CS315

Programming Languages

HW2: Short-Circuit Evaluation in Dart, Javascript, PHP, Python, and Rust

N. Onur VURAL

21902330

SEC: 01

1- How are the boolean values represented?

Dart

truthy: true

EXAMPLE-> bool truthyVal = true;

falsy: false

EXAMPLE-> bool falsyVal = false;

Javascript:

truthy: true, string, integer, expression

EXAMPLES->

var truthyVal = true;

var truthyVal = “non-empty-string”;

var truthyVal = -3; (any integer or float different than 0)

var truthyVal = (7 >= 2); (any correct expression)

falsy: false, string, integer, expression, null, NaN

EXAMPLES->

var falsyVal = false;

var falsyVal = 0; (or -0)

var falsyVal = “”; (or ‘’)

var falsyVal = null;

var falsyVal = NaN;

var falsyVal = (7 <= 2); (any incorrect expression)

PHP:

truthy: bool(true)

EXAMPLES->

$truthyVal = true; (TRUE, True, TRue…); // bool(true)

$truthyVal = -3 (any integer other than 0) // bool(true)

$truthyVal = "anyString"; // bool(true)

$truthyVal = array(3); // bool(true)

falsy: bool(false)

EXAMPLES->

$falsyVal = false; (FALSE, False, FAlse…) // bool(false)

$falsyVal = 0; // bool(false)

$falsyVal = 0.0; // bool(false)

$falsyVal = '0'; // bool(false)

$falsyVal = ''; // bool(false)

$falsyVal = array(); // bool(false)

$falsyVal = NULL; // bool(false)

Python:

truthy: True, string, integer, expression, array, tuple…

EXAMPLES->

truthyVal = True

truthyVal = 1 (any other than 0)

truthyVal = "anyNonEmptyString"

truthyVal = ["anyNonEmptyList", 3, "sfgsdg"]

truthyVal = ("anyNonEmptyTuple", 3, "sfgsdg")

falsy: False, string, integer, expression, null, empty array, empty tuple…

EXAMPLES->

falsyVal = False

falsyVal = 0

falsyVal = ""

falsyVal = []

falsyVal = ()

falsyVal = {}

falsyVal = None

Rust:

truthy: true

EXAMPLE-> let truthy_val = true;

falsy: false

EXAMPLE-> let falsy_val = false;

2- What operators are short-circuited?

Dart: &&, ||

Javascript: &&, ||, ??

PHP: &&, AND, ||, OR, ??

Python: and, or, not, (<, >, >=, <=) when used to compare triplets

Rust: &&, ||

3- How are the results of short-circuited operators computed? (Consider also function calls)

Dart:

```
bool falsyValReturner() {
  print("in function falsyValReturner");
  return false;
}
bool truthyValReturner() {
  print("in function truthyValReturner");
  return true;
}
bool truthyVal = true;
bool falsyVal = false;
```

- **x && y: if x is truthyVal than y, else x**

Evaluation With Short Circuiting

EXAMPLE-> falsyVal && truthyVal; // **false**

EXAMPLE-> falsyVal && truthyValReturner(); // **false**

EXAMPLE-> falsyVal Returner() && truthyValReturner(); // **in function falsyValReturner / false**

Evaluation Without Short Circuiting

EXAMPLE-> truthyVal && falsyVal; // **false**

EXAMPLE-> truthyVal && falsyValReturner(); // **in function falsyValReturner / false**

EXAMPLE-> truthyValReturner() && falsyValReturner(); // **in function truthyValReturner / in function falsyValReturner / false**

- **x || y: if x is falsyVal than y, else x**

Evaluation With Short Circuiting

EXAMPLE-> truthyVal || falsyVal; **// true**

EXAMPLE-> truthyVal || falsyValReturner(); **// true**

EXAMPLE-> truthyValReturner() || falsyValReturner(); **// in function truthyValReturner / true**

Evaluation Without Short Circuiting

EXAMPLE-> falsyVal || truthyVal; **// true**

EXAMPLE-> falsyVal || truthyValReturner(); **// in function truthyValReturner / true**

EXAMPLE-> falsyValReturner() || truthyValReturner(); **// in function falsyValReturner/ in function truthyValReturner / true**

Javascript:

```
function falsyValReturner() {
        document.writeln("entered falsyValReturner");
    return false;
 }
function truthyValReturner() {
            document.writeln("entered truthyValReturner");
    return 1;
}
var truthyVal = “abc”;
var falsyVal = 0;
```

- **x && y: if x is truthyVal than y, else x**

Evaluation With Short Circuiting

EXAMPLE-> falsyVal && truthyVal; // **0**

EXAMPLE-> falsyVal && truthyValReturner(); // **0**

EXAMPLE-> falsyValReturner() && truthyValReturner(); // **in function falsyValReturner / false**

Evaluation Without Short Circuiting

EXAMPLE-> truthyVal && falsyVal; // **0**

EXAMPLE-> truthyVal && falsyValReturner(); // **in function falsyValReturner / false**

EXAMPLE-> truthyValReturner() && falsyValReturner(); // **in function truthyValReturner / in function falsyValReturner / false**

- **x || y: if x is falsyVal than y, else x**

Evaluation With Short Circuiting

EXAMPLE-> truthyVal || falsyVal; // **“abc”**

EXAMPLE-> truthyVal || falsyValReturner(); // **“abc”**

EXAMPLE-> truthyValReturner() || falsyValReturner(); // **in function truthyValReturner / 1**

Evaluation Without Short Circuiting

EXAMPLE-> falsyVal || truthyVal; // **true**

EXAMPLE-> falsyVal || truthyValReturner(); // **in function truthyValReturner / true**

EXAMPLE-> falsyValReturner() || truthyValReturner(); // **in function falsyValReturner/ in function truthyValReturner / true**

- **a ?? b : if a not NULL return a, else b**

```
function defaultVal() {

    document.writeln("I entered to the function defaultVal() to return:");
```

```
    return "functVal";
  }
```

Evaluation Without Short Circuiting

var val;

EXAMPLE-> var myVal = val ?? defaultVal(); // **I entered to the function defaultVal() to return: "functVal"**

Evaluation With Short Circuiting

val = "aStr";

EXAMPLE-> var myVal = val ?? defaultVal(); // **myVal will directly get "aStr"**

PHP:

```
function falsyValReturner() {
  echo "in function falsyValReturner<br \>" . PHP_EOL;
  return false;
}
function truthyValReturner() {
  echo "in function truthyValReturner<br \>" . PHP_EOL;
  return true;
}
$truthyVal = true;
$falsyVal = false;
```

- **x &&, AND y: if x is truthyVal than y, else x**

Evaluation With Short Circuiting

EXAMPLE-> var_dump($falsyVal && $truthyVal); //**bool(false)**

EXAMPLE-> var_dump($falsyVal && truthyValReturner()); //**bool(false)**

EXAMPLE-> var_dump(falsyValReturner() && truthyValReturner()); // **in function truthyValReturner/ bool(false)**

Evaluation Without Short Circuiting

EXAMPLE-> var_dump($truthyVal && $falsyVal); //**bool(false)**

EXAMPLE-> var_dump($truthyVal && falsyValReturner()); // **in function falsyValReturner/ bool(false)**

EXAMPLE-> var_dump(truthyValReturner() && falsyValReturner()); // **in function truthyValReturner / in function falsyValReturner/ bool(false)**

- **x ||, OR y: if x is falsyVal than y, else x**

Evaluation With Short Circuiting

EXAMPLE-> var_dump($truthyVal || $falsyVal); // **bool(true)**

EXAMPLE-> var_dump($truthyVal || falsyValReturner()); // **bool(true)**

EXAMPLE-> var_dump(truthyValReturner() || falsyValReturner()); // **in function truthyValReturner /bool(true)**

Evaluation Without Short Circuiting

EXAMPLE-> var_dump($falsyVal || $truthyVal); // **bool(true)**

EXAMPLE-> var_dump($falsyVal || truthyValReturner()); // **in function truthyValReturner /bool(true)**

EXAMPLE-> var_dump(falsyValReturner() || truthyValReturner()); // **in function falsyValReturner/ in function truthyValReturner /bool(true)**

- **a ?? b : if a not NULL return a, else b**

```
function defaultVal() {
   echo "I entered to the function defaultVal()<br \>" . PHP_EOL;
   return "DefVal";
}
```

Evaluation Without Short Circuiting

EXAMPLE-> $myVal = $val ?? defaultVal();

echo $myVal; // **"DefVal"**

Evaluation With Short Circuiting

$val = "VAL";

EXAMPLE-> $myVal = $val ?? defaultVal();

echo $myVal; // **"VAL"**

Python:

```
def falsyValReturner() :
  print("in function falsyValReturner")
  return False
def truthyValReturner() :
  print("in function truthyValReturner")
  return 1
truthyVal = "abc"
falsyVal = 0
```

- **x and y: if x is truthyVal than y, else x**

Evaluation With Short Circuiting

EXAMPLE-> print(falsyVal and truthyVal)**# 0**

EXAMPLE-> print(falsyVal and truthyValReturner()) **# 0**

EXAMPLE-> print(falsyValReturner() and truthyValReturner()) **# in function falsyValReturner / False**

Evaluation Without Short Circuiting

EXAMPLE-> print(truthyVal and falsyVal) **# 0**

EXAMPLE-> print(truthyVal and falsyValReturner()) **# in function falsyValReturner/ 0**

EXAMPLE-> print(truthyValReturner() and falsyValReturner()) **# in function truthyValReturner/ in function falsyValReturner/ False**

- **x or y: if x is falsyVal than y, else x**

Evaluation With Short Circuiting

EXAMPLE-> print(truthyVal or falsyVal) **# "abc"**

EXAMPLE-> print(truthyVal or falsyValReturner()) **# "abc"**

EXAMPLE-> print(truthyValReturner() or falsyValReturner()) **# in function truthyValReturner/ 1**

Evaluation Without Short Circuiting

EXAMPLE-> print(falsyVal or truthyVal) **# "abc"**

EXAMPLE-> print(falsyVal or truthyValReturner()) **# in function truthyValReturner / "abc"**

EXAMPLE-> print(falsyValReturner() or truthyValReturner()) **# in function falsyValReturner / in function truthyValReturner/ 1**

- **not x: x is True return False, else return True**

Evaluation With Short Circuiting

EXAMPLE-> print(not falsyVal) **# True**

Evaluation Without Short Circuiting

EXAMPLE-> print(not truthyVal) **# False**

- **in ternary and more comparisons, if lhs is False return False, else compute**

```
def numReturner(num) :
  print("in function numReturner")
  return num
```

Evaluation With Short Circuiting

EXAMPLE-> print(5 <= 4 < numReturner(2)) **# False**

Evaluation Without Short Circuiting

EXAMPLE-> print(4 <= 5 < numReturner(2)) **# in function numReturner / False**

Rust:

```
fn falsy_val_returner() -> bool{
  println!("in function falsy_val_returner");
  return false;
}
fn truthy_val_returner() -> bool{
  println!("in function truthy_val_returner");
  return true;
}
let truthy_val = true;
let falsy_val = false;
```

- **x && y: if x is truthyVal than y, else x**

Evaluation With Short Circuiting

EXAMPLE-> println!("{}", falsy_val && truthy_val); // **false**

EXAMPLE-> println!("{}", falsy_val && truthy_val_returner()); // **false**

EXAMPLE-> println!("{}", falsy_val_returner() && truthy_val_returner()); // **in function truthy_val_returner/ false**

Evaluation Without Short Circuiting

EXAMPLE-> println!("{}",truthy_val && falsy_val); // **false**

EXAMPLE-> println!("{}", truthy_val && falsy_val_returner()); // **in function falsyValReturner/ false**

EXAMPLE-> println!("{}", truthy_val_returner() && falsy_val_returner()); // **in function truthy_val_returner/ in function falsyValReturner/ false**

- **x || y: if x is falsyVal than y, else x**

Evaluation With Short Circuiting

EXAMPLE-> println!("{}", truthy_val || falsy_val); // **true**

EXAMPLE-> println!("{}", truthy_val || falsy_val_returner()); // **true**

EXAMPLE-> println!("{}", truthy_val_returner() || falsy_val_returner()); // **in function truthy_val_returner/ true**

Evaluation Without Short Circuiting

EXAMPLE-> println!("{}", falsy_val || truthy_val); // **true**

EXAMPLE-> println!("{}", falsy_val || truthy_val_returner()); // **in function truthy_val_returner/ true**

EXAMPLE-> println!("{}", falsy_val_returner() || truthy_val_returner()); // **in function falsyValReturner/ in function truthy_val_returner/ true**

4- What are the advantages about short-circuit evaluation?

*** short-circuit evaluation helps to efficiently go through decision statements, thus reducing the precious time loss, by the help of short circuit evaluation, the second operand will only be evaluated when its necessary!**

**** short-circuit evaluation helps to eliminate run-time errors**

***** thanks to short-circuit evaluation, complex boolean expressions can be written inside if statements!**

Dart:

EXAMPLE FOR *->bool haveEnoughMoney = haveEnoughMoneyInCard || checkYourPurse(10000);

// **No need to call checkYourPurse() if haveEnoughMoneyInCard has truthy**

EXAMPLE FOR *-> if (haveEnoughMoney && productIsAvailableInShelf(item))

//productIsAvailableInShelf() may have costly operations interacting with large data for example, if haveEnoughMoney is false there is no need to evaluate productIsAvailableInShelf() thanks to short circuit evaluation

EXAMPLE FOR **-> if ((dangerousNumber != 0) && (1000/dangerousNumber > 100))

// (dangerousNumber != 0) may prevent evaluation of 1000/dangerousNumber thanks to short circuit evaluation!

EXAMPLE FOR ***->

if ((dangerousNumber != 0) && (1000/dangerousNumber > 100)) {...}is simpler version of:

if(dangerousNumber != 0){if (1000/dangerousNumber > 100){...}}

Javascript:

EXAMPLE FOR *->

var haveEnoughMoney = haveEnoughMoneyInCard || checkYourPurse(10000);

// **No need to call checkYourPurse() if haveEnoughMoneyInCard has truthy**

EXAMPLE FOR *->

if(haveEnoughMoney && productIsAvailableInShelf(item))

//productIsAvailableInShelf() may have costly operations interacting with large data for example, if haveEnoughMoney is false there is no need to evaluate productIsAvailableInShelf() thanks to short circuit evaluation

EXAMPLE FOR **->

if ((dangerousNumber != 0) && (1000/dangerousNumber > 100))

// (dangerousNumber != 0) may prevent evaluation of 1000/dangerousNumber thanks to short circuit evaluation!

EXAMPLE FOR ***->

if ((dangerousNumber != 0) && (1000/dangerousNumber > 100)) {...} is simpler version of:

if(dangerousNumber != 0){if (1000/dangerousNumber > 100){...}}

PHP:

EXAMPLE FOR *->

$haveEnoughMoney = $haveEnoughMoneyInCard || checkYourPurse(10000);

// **No need to call checkYourPurse() if haveEnoughMoneyInCard has truthy**

EXAMPLE FOR *->

if($haveEnoughMoney && productIsAvailableInShelf($item))

//productIsAvailableInShelf() may have costly operations interacting with large data for example, if haveEnoughMoney is false there is no need to evaluate productIsAvailableInShelf() thanks to short circuit evaluation

EXAMPLE FOR **->

if (($dangerousNumber != 0) && (1000/$dangerousNumber > 100))

// (dangerousNumber != 0) may prevent evaluation of 1000/dangerousNumber thanks to short circuit evaluation!

EXAMPLE FOR ***->

if (($dangerousNumber != 0) && (1000/$dangerousNumber > 100)){...} is a simpler form of:

if ($dangerousNumber != 0){if (1000/$dangerousNumber > 100) {…}}

Python:

EXAMPLE FOR *->

haveEnoughMoney = haveEnoughMoneyInCard or checkYourPurse(10000)

// **No need to call checkYourPurse() if haveEnoughMoneyInCard has truthy**

EXAMPLE FOR *->

if(haveEnoughMoney and productIsAvailableInShelf(item)):

//productIsAvailableInShelf() may have costly operations interacting with large data for example, if haveEnoughMoney is false there is no need to evaluate productIsAvailableInShelf() thanks to short circuit evaluation

EXAMPLE FOR **->

if ((dangerousNumber != 0) and (1000/dangerousNumber > 100)):

// (dangerousNumber != 0) may prevent evaluation of 1000/dangerousNumber thanks to short circuit evaluation!

EXAMPLE FOR ***->

if ((dangerousNumber != 0) and (1000/dangerousNumber > 100)): is a simpler form of:

if (dangerousNumber != 0): if (1000/dangerousNumber > 100):

Rust:

EXAMPLE FOR *->

let have_enough_money = have_enough_money_in_card || check_your_purse(10000);

// **No need to call check_your_purse() if have_enough_money_in_card has truthy**

EXAMPLE FOR *->

if have_enough_money && product_is_available_in_shelf(item.to_owned())

// product_is_available_in_shelf () may have costly operations interacting with large data for example, if haveEnoughMoney is false there is no need to evaluate product_is_available_in_shelf () thanks to short circuit evaluation

EXAMPLE FOR **->

if (dangerous_number != 0) && (1000/dangerous_number > 100)

// (dangerous_number != 0) may prevent evaluation of 1000/ dangerous_number thanks to short circuit evaluation!

EXAMPLE FOR ***->

if (dangerous_number != 0) && (1000/dangerous_number > 100) {…}is a simpler form of:

if dangerous_number != 0 {if 1000/dangerous_number > 100 {...}}

5- What are the potential problems about short-circuit evaluation?

* It can disregard functions that may completing a required part of the program overall...

** Efficency may decrease as well due to compiler to check for short-circuits which may result in additional cycles and time loss

Dart:

- **in expression if (checkStorageAvailability(1500) && checkResourceAvailability(1500)), both functions need to make notify admin operation but due to short-circuit evaluation, checkResourceAvailability(1500) is never evaluated**

```
bool checkStorageAvailability(int allocationSize){

  if(allocationSize < 1000){

    print("NO PROBLEM");

    return true;

  }

  else{
```

```
    print("NOTIFYING SYSTEM ADMIN ABOUT STORAGE PROBLEM....");
    // PROCESSING
    // notifyAdmin("msg: extension required for storage");
    return false;
  }
}

bool checkResourceAvailability(int resourceAmount){
  if(resourceAmount < 1000){
     print("NO PROBLEM");
     return true;
  }
  else{
    print("NOTIFYING SYSTEM ADMIN ABOUT RESOURCE PROBLEM....");
    // PROCESSING
    // notifyAdmin("msg: extension required for resources!!!");
    return false; }
}
```

EXAMPLE ->

```
if ( checkStorageAvailability(1500) && checkResourceAvailability(1500)){
      print("System works properly");}
 else{print("System isn't working properly");}
```

// Will print NOTIFYING SYSTEM ADMIN ABOUT STORAGE PROBLEM.... / System works properly

//but will not print NOTIFYING SYSTEM ADMIN ABOUT RESOURCE PROBLEM....

Javascript:

- **in expression if (checkStorageAvailability(1500) && checkResourceAvailability(1500)), both functions need to make notify admin operation but due to short-circuit evaluation, checkResourceAvailability(1500) is never evaluated**

```
unction checkStorageAvailability(allocationSize){
    if(allocationSize < 1000){
      document.writeln("NO PROBLEM");
      return true;
    }
    else{
     document.writeln("NOTIFYING SYSTEM ADMIN ABOUT STORAGE PROBLEM....");
     // PROCESSING
     // notifyAdmin("msg: extension required for storage");
     return false;
    }
  }

  function checkResourceAvailability(resourceAmount){
    if(resourceAmount < 1000){
      document.writeln("NO PROBLEM");
      return true;
    }
    else{
     document.writeln("NOTIFYING SYSTEM ADMIN ABOUT RESOURCE PROBLEM....");
```

```
      // PROCESSING
      // notifyAdmin("msg: extension required for resources!!!");
      return false;
    }
  }
```

EXAMPLE ->

```
if ( checkStorageAvailability(1500) && checkResourceAvailability(1500)){
    document.writeln("System works properly");
  }
  else{
    document.writeln("System isn't working properly");
  }
```

// Will print NOTIFYING SYSTEM ADMIN ABOUT STORAGE PROBLEM.... / System works properly

//but will not print NOTIFYING SYSTEM ADMIN ABOUT RESOURCE PROBLEM....

PHP:

- **in expression if (checkStorageAvailability(1500) && checkResourceAvailability(1500)), both functions need to make notify admin operation but due to short-circuit evaluation, checkResourceAvailability(1500) is never evaluated**

```
function checkStorageAvailability($allocationSize){
 if($allocationSize < 1000){
   echo "NO PROBLEM<br \>" . PHP_EOL;
   return true;
 }
```

```
  else{
    echo "NOTIFYING SYSTEM ADMIN ABOUT STORAGE PROBLEM....<br \>" .
PHP_EOL;
    // PROCESSING
    // notifyAdmin("msg: extension required for storage");
    return false;
  }
}

function checkResourceAvailability($resourceAmount){
  if($resourceAmount < 1000){
     echo "NO PROBLEM<br \>" . PHP_EOL;
     return true;
  }
  else{
    echo "NOTIFYING SYSTEM ADMIN ABOUT RESOURCE PROBLEM....<br \>" .
PHP_EOL;
    // PROCESSING
    // notifyAdmin("msg: extension required for resources!!!");
    return false;
  }
}
```

EXAMPLE ->

```
if ( checkStorageAvailability(1500) && checkResourceAvailability(1500)){
      echo "System works properly<br \>" . PHP_EOL;
}
```

```
else{
    echo"System isn't working properly<br \>" . PHP_EOL;
}
```

// Will print NOTIFYING SYSTEM ADMIN ABOUT STORAGE PROBLEM.... / System works properly

//but will not print NOTIFYING SYSTEM ADMIN ABOUT RESOURCE PROBLEM....

Python:

- **in expression if (checkStorageAvailability(1500) && checkResourceAvailability(1500)), both functions need to make notify admin operation but due to short-circuit evaluation, checkResourceAvailability(1500) is never evaluated**

```
def checkStorageAvailability(allocationSize):
  if(allocationSize < 1000):
    print("NO PROBLEM")
    return True
  else:
    print("NOTIFYING SYSTEM ADMIN ABOUT STORAGE PROBLEM....")
    # PROCESSING
    # notifyAdmin("msg: extension required for storage")
    return False

def checkResourceAvailability(resourceAmount):
  if(resourceAmount < 1000):
    print("NO PROBLEM")
    return True
  else:
```

```
    print("NOTIFYING SYSTEM ADMIN ABOUT RESOURCE PROBLEM....")

    # PROCESSING

    # notifyAdmin("msg: extension required for resources!!!")

    return False
```

EXAMPLE ->

```
if ( checkStorageAvailability(1500) and checkResourceAvailability(1500)):

     print("System works properly")

else:

     print("System isn't working properly")
```

// Will print NOTIFYING SYSTEM ADMIN ABOUT STORAGE PROBLEM.... / System works properly

//but will not print NOTIFYING SYSTEM ADMIN ABOUT RESOURCE PROBLEM....

Rust:

- **in expression if (checkStorageAvailability(1500) && checkResourceAvailability(1500)), both functions need to make notify admin operation but due to short-circuit evaluation, checkResourceAvailability(1500) is never evaluated**

```
fn check_storage_availability(allocation_size : i32) -> bool{

  if allocation_size < 1000 {

    println!("NO PROBLEM");

    return true;

  }

  else{

    println!("NOTIFYING SYSTEM ADMIN ABOUT STORAGE PROBLEM....");

    // PROCESSING

    // notifyAdmin("msg: extension required for storage");
```

```
    return false;
  }
}

fn check_resource_availability(resource_amount : i32) -> bool{
  if resource_amount < 1000 {
     println!("NO PROBLEM");
     return true;
  }
  else{
    println!("NOTIFYING SYSTEM ADMIN ABOUT RESOURCE PROBLEM....");
    // PROCESSING
    // notifyAdmin("msg: extension required for resources!!!");
    return false;}
  }
```

EXAMPLE ->

```
if ( check_storage_availability(1500) && check_resource_availability(1500)){
      println!("System works properly");
    }
    else{
      println!("System isn't working properly");
    }
```

// Will print NOTIFYING SYSTEM ADMIN ABOUT STORAGE PROBLEM.... / System works properly

//but will not print NOTIFYING SYSTEM ADMIN ABOUT RESOURCE PROBLEM....

ANALYSIS AND DETAILS

1. Write a paragraph discussing, in your opinion, which language is the best for short-circuit evaluation considering advantages and disadvantages. Explain why.

When it comes to short-circuit evaluations, Python turns out to be the most efficienct for many aspects overall. First of all, due to design principles of the language Python manages to supply the most amount of short circuit operations among all five languages. Essentially all five languages provide some sort of && and || operator, but Python makes it possible for those operators to operate on almost every type possible such as integers, strings, arrays, tuples and many more besides the True/False value. This design choice of Python, in other words, using almost everything as boolean variables and returning their exact value instead of returning a generic bool(true) or bool(false) gives the opportunity for short-circuit operators to flexibly evaluate complex expressions while being able to return exact values for variables at the end. Dart, Rust and PHP does not allow such flexibility wheras Javascript shows parallelism with Python in this issue. However, at this point it must be emphasized that although using many values as boolean in short circuit evaluation is generally efficient in terms of flexible operations, this may also bring negative effects as well in terms of creating confusion, disturbing consistancy and readibility. Putting this issue aside, among the languages, Python gives additional short circuit operators such as not and comparison operators. Especially the design choices of Python concerning short circuit operators makes it possible to write equalities as boolean expressions such as (1 < 2 < 3) instead of writing (1 < 2 && 2 < 3) which increases the writability in a great extent. Finally it must be addressed that Python is not case-insensitive like PHP and does not possible to use different values for generic truthy or falsy values such as TRUE, true, TrUE… This is also present in short-circuit operators as in PHP all and, &&, AND, OR, or, || are valid but not in Pyhton. Although this may suggest Rust is more efficent than Pyhton in terms of being case-insensitive, it must be noted that this issue has a tradeoff as it increases writability but gives harm to readability a lot at the same time. Overall Python seems to be the best language among all five for short circuit evaluation since it; gives the necessary and, or operators and furthermore offers additional ones, makes it possible to use a wide-range of boolean values in short-circuit evaluation.

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 checking the basic principles (such as defining variables, making operations, defining variables and functions) of each five language to remember how to write and execute programs successfully. In this respect In this respect I checked documents that offer a basic knowledge on those languages [1, 2, 3, 4, 5] and watched video tutorials [6, 7, 8, 9, 10] to remember how to execute programs from command propmpt. After refreshing my knowledge with mentioned sources, I proceeded my research by checking the representations of boolean values. Accordingly I used offical documentations or other online sources [11, 12, 13, 14, 15, 16, 17]. As a follow-up to this, it was time to search for short-circuiting principles for each language and corresponding operators. In this respect, I once again used online sources [18, 19, 20, 21, 22, 23, 24] as well as my previous knowledge coming from Mr.Güvenir's lectures in CS315. In this respect, Mr.Güvenir's teachings and the quiz he gave about booleans and short-circuit principles were crucial for me to understand the overview of the topic and furthermore they guided me during my research process to make sure that I was on the right path and gathering the necessary information. After this, I started to write programs in each language. I created files having extensions of .dart, .html, .php, .py, .rs by using Atom text editor . While working to answer the questions in example programs, I wanted to show the grader how the program answers those questions. Because of that, I wrote my programs in a specific way so that the grader can see which question is answered in regard to what examples in a clear and consistant way. In this regard, I outputted question itself, the answer and the example that explains my point to the console (outputs are on browser in Javascript program). While working on these programs, I needed to make sure that they do not produce error and working as expected. For this reason, I may have selected Dijkstra server, but as it takes a considerable amount of time to upload the file again after each change in the document and check the output from PUTTY console, I used a rather different approach. For each language I used an online compiler instead and it was much easier to experiment with my programs from there and to detect any errors [25 - 29]. After I completed my programs from online compilers, I wanted to make sure that they were working on Dijkstra as well and in the same way, therefore I entered Dijkstra server using Bilkent VPN, uploaded the files via FileZilla and used the console via PUTTY. From PUTTY console, I tested my example programs once again. It was crucial to compare the outputs of server with the previous outputs coming from online compilers to make sure that the results were in parallel with my expectancy.

Sources

Language Basics

Dart: [1] https://www.geeksforgeeks.org/dart-tutorial/

Javascript: [2] https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics

PHP: [3] https://www.w3schools.com/php/

Python: [4] https://www.w3schools.com/python/python_intro.asp

Rust: [5] https://www.rust-lang.org/learn

Video Tutorials

Dart: [6] https://www.youtube.com/watch?v=XoZ01mY-cUg

Javascript: [7] https://www.youtube.com/watch?v=821C5aJ3SLM

PHP: [8] https://www.youtube.com/watch?v=Z6JKhoMIjHE

Python: [9] https://www.youtube.com/watch?v=pFYcAOsNyvs

Rust: [10] https://www.youtube.com/watch?v=pFYcAOsNyvs

Boolean Information Sources

[11] https://api.dart.dev/stable/2.14.4/dart-core/bool-class.html

[12]https://www.tutorialspoint.com/dart_programming/dart_programming_logical_operators.htm

[13]https://www.geeksforgeeks.org/javascript-boolean/

[14]https://www.w3schools.com/python/python_booleans.asp

[15]https://doc.rust-lang.org/reference/types/boolean.html

[16]https://www.php.net/manual/tr/language.types.boolean.php

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

Short-circuit Information Sources

[18]https://kodify.net/csharp/if-else/short-circuit-if/

[19]https://www.geeksforgeeks.org/short-circuit-evaluation-in-programming/

[20]https://stackoverflow.com/questions/89154/benefits-of-using-short-circuit-evaluation

[21]https://www.geeksforgeeks.org/short-circuiting-techniques-python/

[22]https://www.pythonpool.com/short-circuit-evaluation-python/

[23]https://stackoverflow.com/questions/5694733/does-php-have-short-circuit-evaluation

[24]https://newbedev.com/does-php-have-short-circuit-evaluation

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