- **PHP:** Both **square brackets** and **curly braces** can be used interchangeably for accessing array elements.

```
$array1[0]
$array1{0}
```

- **Python:** Uses **square brackets** as legal subscript.

```
list1[0]
```

- **Rust:** Uses [] **(square brackets)** as legal subscript.

```
array1[0];
```

2. Are subscripting expressions in element references range checked?

- **Dart:** **Yes**, trying to reference an element outside of length of the list throws `IndexOutOfRangeException` error.

```
var a = listGrowable[listGrowable.length]; // index out of bounds error
```

- **Javascript: Yes,** checked but when a user want to access an index larger than array size the array grows automatically and does not generate an error.

- **PHP: Yes,** PHP arrays are like associative arrays and does not generate an error depending on whether references are within the length of the array.

```
var_dump($array1[100]); // NULL
```

- **Python: Yes**, trying to reference an element outside of length of the list throws "`list index out of range`" error.
- **Rust:** `Compile time error` due to index out of bounds.

```
array1[array1.len()] // Compile time error
```

3. When are subscript ranges bound?

- **Dart:** Subscript ranges are **dynamically bound** at the **run-time**. Supports **heap-dynamic arrays**.
- **Javascript:** Subscript ranges are **dynamically bound** at the **run-time**. Supports **heap-dynamic arrays**.
- **PHP:** Subscript ranges are **dynamically bound** at the **run-time**. Supports **heap-dynamic arrays**.
- **Python:** Subscript ranges are **dynamically bound** at the **run-time**. Supports **heap-dynamic arrays**.
- **Rust:** Subscript ranges are bound at the **compile-time** and they are **stack allocated arrays**.

4. When does allocation take place?

- **Dart:** Allocation takes place in the heap at **run time.**
- **Javascript:** Allocation takes place in the heap at **run time.**
- **PHP:** Allocation takes place in the heap at **run time.**

- **Python:** Allocation takes place in the heap at **run time.**
- **Rust:** Allocation takes place in the heap at **compile time.**

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

- **Dart:** Allows **both rectangular multidimensional** and **ragged** arrays.

```
// multidimensional list
List<List<int>> multiDimList = [[1,2], [2,4]];
// ragged list
List<List<int>> raggedList = [[1,2,3], [2,4]];
```

- **Javascript:** Allows **array of arrays which simulates both rectangular multidimensional** and **ragged** arrays.

```
var multiDimArray = [
        [ ],
        [ ],
        [ ] ]

var raggedArray = [
        [ [ ] ], [ [ ] ],
        [ ],
        [ ] ];
```

- **PHP:** Allows **both rectangular multidimensional** and **ragged** arrays.

```
$multi_dim_arr = array ( array ( 1, 2),
        array ( 2,4 ));

$ragged_arr = array ( array ( 1, 2, 3),
        array ( 2,4 ));
```

- **Python:** Allows **array of arrays which simulates both rectangular multidimensional** and **ragged** arrays.

```
multi_dim_list = [[1,2], [2,4]]
```

```
ragged_list = [[1,2,3], [2,4]]
```

- **Rust:** Allows **both rectangular multidimensional** and **ragged** arrays.

```
let mut multi_dim_arr = [[0u8; 2]; 2];
let ragged_arr = [[0u8; 3]; 2];
```

6. What is the maximum number of subscripts?

- **Dart:** It depends on the compiler specifications.
- **Javascript:** The exact maximum limit of an array size thus maximum number of subscripts is `2^32 - 1`, because of the Javascript memory restrictions.
- **PHP:** Limit to the **amount of memory your script can use** which can **changed in the 'memory_limit'** in php configuration.
- **Python:** It depends the **memory of the PC** and whether **memory allocation** is sustainable for a given dimension.
- **Rust:** Determined by the *actual* size in memory that the OS will reserve for the array.

7. Can array objects be initialized?

- **Dart:** Yes, they can initialized.

```
// fixed length array
List<int> listFixedLength= new List(5);
// dynamic size  array
List<int> listGrowable = [1, 2, 3];
```

- **Javascript:** Yes, they can initialized.

```
var list1 = [1, 2, 3];
```

- **PHP:** Yes, they can initialized.

```
$array2 = array(1, 2, 3, 4, 5);
```

- **Python:** Yes, they can initialized.

```
list_growable = [1, 2, 3, 4, 5]
```

- **Rust:** Yes, they can initialized.

```
let array2: [i32; 5] = [1, 2, 3, 4, 5];
```

8. Are any kind of slices supported?

- **Dart: Yes,** arrays (lists) supports **method sublist** for slices.

```
listGrowable = [1, 2, 3, 4, 5];
List<int> slice = listGrowable.sublist(2,4); // [3, 4]
```

- **Javascript: Yes** via **slice(start index, end index)** method.

```
var arr = [1, 2, 3, 4, 5];
var slice = arr.slice(2,4); // [3, 4]
```

- **PHP: Yes,** arrays supports **method array_slice** for slices.

```
$arr = array(1, 2, 3, 4, 5);
$slice = array_slice($arr, 2, 4); // Array ( [0] => 3 [1] => 4 [2] => 5 )
```

- **Python: Yes**, Python arrays (lists) can sliced **via square brackets once start and end index specified**.

```
list_growable = [1, 2, 3, 4, 5]
slice = list_growable[2:4] # [3, 4]
```

- **Rust: Yes,** Provides a way to look into a block of memory represented as a pointer and a length. Slices have a dynamic size unlike arrays and do not coerce to arrays

```
let mut arr = [3, 4];
let slice = &mut arr[..]; // [3, 4]
```

9. Which operators are provided?

- **Dart:** Supports **"+"," ==", "[]" and "[]="** operators for array operations.

```
List<int> l1 = [1];
List<int> l2 = [2, 3];
List<int> l3 = l1 + l2;
print(l1 == l2);
listFixedLength[0] = 120; // fixedLengthList[0] == 120
var item1 = listFixedLength[0]; // item1 == 120
```

- **Javascript:** Supports **"[]", "[]="** operators.

```
arr[0] = 12; // []=
var item1 = arr[3]; // []
```

- **PHP:** Supports **"+"," ==", "!=" , "===", "!=="and "<>"** operators.

```
// + (Union)
$x = array('key' => 12);
$y = array('value' => 7);
$z = $x + $y;
// == (Equal)
$x = array("key" => 21);
$y = array("key" => "21");
var_dump($x == $y);
// === (Identical)
var_dump($x === $y);
// !=, <> (Not Equal)
var_dump($x != $y);
```

```
var_dump($x <> $y);
// !== (Non Identical)
var_dump($x !== $y);
```

- **Python:** Supports **“[]”,” +“, “+=” , “==”, “[:]”and “[]=“, “*”, “*=”, “!=”** operators.

```
l1 = [1]
l2 = [2, 3] # []
l3 = l1 + l2 # +
l1 += l3  # +=
l1 == l3  # ==
l1 != l3 # !=
l1 *= 2 # *=
l1 = l1 * 2 # *
l3[0] = 23 # []=
```

- **Rust:** Supports **“[]”, “[T;N]”and “[]=“, “&[]”** operators.

```
multi_dim_arr[0] = [1, 2]; // []=
let rust_arr = [0u8; 32]; // []
let val = &rust_arr[..]; //&[]
```

10. Are associative arrays available?

- **Dart:** **Map objects** in Dart considered as **associative arrays**.

```
var assocArray = {'CS315': "Programming Languages", 'CS224':'Computer Organization'};
```

- **Javascript:** **No,** Arrays with named indexes are called associative arrays (or hashes). Javascript **does not support associative arrays**.

- **PHP:** Has associative array structures.

```
$assoc_array = array("CS315"=>"Programming Languages",
"CS224"=>"Computer Organization");
```

- **Python: Dictionaries** are associative arrays in Python.

```
assoc_array = {'CS315': 'Programming Languages', 'CS224': 'Computer
Organization'}
```

- **Rust:**

```
let mut assoc_array = HashMap::new();
assoc_array.insert("CS315","Programming Languages");
assoc_array.insert("CS224","Computer Organization");
```

Follow-up Questions:

Question 1:

After doing the homework, I decided that I would prefer the array structures of Python language because arrays in Python are both comfortable to use and have high writability/readability. First, Python arrays are heap dynamic therefore memory management doesn't have to controlled by the user which increases the flexibility of the language with an efficiency trade-off where compiled languages such as Rust are more advantageous. The arrays of the Python language can grow and shrink during program execution as a result of the dynamic subscript binding and storage allocation which again expresses flexibility aspect of the Python language. Python allows slices together with multidimensional and ragged arrays when expressive multidimensional array structures being used together with various expressive array operations such as concatenation and element membership operations. Since executing diversified range of array operations is relatively easier than other languages discussed in the homework, to me Python is more suitable to do data science and scientific computing. Moreover, Python lists are heterogeneous and supporting various data types as

well as objects which in turn increases the flexibility of the language more. Also, type of a variable determined at run-time without the need of explicit declaration of type increasing the writability of the language with a trade-off of readability.

Question 2:

Overall, my first resource doing the assignment was the language documentations of the given programming languages. I investigated array sections of each language together with array examples given in that language. Whenever I couldn't reach specific information by looking at documentation, I searched sites such as StackOverflow and Tutorials-point, Geeksforgeeks for further explanation. I only took account of information with references citing reliable resources about that specific language. Additionally, I used online compilers to experiment with the languages given in the homework to verify the knowledge that I acquired via online language documentations. I pay attention to using the online compilers provided by the official site of the language documentation (such as https://play.rust-lang.org/ for Rust. and https://dartpad.dev/ for Dart) or the ones provided by the Dijkstra machine at Bilkent's servers. Some questions such as questions in the homework related with binding and storage allocation which concerned more with the compilers and interpreters. In those questions, I tried to understand the role of compiler or interpreter when binding subscriipts and allocating memory. After I finished reviewing language documentation, I visited StackOverflow to check whether there are some corner cases or uncommon use cases left when using arrays in the specified language regarding the questions given in the homework. The question that I struggled most throughout the homework was "maximum number of subscripts" since given that the programming language supports multidimensional arrays most of the time answer of the question depends to the memory allocation reserved for that specific programming language. Section below I documented my own observations and struggles that I encounter learning each of these languages.

Dart:

- Good documentation, together with a tour to the Dart language explaining the language properties: https://dart.dev/guides/libraries/library-tour.
- Used online compiler provided by Dart's official website: https://dartpad.dev/

- Concise guide about Dart lists and arrays: http://blog.sethladd.com/2011/12/lists-and-arrays-in-dart.html.
- Core library documentation about Lists in Dart: https://api.dart.dev/stable/2.7.2/dart-core/List-class.html

Javascript:

- Good site for learning about web based languages: https://www.w3schools.com/js/js_arrays.asp
- Another website for language specific tutorials: https://www.tutorialrepublic.com/javascript-tutorial/javascript-arrays.php
- Mozilla's own documentation for Javascript: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

PHP:

- Offical documentation of the PHP language: https://www.php.net/manual/en/language.types.array.php
- Good site for learning about web based languages: https://www.w3schools.com/php/

Python:

- Google's documentation of the Python language: https://developers.google.com/edu/python/lists
- I used Dijkstra's interpreter to run Python scripts.
- Offical Python documentation: https://docs.python.org/3/tutorial/datastructures.html

Rust:

- Offical documentation of the Rust language array structures:https://doc.rust-lang.org/std/primitive.array.html
- Online compiler provided by the Rust's offical site: https://play.rust-lang.org/
- Taking slice in Rust language: https://stackoverflow.com/questions/25428920/how-to-get-a-slice-as-an-array-in-rust
- Multidimensional arrays in Rust: https://stackoverflow.com/questions/13212212/creating-two-dimensional-arrays-in-rust