CS315 HW1

Mustafa Efe Tamyapar
ID:21902856
Section:1

DART

1)Dart arrays allow integers as subscript types.

```
var arr1 = ["tamyapar","mustafa","efe"];
var raggedArr = [[1],[2,3]];
String surname = arr1[0];
```

2)Range check is done at runtime in dart. Range is not checked in compile time.

```
print(arr2[4]); //Gives an error due to range of the array being 0
```

```
RangeError (index): Invalid value: Only valid value is 0: 4
#0      List.[] (dart:core-patch/growable_array.dart:147:60)
#1      main (file:///home/cs/efe.tamyapar/315HW1/exmp.dart:19:13)
#2      _startIsolate.<anonymous closure> (dart:isolate-patch/isolate_patch.dart:305:19)
#3      _RawReceivePortImpl._handleMessage (dart:isolate-patch/isolate_patch.dart:172:12)
```

3)Range is bounded in compile time. We can get the range of an array using .length .

4)In Dart, allocation takes place at compile time. Because when I try to call an array which is not allocated, the program does not compile.

String me = arr4[0];

```
exmp.dart:28:15: Error: Getter not found: 'arr4'.
  String me = arr4[0];
              ^^^^
```

5)In Dart both rectangular and ragged arrays are allowed.

```
  stdout.write("Multidimensional arrays: ");
  var multi = List.generate(3, (i) => List.generate(3, (j) => i + j));
  print("Rectangular : ");
  print(multi);

  stdout.write("Ragged : ");
  var ragged = [[1],[2,3]];
  print(ragged);
```

6)In Dart, array objects can be initialized.

```
  var album = ["Hallo",1, "Easy",2)]; //array of objects
  stdout.write("Objects of the album array: ");
  print(album[0]);
  print(album[2];
```

7)In Dart, the sublist method is used for slices. sublist(int 1 , int 2) method returns the elements from the int1 index until the int2 index. int2 index not included. And in the sublist(int3) method, this method returns all the elements starting from the int3 index until the end.

```
 var kitchen = ["k nife", "spoon", "fork", "cooker", "mixer"];
 print(colors.sublist(0, 3)); // [knife, spoon, fork]
 print(colors.sublist(3)); // [cooker, mixer]
```

8)In Dart these operations are provided for arrays: +, [], []=, ==.

```
  var arr3 = [1];
  var arr4 = [2, 2];
  var arr5 = arr3 + arr4;//+
  print(arr5);

  var arr7 = [1];
  arr7[0] = 120; // []
  print(arr7);

  var elmt = arr7[0]; // []=
  print(elmt);

  print(arr5 == arr7);//==
```

Javascript

1)In javascript, integer is the only type allowed for subscripting. Other types will return undefined.

```
var me = ["mustafa", "efe","tamyapar"];
```

2)Javascript arrays checks range of the array.

```
var arr2 = [9,8,7,6,5,4];
arr[10]; //Does not gives a error but says undefined
```

```
First element: undefined                                    array.html:22
```

3)In Javascript, range checks is made in runtime. Because javascript uses heap dynamic array.Also javascript will not return garbage value even though when it is out of bound. It is an other indicator that range check is made in run time.

4)In javascript, allocation takes place in run time because javascript arrays are heap - dynamic.

5)Javascript allows both recntangular and ragged multidimensional arrays.

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

```
console.log("Ragged Array example ");

for(let i = 0; i < arrRagged.length; i++){
  console.log(arrRagged[i]);
}

var Movies = [
    ["Harry Potter 1", 5],
    ["Harry Potter 2", 4],
    ["Harry Potter 3", 3],
    ["Harry Potter 4", 5],
    ["Harry Potter 5", 1]
];
```

6)Object arrays can be initialized in javascript arrays.

```
const personal = {Name:"Mustafa", lastName:"Tamyapar", number:123456};
console.log("Object example ");
var objArr = [personal];
console.log(objArr[0]);
```

7)Javascript arrays support slices

```
console.log("whole array: ");
var colors = ["Black", "Grey", "Purple", "Yellow", "Pink"];
for(let i = 0; i < colors.length; i++){
  console.log(colors[i]);
}
console.log("Slice: ");
var darkerColors = colors.slice(0, 2); // ["Black", "Grey"]
```

8)In javascript all the arithmetic, relational operators are allowed for arrays.

```
var opArr1 = [1,2,3,4,5];
var opArr2 = [2,4,6,8,10];
var x = opArr1[0] + opArr2[0]; //+
var y = opArr2[4] % 5 //%
var z = opArr1[1] - opArr2[2] //-

for (x=5; x>2; x--) //>
  console.log("x="+x);

if( opArr1 != opArr2 )
    console.log('Wow not equal'); //!=
```

PHP

1)Php arrays allow integer, float, string, boolean, character, null for subscripts.

```
$arr = array("Hello", "From","the","Other","Side");
print_r($arr);
```

2)Range check is made in run time in PHP. Range is not checked in compile time

echo $arr[5], "
"; //gives error due to arrays boun being 4

```
Accessing elements :<br>Hello<br>From<br>PHP Notice:  Undefined offset: 5 in /home/cs/efe.tamyapar/315HW1/exmp.php on line 14
```

3)Range is bound in run time. Because when I try to get an element at a range which is out of bound, my file compiles but gives an error when I run the code.

4)In PHP allocation takes place in runtime. Because when I try to allocate array which is not initialized it gives a run time error.

```
Accessing elements :<br>Hello<br>From<br>PHP Notice:  Undefined variable: arr2 in /home/cs/efe.tamyapar/315HW1/exmp.php on line 18
```

5)PHP language allows ragged multidimensional arrays but does not allows rectangular multidimensional arrays.

```
$friends = array (
    array("Name" => "Mustafa", "Age" => 20),
    array("Name" => "Onat", "Age" => 21)
);
```

6)PHP allows object initialization.

```
$meal = array("Red lentil soup", "meatballs", "pasta", "yoghurt","grapes",
"wine");
echo "Soup of the day is: " . $meal[0] . " <br>";
```

7)PHP supports slices. Method array_slice($arr,starting index, end index) is used in php.

```
$animals = array("cat","dog","fish","whale");
print_r(array_slice($animals,0,2));//[0] => cat [1] => dog
```

8)PHP arrays provide + (union) , == (equality) , === (identity) , != (non equality) !== (non identity), <> (non equality) operators.

```
$sum = array(1);
$sum2 = array(9, 9);
$sum3 = $sum + $sum2;//+(union)
echo "+ operation <br>";
print_r($sum3);//[0] => 1 [1] => 9

echo "== (equality)operation <br>";
var_dump($sum2 == $sum3);// ==(equality)

echo "=== (identity)operation <br>";
var_dump($sum2 == $sum3);// ===(identity)

echo "!= (non equality)operation <br>";
var_dump($sum2 != $sum3);//!=(non equality)

echo "!== (non identity)operation <br>";
var_dump($sum2 !== $sum3);//!==(non identity)

echo "<>(non equality)operation <br>";
var_dump($sum2 <> $sum3);//<>(non equality)
```

Python

1)Python language allows integer, float,double, boolean and char for subscripts.

```
city = np.array([35, 6 , 34, 9, 48])

 #Accessing elements in Python arrays.
print("The first element is: ")
print(city[0])
print("The last element is: ")
print(city[-1])
```

2)Range check is made int he run time in python. Range is not checked in compile time.

```
print(city[10])
```

```
    print(city[10]); #Gives an error due to range of the array being 4.
IndexError: index 10 is out of bounds for axis 0 with size 5
```

3)In python arrays, range is checked in run time. Because as in the previous example, when I try to get the element which is out of bound, my code compiles but does not run.

4)Python language allocates arrays at runtime because when i try to call an array which is not created. My code compiles but gives a run time error.

```
Traceback (most recent call last):
  File "exmp.py", line 20, in <module>
    alloca = np.arr4[0]
AttributeError: module 'numpy' has no attribute 'arr4'
```

5)In python rectangular and ragged arrays are both allowed.

```
print("Rectangular multidimensional array example: ")
rectangular = np.array([[1,1,1,1],[2,2,2,2],[3,3,3,3],[4,4,4,4]])
print("Ragged multidimensional array example: ")
ragged = np.array([[1],[2,2],[3,3,3],[4,4,4,4]],dtype=object)
```

6)In python array objects can be created.

```
elt = np.array(['b', 3, 3.5])
```

7)Python supports slices.

```
print("Slice example:")
sauce = np.array(['k','m','s','h','b'])
print(sauce)
print(sauce[0:2])# [k,m]
print(sauce[3:])#[h,b]
print(sauce[:2])#[k,m]
print(sauce[:] )#Copies whole array
```

8)Python arrays allow + ,-, * , / for 2 arrays (2 same shaped arrays) and adding integer to all elements, ,==,!=, [],=[],>=(relational operations), matrix multiplication(@) operations in array.

```
arrOp= np.array([1])
arrOp2 = np.array([2,20,30,14,28])
arrOp3 = np.array([9,3,5,2,10])
```

```
arrOp4 = arrOp3 + arrOp2 #+(shapes of the arrays should be same)
print("+ operation with 2 arrays ")
print(arrOp4)

arrOp8 = arrOp3 + 2 #+
print("+ operation with integer ")
print(arrOp8)

arrOp11 = arrOp3 - 2 #-
print("- operation with integer ")
print(arrOp11)

arrOp6 = arrOp3 * 2 #*
print("* operation with integer ")
print(arrOp6)

arrOp7 = arrOp3 * arrOp2 #*(2 arr)
print("* operation with 2 arr ")
print(arrOp7)

arrOp9 = arrOp3 / arrOp2 #*(2 arr)
print("/ operation ")
print(arrOp9)

print("== operation ")
print(arrOp4 == arrOp3) #==

print("!= operation ")
print(arrOp4 != arrOp3) #!=

arrOp30 = np.array([1]) #[]
print("[] operation ")
print(arrOp30)

arrOp30[0] = 23 #=[]
print("=[] operation ")
print(arrOp30)

arrOp10 = arrOp2 >= 20 #>=
print(">= operation ")
print(arrOp30)

multi_dim1 = np.array([[3,2],[0,1]])
multi_dim2 = np.array([[3,1],[2,1]])

print("@(matrix multip) operation ")
print(multi_dim1 @ multi_dim2) #matrix multiplication @
```

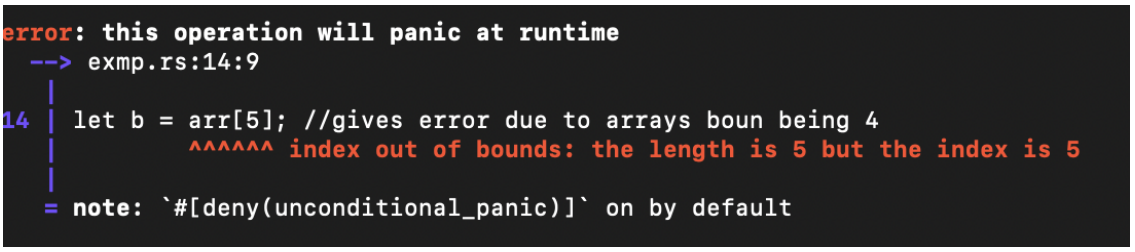
Rust

1)In rust language, allowed subscript types are:integer,boolean,charachter and string.

```
let arr: [i32; 5] = [1, 2, 3, 4, 5];
let names:[String; 4] =
["Beliz".to_string(),"Meleknaz".to_string(),"Doğa".to_string(),"Bengisu".to_s
tring()];
```

2)Yes bounds are checked in rust.Also in rust, arrays have fixed size. We initiate while creating the array.

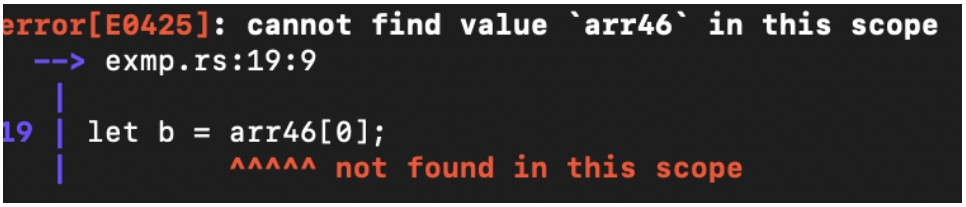
let b = arr[5]; //gives error due to arrays boun being 4



3)Bound check is made in compile time in rust. As we initiate bounds while creating, array can not exceed that bound.

4)In rust, allocation takes place in compile time. Because when i try to call an element from an array which is not created my program gives compile time error.

```
let b = arr46[0];
```



5)Rust allows rectangular multidimensional arrays but does not allows ragged multidimensional arrays.

```
let mut rectangular = [[0u8; 4]; 3];
rectangular[0] = [72, 81 ,63,49];
```

6)Array objects can be initialized in rust arrays.

```
let grades:[i32; 4] = [30,40,25,20];
let g = grades[0];
println!("First grade is: {}", g);
```

7)Rust supports slices.

```
let whole: [i32; 4] = [11, 22, 33, 44];

let slice: &[i32] = &whole;
println!("Whole: {:?}", slice);//Returns whole array

let slice2 = &whole[0..2];//[11,22]

let slice3: &[i32] = &whole[0..3];,//[11,22,33]
```

8)In Rust language, [](asssgnment) , []=(getting an element) ,&[](copying the array) operations are provided.

```
let opArr =[i32; 4]; // []
println!("[] operation {}");
print(opArr;

let opArr2: [i32; 5] = [4, 8, 16, 32, 64];
let elmt = opArr2[0]; // []=
println!("[]= operation {}");
print(elmt);

let result = &opArr2[..];// &[]
println!("&[] operation {}");
println!("Result is {:?}", result);
```

Which language is the best for array operations?

Among the five languages which were given I think that python arrays is the one which is the most useful for operations.I choosed python because , firstly python is the language the most readable and writable among these languages. Secondly, python arrays are heap dynamic and i found heap dynamic arrays advantageous. Heap dynamic arrays are better because their size can change after the creation. This makes developers life easier for huge datasets. Also in python numpy library, as we worked on , there are lots of operators and methods to use for which is also a advantageous for complex programs and calculations which can be done using the python arrays.Moreover, python has specific methods to use for data science and mathematical operations hence using python arrays for complex mathematical analysis and applications will be useful for the developer. In the another aspect, we know that python is heap dynamic and range check, allocation are made in the run time. I think this situation makes developer's work easier because when I write a program if there is a bug in the code, other parts of the code will work as they should be until the line where the bug is which make the software development process smoother compared to compile time checks.

My learning Strategy

Firstly, when ı started to work on this assignment I looked at to the professor Guvenir's dijkstra repository. There were array examples and other examples on each language which is given in this assignment. I practised and viewed these repositories. After viewing all the repositories I started with dart language to work on. Firstly I looked at the array documentation while working for that language. For the references and for getting the maximum efficiency I try look at official documentations for the each language. I tried the examples which were given in the official documentation. In the most of the languages official documentation was clear and easy to understand but there were no answers for some of the questions which were given in the assignment. When I could not get my answer from documentation of the language, I searched on google for answers but most of the time there were redundant answers which were not clear. To get away from that I looked up to stackoverflow, Tutorials-point, w3schools sites. There were answers to questions with examples which made me understand language in a comprehensive way. In these answers questions were from the developers so in each answer there were code snippets which helped

me the learn the syntax of the language as well as the topic which i was trying to learn. The research process differed for each language. Python had the most resources to get my answers fast and easiest one to learn for me. Documentation of the python was the most comprehensive and clear one and also there were lots of questions on the stackoverflow etc. on the topics which I was trying to get answer. On the opposite hand I found that rust has the least sources for learning the language. Even though documentation was clear there were not much examples to understand the syntax easily. Ahead from that there were not many questions on stackoverflow etc to get my answer and understand the topic easily.When I look at the research process by language wise . I struggled at 4th question the most, which is about where array allocation takes place. In this question my strategy is calling arrays which are not created in the program. By that strategy i could see where the error was given and hence where the allocation takes place. Also I struggled with 5th question which is on multidimensional arrays. Most of the language's official documentation did not mentioned ragged multidimensional arrays or did not give an example on this. As a solution for that I used the sites which i mentioned before.After my researches for questions, I write my code on visual studio code than using filezilla , I connected to dijkstra server and tested my files on dijkstra server except the javascript one. I used google chrome to test .html files . After creating html files I opened up to html file and from the web page I inspected and looked to the console. After the test I added necessary comments to my files and enhanced the readability of my files.

References

5. Data Structures — Python 3.10.0 documentation. (2021). Python Documentation.

https://docs.python.org/3/tutorial/datastructures.html

array - Rust. (2021). Rust. https://doc.rust-lang.org/std/primitive.array.html

Dart - How to initialize a jagged array? (2018, December 23). Stack Overflow.

https://stackoverflow.com/questions/53901575/dart-how-to-initialize-a-jagged-array

Data types — NumPy v1.21 Manual. (2021). Numpy.

https://numpy.org/doc/stable/user/basics.types.html

Data Types - The Rust Programming Language. (2021). MIT Rust.

https://web.mit.edu/rust-lang_v1.25/arch/amd64_ubuntu1404/share/doc/rust/html/book/second-edition/ch03-02-data-types.html

Expressions and operators - JavaScript | MDN. (2021, September 24). Javascript Operators.
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators

Is it possible to make an Array of Objects in Dart/Flutter. (2019, November 29). Stack Overflow.
https://stackoverflow.com/questions/59108853/is-it-possible-to-make-an-array-of-objects-in-dart-flutter

JavaScript Array slice() Method. (2021). Javascript Documentation.
https://www.w3schools.com/jsref/jsref_slice_array.asp

JavaScript Operators. (2021). Javascript Documentation.
https://www.w3schools.com/js/js_operators.asp

Kawar, D. (2020, February 3). *Working with multi-dimensional List in Dart - Flutter Community*. Medium.
https://medium.com/flutter-community/working-with-multi-dimensional-list-in-dart-78ff332430a

Multi-dimensional lists in Python. (2021). Python.
https://www.tutorialspoint.com/multi-dimensional-lists-in-python

Oiku, J. (2020, April 11). *Jagged Array in JavaScript*. DEV Community.
https://dev.to/osejudith/jagged-array-in-javascript-18og

PHP: array - Manual. (2021). PHP Documentation.
https://www.php.net/manual/en/function.array

PHP: array_slice() function. (2020a, February 26). W3resource.
https://www.w3resource.com/php/function-reference/array_slice.php

PHP: array_slice() function. (2020b, February 26). W3resource.
https://www.w3resource.com/php/function-reference/array_slice.php

PHP Data Types. (2021). PHP. https://www.w3schools.com/php/php_datatypes.asp

Rust Arrays Tutorial | KoderHQ. (2021). Koderhq. https://www.koderhq.com/tutorial/rust/array/#access-indexer

Srinivasulu, R. (2018, September 25). *Overview of Basic Numpy Operations*. Pluralsight. https://www.pluralsight.com/guides/overview-basic-numpy-operations

std::ops - Rust. (2021). Rust Documentation. https://doc.rust-lang.org/std/ops/index.html

sublist method - List class - dart:core library - Dart API. (2021). Api.Dart.Dev. https://api.dart.dev/stable/2.14.2/dart-core/List/sublist.html

A tour of the Dart language. (2021). Dart.Dev. https://dart.dev/guides/language/language-tour

Understanding slice notation. (2009, February 3). Stack Overflow. https://stackoverflow.com/questions/509211/understanding-slice-notation