DART

Questions:

1. What types are legal for subscripts?

Constant and non-constant variables, final variables, integer lists.

```
var melis = const [];

final flower = const [];

melis = [8, 8, 8];

var tree = [];

tree = [42];
```

2. Are subscripting expressions in element references range checked?

Yes, they are checked.

```
var butterfly = [12];

//print(butterfly[13]); //this gives an error because butterfly list accepts only 12
                         //integers, trying to reach the 13th one therefore gives an
                        //error
```

3. When are subscript ranges bound?

During compile-time (static).

4. When does allocation take place?

It takes place during compile-time.

```
//blossom[0]; //trying to reach variable blossom before creating it gives an error
              //because it is not yet created

var blossom = const [];

blossom = [1, 2, 3, 4, 5, 6, 7, 8, 9]; //variable blossom is initialized here
```

```
print(blossom[0]); //this line will not cause any error because blossom is
//created above
```

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

Both arrays are allowed in Dart.

```
int myNum = 25;

//multidimensional arrays

int row = 2;

int col = 4;

var myList = List.generate(row, (i) => List.filled(col, i)); //generating the list
                                                              //and then filling it in

myList[0][1] = myNum;

print(myList);


//ragged arrays

var matrix = List.generate(4, (_) => List.filled(4, 0));

print(matrix);
```

6. Can array objects be initialized?

In Dart, arrays are list objects. So yes, they can be initialized.

```
//a list of integer objects

List<int> thisList = List.filled(1, 0, growable: true);

int deneme = 8;

thisList[0] = deneme;
```

print(thisList);

7. Are any kind of slices supported?

Yes, slicing of strings is supported with the method substring().

String myString = "123456789";

myString = myString.substring(3); //the output string will start from 4...9, 123 //is sliced

print(myString);

8. Which operators are provided?

Dart supports the following operators:

- arithmetic operators:

  +, -, -expr, *, /, ~/, %

assert(1 + 2 == 3);

assert(8 - 3 == 5);

assert(2 - 4 == -2);

assert(2 * 2 == 4);

assert(9 / 2 == 4.5); //returns a double

assert(9 ~/ 2 == 4); //returns an integer

assert(5 % 4 == 1); //returns remainder

- increment and decrement operators:

  ++var, --var, var++, var--

int c = 2;

++c;

```
int m = 8;

--m;

print(c);

print(m);


int n = 3;

n++;

int i = 4;

I--;

print(n);

print(i);
```

- equality and relational operators:

  ==, !=, >, <, >=, <=

```
var flag = 5;

var map = 4;

if (map < flag) {

        print("Success.");

}
```

- type test operators:

  as, is, is!

- assignment operators:

=, *=, %=, >>>=, ^=, +=, /=, <<=, &=, |=, -=, ~/=, >>=

```
var table = 5;

table *= 3;

assert(table == 15);

print(table);


var sky = 8;

sky %= 3;

assert(sky == 2);

print(sky);


var ocean = 4;

ocean += 1;

assert(ocean == 5);

print(ocean);


double wind = 10;

wind /= 5;

assert(wind == 2.0);

print(wind);
```

```
var daisy = 5;
daisy -= 5;
assert(daisy == 0);
print(daisy);


var water = 9;
water ~/= 4;
print(water);
```

- logical operators:

  !expr, ||, &&

- bitwise and shift operators

  &, |, ^, ~expr, <<, >>, >>>

```
final smoothie = 0x12;
final oatmeal = 0x0a;
assert((smoothie & oatmeal) == 0x01);
assert((smoothie & ~oatmeal) == 0x03);
assert((smoothie | oatmeal) == 0x0b);
assert((smoothie ^ oatmeal) == 0x0c);
assert((smoothie << 4) == 0x120);
assert((smoothie >> 4) == 0x02);
print(smoothie);
```

print(oatmeal);

}

- conditional expressions:

  condition ? expr1 : expr2

JAVASCRIPT

Questions:

1. What types are legal for subscripts?

   Strings, integers.

const names = [“Melis”, “Cagan”, “Turkey”];

2. Are subscripting expressions in element references range checked?

   Yes, they are checked.

console.log(names[4]); //this gives an error because names array has only 3 //times, therefore trying to reach the 3rd item gives an error

3. When are subscript ranges bound?

   During run-time (heap-dynamic).

4. When does allocation take place?

   It takes place during run-time.

//myArr[8]; //since myArr is not yet defined, this line gives an error

const myArr = [];

myArr[0] = 1;

myArr[1] = 2;

myArr[2] = 2;

console.log(myArr[0]);

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

Rectangular multidimensional arrays are allowed.

```
//multidimensional array

const myArray = [[9, 8, 7], [6, 5, 4], [3, 2, 1]];

console.log(myArray[0][1]);

//jagged array

let mySecondArr = [[3, 2], [1, 4, 5], [6, 7, 8, 9]];

let total = 0;

let average = 0.0;

for (let row = 0; row < mySecondArr.length; ++row) {

  for (let col = 0; col < mySecondArr[row].length; ++col) {

     total += grades[row][col];

  }

  average = total / mySecondArr[row].length;

  console.log("Student " + parseInt(row + 1) + " average: " +
average.toFixed(2));

  total = 0;

  average = 0.0;
}
```

6. Can array objects be initialized?

Yes, they can be initialized.

```
let myFruits = [
{
   "color": "orange",
   "type": "tangerine",
   "number": 1
 },
 {
   "color": "green",
   "type": "apple",
   "number": 1
 },
]
```

7. Are any kind of slices supported?

Yes, the slice() method in Javascript returns a shallow copy of a portion of an array. This portion is put into a new array from start to end without end included (start and end represents index items).

```
const trialArr = [1, 2, 3, 4, 5];
console.log(trialArr.slice(1));
```

console.log(trialArr.slice(2));

8. Which operators are provided?

Javascript supports the following operators:

- arithmetic operators:

  +, -, *, /, %, ++, --

let x = 100 + 50;

console.log(x);

let y = 20 - 2;

console.log(y);

let z = 2 * 2;

console.log(z);

let m = 10 / 5;

console.log(m);

let n = 10 % 8;

console.log(n);

++m;

console.log(m);

--n;

console.log(n);

- comparison operators:

  ==, ===, !=, >, <, >=, <=

let a = 1;

let b = 1;

let c = 2;

console.log(a == b); //true

console.log(a != b); //false

console.log(a === ‘1’); //false

console.log(c > a); //true

console.log(a >= b); //true

console.log(c <= a); //false

- logical operators:

  &&, ||, !

console.log(c > a) && (c > b); //true

console.log(a == b) || (a > b); //true

console.log(!a);

- assignment operators:

  =, +=, -=, *=, /=, %=

let i = 2;

i += 5;

console.log(i);

i -= 4;

console.log(i);

```
i *= 9;

console.log(i);

let k = 10;

k /= 10;

console.log(k);

k %= 2;

console.log(k);
```

- ternary operator:

  <condition> ? <value1> : <value2>;

```
let yourAge = 21;
console.log((age < = 21) ? "Too young" : "Too old");
```

PHP

Questions:

1. What types are legal for subscripts?

   The data type of array subscript is integer or int in PHP.

```
$string = 'abcdef';
echo $string[0];
echo $string[1];
echo $string[2];
echo $string[3];
echo $string[4];
echo $string[5];
$n = "\n";
echo $n;
```

2. Are subscripting expressions in element references range checked?

No, they are not checked.

```
echo $string[7]; //this does not give any errors like other programming
//languages, it just prints random numbers
```

3. When are subscript ranges bound?

During compile-time (static).

4. When does allocation take place?

It takes place during compile-time.

```
$melis[2]; //this also does not give any errors even if melis is not yet declared, it
//prints random numbers
$melis = ‘melis’;
```

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

Multidimensional arrays are allowed.

```
//multidimensional array
$myArray = array (
  array("Melis", 27, 10),
  array("Cagan", 16, 11),
  array("Ata", 28, 12),
  array("Cigdem", 13, 14)
);
```

6. Can array objects be initialized?

Yes, they can be initialized.

```
class myFlowers

{

   function blossom()
```

```
    {
        echo "You are a flower!";
    }
}
$flower = new myFlowers;
$flower->blossom();
```

7. Are any kind of slices supported?

Yes, array slicing is supported with substr() method.

```
$myName = substr("melis", 0, -1);
echo $myName;
$n = "\n";
echo $n;
$myName = substr("melis", 2, -1);
echo $myName;
$n = "\n";
echo $n;
```

8. Which operators are provided?

- arithmetic operators:

  +, -, *, /, %, **

```
$m = 2;
$k = 3;
echo $m + $k;
$n = "\n";
echo $n;
echo $m - $k;
$n = "\n";
echo $n;
echo $m * $k;
$n = "\n";
echo $n;
echo $m ** $k;
$n = "\n";
echo $n;
$y = 10;
$z = 2;
echo $y / $z;
$n = "\n";
echo $n;
echo $y % $z;
$n = "\n";
echo $n;
```

- comparison operators:

  ==, ===, !=, >, <, >=, <=

```
$var1 = 10;
$var2 = 10;
var_dump($var1 == $var2); //true
$n = "\n";
echo $n;
var_dump($var1 != $var2); //false
$n = "\n";
echo $n;
$var3 = '30';
var_dump($var1 === $var3); //false
$n = "\n";
echo $n;
var_dump($var3 > $var2); //true
$n = "\n";
echo $n;
var_dump($var1 < $var2); //false
$n = "\n";
```

```
echo $n;
var_dump($var1 >= $var2); //true
$n = "\n";
echo $n;
var_dump($var1 <= $var2); //true
$n = "\n";
echo $n;
```

- increment and decrement operators:

  $x++, ++$x, $x--, --$x

```
$a = 1;
$a++;
print_r($a);
$n = "\n";
echo $n;
$a--;
print_r($a);
$n = "\n";
echo $n;
$b = 3;
++$b;
print_r($b);
$n = "\n";
echo $n;
--$b;
print_r($b);
$n = "\n";
echo $n;
```

- logical operators:

  and, or, xor, &&, ||, !

```
$c = true && false;
var_dump($c);
```

```
$n = "\n";
echo $n;
$d = true || false;
var_dump($d);
$n = "\n";
echo $n;
$e = 30;
$f = 40;
if ($a and $b) {
          echo "This is a test";
  $n = "\n";
  echo $n;
}
if ($a or $b) {
          echo "This is another test";
  $n = "\n";
  echo $n;
}
if (!$a) {
          echo "This is again a test";
  $n = "\n";
  echo $n;
}
```

- array operators:

  +, ==, ===, !=, !==

```
$melo = array("g" => "galatasaray", "a" => "altay");
$pasta = array("f" => "fenerbahce", "t" => "trabzonspor", "b" => "besiktas");
$cake = $melo + $pasta;
echo "Union of \$pasta and \$cake is: ";
var_dump($cake);
$n = "\n";
echo $n;
var_dump($pasta == $cake); //false
```

```
$n = "\n";
echo $n;
var_dump($pasta === $cake); //false
$n = "\n";
echo $n;
var_dump($pasta != $cake); //true
$n = "\n";
echo $n;
var_dump($pasta !== $cake); //true
$n = "\n";
echo $n;
$name1 = "Beautiful";
$name2 = " ";
$name3 = "Melis";
$result = $name1 . $name2 . $name3;
echo $result;
$n = "\n";
echo $n;
```

- string operators:

  ., .=

```
$name1 = “Beautiful”;
$name2 = “ ”;
$name3 = “Melis”;
$n = “\n”;
$result = $name1 . $name2 . $name3 . $name4;
echo $result;
```

- conditional assignment operators:
  ?:, ??

```
$myVar1 = 10;
$myVar2 = 20;
$myVar3 = ($myVar1 > $myVar2 ) ? $myVar1 :$myVar2;
echo "Value of result is: $myVar3";
$n = "\n";
```

echo $n;
$myVar3 = ($myVar1 < $myVar2) ? $myVar1 :$myVar2;
echo "Value of result is: $myVar3";
$n = "\n";
echo $n;

PYTHON

Questions:

1. What types are legal for subscripts?

    Strings, integers.

myName = “Melis”

print(myName[0])

print(myName[1])

print(myName[2])

print(myName[3])

print(myName[4])

2. Are subscripting expressions in element references range checked?

    Yes, they are checked.

#print(myName[5]) #this gives an error because the string myName is
#composed of only 4 letters, trying to reach the 5th one therefore gives an error

3. When are subscript ranges bound?

    Subscript ranges are dynamically bound and the storage allocation is also dynamic, meaning that it happens at run-time.

4. When does allocation take place?

    It takes place during run-time.

```
#myFather[0] #this gives an error because myFather is not yet declared
myFather = "Ata"
print(myFather[0]) #because myFather is declared above, this line will
                    #execute without errors
```

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

Both are allowed in Python with numpy. However, a jagged array in python is a list of lists.

```
#multidimensional array
x = np.array([[9, 8, 7], [6, 5, 4]])
print(x.shape)
print(x[1, 2])
#jagged array
Bilkent = {"Classes":["CS315", "CS319"], "Sports":["Tennis", "Basketball"]};
```

6. Can array objects be initialized?

Yes, they can be initialized.

```
myArray = np.empty(5, dtype=object)

print(array)

class myClass:
    def __init__(self, a_melis, b_melis, c_mother, c_father):
        self.a = a_melis
        self.b = b_melis
        self.mother = c_mother
        self.father = c_father
```

7. Are any kind of slices supported?

In python, slice notation for any sequential data type like lists, strings, tuples, bytes, bytearrays, and ranges are supported.

```
myMother = "Cigdem"
print(myMother[0:3]) #this will take the string from 0th index till second index
print(myMother[1:3])  #this will take the string from first index till second
                       #index
print(myMother[0:4]) #this will take the string from first index till 3rd index
print(myMother[1:5]) #this will take the string from first index till 4th index
print(myMother[2:4]) #this will take the string from second index till second
                              #index
print(myMother[3:5]) #this will take the string from 3rd index till 4th index
```

8. Which operators are provided?

- arithmetic operators:

  +, -, *, /, %, **, //

```
m = 3

n = 4

k = m + n

print(k)

l = 5

i = 8

o = i - l

print(o)

e = 10

w = 5

y = e / w #result is division (float)

z = e // w #result is floor
```

g = e * w

v = e ** w #power

j = e % w #remainder

print(y)

print(z)

print(g)

print(v)

print(j)

- assignment operators:

  =, +=, -=, *=, /=, %=, //=, **=, &=, |=, ^=, >>=, <<=

melis = 5

melis *= 4

print(melis) #output is 20

melis -= 5

print (melis) #output is 0

- comparison operators:

  ==, !=, >, <, >=, <=

computer = 1

glass = 2

print(computer < glass) #true

print(computer >= glass) #false

print(computer != glass) #true

- logical operators:

  and, or, not

al = true;

bal = false;

print(al and bal) #output is false

print(al or bal) #output is true

print(not al) #output is false

- identity operators:

  is, is not

mel = 30
isa = 20
ays = mel
print(mel is not isa) #true
print(mel is ays) #true

- membership operators:
  in, not in

x = 24
y = 30
z = 23
list = [24, 30, 23, 45]
if (x not in list):
  print("False alarm!")
else:
  print("Correct!")

- bitwise operators
  &, |, ^, ~, <<, >>

cable = 45

chart = 20
print(cable & chart)
print(cable | chart)
print(~cable)
print(cable ^ chart)
print(cable >> 2)
print(cable << 2)

RUST

Questions:

1. What types are legal for subscripts?

    Integers.

let my_array:[i32;5] = [1, 2, 3, 4, 5]; //myArray has a length of 5 and consists //of 32 bit integers

println!("Array is: {} ", my_array[0]);

let my_array2:[i32;3] = [0;3]; //arrays can also be declared like this
println!("My other array is: {} ", my_array2[0]);
let mut my_array3:[i32;2] = [7, 8]; //content of mutable arrays cannot be //changed
println!("Mutable array is: {} ", my_array3[0]);

2. Are subscripting expressions in element references range checked?

    Yes, they are checked.

my_array3[1] = 9; //this line is allowed, it will change the second index of //mutable array
//myMutableArray[3] = 12; //this line gives an error because mutable array //has a length of 2, therefore trying to reach the 3rd index gives an error

3. When are subscript ranges bound?

    They are bound at run-time.

4. When does allocation take place?

It takes place at compile-time. The allocation is local to a function call.

my_array4[0]; //this gives an error because my_array4 is not yet declared

let my_array4:[i32;3] = [9, 8, 7];

my_array4[2] = 6; //this line executes without errors because checkArr has //been declared above

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

Rectangular arrays are allowed, ragged arrays are not in Rust.

```
//multidimensional arrays
let mut my_array5 = [[0u8; 3]; 4];
my_array5[0][1] = 6;
println!("Multidimensional array is: {} ", my_array5[0][1]);
```

6. Can array objects be initialized?

Yes, indeed they have to be initialized.

```
let _: [u8; 3] = [1, 2, 3];

let _: [&str; 3] = ["1", "2", "3"];

let _: [String; 3] = [

   String::from("1"),

   String::from("2"),

   String::from("3")

];
```

7. Are any kind of slices supported?

No, it is not supported in Rust.

8. Which operators are provided?

- arithmetic operators:

  +, -, *, /, %

```
let var1 = 2;

let var2 = 4;

let mut result:i32;

result = var1 + var2;

println!(“Sum is: {} ”, result);

result = var1 - var2;

println!(“Substraction is: {} ”, result);

result = var1 * var2;

println!(“Multiplication is: {} ”, result);

result = var1 / var2;

println!(“Division is: {} ”, result);

result = var1 % var2;

println(“Remainder is: {} ”, result);
```

- logical operators:

  &&, ||, !

- comparison operators:

  >, <, ==, !=, >=, <=

```
let m = 7;
let k = 8;
if (m < k) && (k > m) {
        println!("True example.");
        }
        let my_boolean = false;
        if !my_boolean {
                println!("True example again.");
        }
if (m > 7) || (k > 7) {
        println!("True example again.");
        }
```

- bitwise operators
  &, |, ^, !, >>, <<

```
let a:i32 = 0;
let b:i32 = 1;
let mut c:i32;
c = a & b;
println!("1st result is: {} ", c);
c = a | b;
println!("2nd result is: {} ", c);
c = a ^ b;
println!("3rd result is: {} ", c);
c = !b;
println!("4th result is: {} ", c);
c = a << b;
println!("5th result is: {} ", c);
c = a >> b;
println!("Last result is: {} ", c);
```

- assignment operators
  +=, -=, *=, /=, %=, <<=, >>=, &=, |=, ^=

```
let mut i = 8;
i += 4;
println!("i is: {} ", i);
```

```
i -= 2;
println!("i is: {} ", i);
i *= 4;
println!("i is: {} ", i);
i /= 2;
println!("i is: {} ", i);
i %= 10;
println!("i is: {} ", i);
i &= 2;
println!("i is: {} ", i);
i |= 1;
println!("i is: {} ", i);
i ^= 2;
println!("i is: {} ", i);
i <<= 1;
println!("i is: {} ", i);
i >>= 1;
println!("i is: {} ", i);
```

Which language is the best for array operations?

In my opinion, Python is the best programming language for array operations. First of all, it is the most readable language that I have ever seen, which makes it very easy to code in Python. It is not complicated at all. In addition, it has index checking at compile-time. For example, trying to reach the 4th element of an array which has 3 elements gives "index out of bounds" error directly at compile-time, before running the code. However, this is not the case in other languages. For instance, Javascript compiles without any errors and outputs "undefined" on the console, rather than giving any run-time or compile-time errors. Also, other languages give only run-time errors, not compile-time errors, which makes Python a very good language for array operations. Furthermore, Python also supports negative indexing and since arrays are dynamic in Python, they get can bigger or smaller in size. Python also has associative arrays and list compherensions when initializing arrays. There are also some useful operations on arrays that Python supports: array concatenation, operator merging, etc. Jagged arrays and array slicing are also supported by Python which is usually not the case for other programming languages. To sum up, these features make Python the best programming language for array operations.

My Learning Strategy

First of all, without starting to write any code snippets, I did a general research on the programming language that I am studying at that moment. This allowed me to have a general information about that language and made me familiar with the syntax of the language. After doing this general research, I started to research the specific eight questions that we have in the homework. I tried to write code snippets for each question rather than writing a whole program and this allowed me to learn the rules of the programming language better without mixing them up and being drowned in errors later on. I tried to look into every resource that I can found so that I can have a better understanding of the specific rule that will be the answer of a question. However, I also encountered some unreliable resources that lack necessary information and have redundant information and I eliminated them. Later on, after writing code snippets for every question, I brought them together in a whole program and put the program in Atom text editor, which is my favorite one to use. My teacher David told me about Atom. After saving my program with the proper extension (.html, .rs, .py, .dart, .php), I opened terminal on my MacBook and ran my program. At first, I encountered many errors which I inspected, of course. Then, I studied the errors that I got by googling them or simply correcting them if they are simple mistakes that I did not notice (for example, accidentally writing ; at the end of lines in Python or forgetting to put the closing parenthesis). After I resolved all of the errors, I opened FileZilla and connected to the school server. Since I am connected to school WIFI because I am staying at dorms, I did not have to use VPN which made my life a lot easier because VPN works very slowly on my computer. Then, I uploaded my program to the server and tested it again on Terminal (with ssh, I do not need Putty on MacOS). However, I want to note that Javascript cannot be tested like other four languages on Dijkstra or Mac Terminal. Therefore, I did a brief research about how to run my Javascript program. We have learned this in our freshman year but I realized that I forgot it. After doing my research, I learned that Javascript can be run on Vscode. Then, I opened Vscode and changed some JSON settings in order to be able to run Javascript code. After that, I put my Javascript file on Vscode and ran it. When I clicked on the run button, Google Chrome has opened up and there it was, my program. However, to be able to see the output and the errors if any on the console, I right clicked on the page and clicked on “Inspect”, then I clicked on “console”. I checked my program from the console and resolved the mistakes if I had any. After being completely sure that my code works properly without any errors and it is a clean code with proper comments and everything, I closed the program. Then, I checked my pdf report of the homework because I made many changes on the code snippets since I resolved many errors and added some stuff in order to test my program better. After I made sure that my pdf report is proper as well, I moved on to the next language. I repeated this process for every five languages in the homework. This homework was great to learn about several interesting programming languages that are being used very often in computer science world nowadays.

References

“A tour of the Dart language,” *Dart*. [Online]. Available: https://dart.dev/guides/language/language-tour. [Accessed: 24-Nov-2021].

“Create a 2-dimensional array, in Dart”, *programming-idioms*. [Online]. Available: https://programming-idioms.org/idiom/26/create-a-2-dimensional-array/631/dart. [Accessed: 24-Nov-2021].

C. Wallace, “Jagged_array - data structures in rust,” *jagged_array - data structures in Rust //*, 27-May-2017. [Online]. Available: https://lib.rs/crates/jagged_array. [Accessed: 24-Nov-2021].

"Dart Documentation," *Dart*. [Online]. Available: https://dart.dev/guides. [Accessed: 24-Nov-2021].

"Dart static keyword," *GeeksforGeeks*, 25-Jul-2021. [Online]. Available: https://www.geeksforgeeks.org/dart-static-keyword/. [Accessed: 24-Nov-2021].

D. Kawar, "Working with multi-dimensional list in Dart," *Medium*, 29-Jan-2020. [Online]. Available: https://medium.com/flutter-community/working-with-multi-dimensional-list-in-dart-78ff332430a.[Accessed: 24-Nov-2021].

"How to create an array in Dart," *Educative*. [Online]. Available: https://www.educative.io/edpresso/how-to-create-an-array-in-dart. [Accessed: 24-Nov-2021].

"How to declare an array in rust code example," *how to declare an array in rust Code Example*. [Online]. Available: https://www.codegrepper.com/code-examples/rust/how+to+declare+an+array+in+rust. [Accessed: 25-Nov-2021].

"How to print superscript and subscript in python?," *GeeksforGeeks*, 24-Jan-2021. [Online]. Available: https://www.geeksforgeeks.org/how-to-print-superscript-and-subscript-in-python/. [Accessed: 24-Nov-2021].

JavaScript arrays. [Online]. Available: https://www.w3schools.com/js/js_arrays.asp. [Accessed: 24-Nov-2021].

JavaScript arrays. [Online]. Available: https://www.w3schools.com/JS//js_arrays.asp. [Accessed: 24-Nov-2021].

"Multidimensional array in JavaScript," *GeeksforGeeks*, 27-Jul-2021. [Online]. Available: https://www.geeksforgeeks.org/multidimensional-array-in-javascript/. [Accessed: 24-Nov-2021].

"Multidimensional Array Dart code example," *multidimensional array dart Code Example*. [Online]. Available: https://www.codegrepper.com/code-examples/dart/multidimensional+array+dart. [Accessed: 24-Nov-2021].

"Numpy n-dimensional array(ndarray)," *w3resource*. [Online]. Available: https://www.w3resource.com/numpy/ndarray/index.php. [Accessed: 24-Nov-2021].

"NumPy v1.21 manual," *Overview - NumPy v1.21 Manual*. [Online]. Available: https://numpy.org/doc/stable/. [Accessed: 24-Nov-2021].

O. Polesny, "JavaScript array of objects tutorial – how to create, update, and loop through objects using JS array methods," *freeCodeCamp.org*, 14-May-2020. [Online]. Available: https://www.freecodecamp.org/news/javascript-array-of-objects-tutorial-how-to-create-update-and-loop-through-objects-using-js-array-methods/. [Accessed: 25-Nov-2021].

"Object - manual," *php*. [Online]. Available: https://www.php.net/manual/tr/language.types.object.php. [Accessed: 25-Nov-2021].

P. Team, "Printing subscript in Python - Pretag," *Pretag development team*. [Online]. Available: https://pretagteam.com/question/printing-subscript-in-python. [Accessed: 24-Nov-2021].

"PHP: Substr() function," *GeeksforGeeks*, 09-Mar-2018. [Online]. Available: https://www.geeksforgeeks.org/php-substr-function/. [Accessed: 24-Nov-2021].

PHP operators. [Online]. Available: https://www.w3schools.com/php/php_operators.asp. [Accessed: 24-Nov-2021].

PHP Multidimensional Arrays. [Online]. Available: https://www.w3schools.com/php/php_arrays_multidimensional.asp. [Accessed: 24-Nov-2021].

"Python operators," *GeeksforGeeks*, 07-Sep-2021. [Online]. Available: https://www.geeksforgeeks.org/python-operators/. [Accessed: 24-Nov-2021].

Python strings. [Online]. Available: https://www.w3schools.com/python/python_strings.asp. [Accessed: 24-Nov-2021].

"Rust - array," *GeeksforGeeks*, 31-Mar-2021. [Online]. Available: https://www.geeksforgeeks.org/rust-array/. [Accessed: 24-Nov-2021].

"Rust - operators," *GeeksforGeeks*, 21-Jun-2021. [Online]. Available: https://www.geeksforgeeks.org/rust-operators/. [Accessed: 24-Nov-2021].

"The N-dimensional array (ndarray)¶," *The N-dimensional array (ndarray) - NumPy v1.21 Manual*. [Online]. Available: https://numpy.org/doc/stable/reference/arrays.ndarray.html. [Accessed: 24-Nov-2021].

T. Decker and T. Decker, *Codeigo*, 18-Apr-2020. [Online]. Available: https://codeigo.com/python/printing-subscript-and-superscript. [Accessed: 24-Nov-2021].

https://stackoverflow.com/questions/53797581/easily-check-if-a-number-is-in-a-given-range-in-dart/53797617

https://stackoverflow.com/questions/13318207/how-to-get-a-random-number-from-range-in-dart

https://stackoverflow.com/questions/25801005/difference-between-dynamic-and-static-type-in-dart

https://stackoverflow.com/questions/57860596/creating-a-2d-array-in-flutter-dart

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

https://stackoverflow.com/questions/24391892/printing-subscript-in-python

https://stackoverflow.com/questions/13212212/creating-two-dimensional-arrays-in-rust

https://stackoverflow.com/questions/43323865/how-to-make-an-array-of-objects-in-python/43323943