21801744
Uğur Erdem Seyfi
CS315 - 01

CS 315 Homework 02 Report

Javasccript

1. What are the types of loop control variables?
It seems JavaScript has no such restrictions on variables. I tried using counter variables of type string, boolean and integer and it worked without problem so far:

```
// 1. What are the types of loop control variables?

// Variable of type Integer
for(let i = 0; i < 10; i++){
	console.log(i);
}

// Variable of type Boolean
var n = 3;
for(var i = n + 1, divisibleByFive = false; !divisibleByFive; i++){
	if( i % 5 == 0){
		divisibleByFive = true;
		console.log("The first number that is divisible by 5 after " + n + " is :" + i);
	}
}

// Variable of type String
for(var i = "a"; i != "aaaa"; i = i + "a"){
	console.log(i);
}
```

2. What are the scopes of loop control variables?

Depends on how you write the initialization statement in the for loop. For example if you declare your variables using “var” keyword, they are only available in the scope of the loop, however if you initialize them without “var” keyword, they are basically functioning as variables globally declared (if a variable with the same name declared before, it changes that variables’ value).

```
// 2. What are the scopes of loop control variables?
console.log("-2-");

let x = 1;
for(x = 3; x < 10; x++){
        // nothing
}
console.log( x); // 10


for(let x = 3; x < 10; x++){
        // var z = 2;
        for(z = 0; z < 5; z++){
                // nothing
        }
}
console.log( z); // 5
```

3. Is it legal for the loop control variable or loop parameters to be changed in the loop, and if so, does the change affect loop control?
It is legal and it also effects the loop control statements. For example, the following code, prints the odd numbers in such a way. As the loop upgrades the value of n, the control condition is also affected.

```
// 3. Is it legal for the loop control variable or loop parameters to be changed in the loop, and if so, does
the change affect loop control?
console.log("-3-");

n = 10;
for(var i = 0; i < n; i++){
        i = i + 1;
        n = n + 0.5;
        console.log(i);
}
```

4. Are the loop parameters evaluated only once, or once for every iteration?
It should be only once because otherwise the control statement “i < n” would act similar to the statement “i < 10” however, the variable “i” exceeds the initial value of n (which is 10) at some point, which implies the loop parameters evaluated for every iteration.

PHP

1. What are the types of loop control variables?

As it can be seen from the following example, it seems every type can be used as a loop control variable in PHP:

```
// 1. What are the types of loop control variables?
for($i = 0, $a = "a"; $i < 10; $i = $i + 1){
        echo $a . "\n";
        $a = $a . "a";
}
```

```
$n = 7;
for($i = $n + 1, $isDivisibleBy5 = false; ! $isDivisibleBy5; $i = $i + 1){
        if( $i % 5 == 0){
                echo "First number after " . $n . " that is divisible by 5 is " . $i;
                $isDivisibleBy5 = true;
        }
}
```

2. What are the scopes of loop control variables?

The following code indicates that the initial values declared in the for statement are considered as the global variables declared outside.

```
// 2. What are the scopes of loop control variables?
// $x = 10; -this part does not change anything actually-
for($x = 0; $x < 7; $x += 1){
        // nothing
}

echo "\n" . $x; // 7
```

3. Is it legal for the loop control variable or loop parameters to be changed in the loop, and if so, does the change affect loop control?

It is legal. And it has a change effect on the loop control as it can be seen the result of the following code is 20 instead of 10.

```
// 3. Is it legal for the loop control variable or loop parameters to be changed in the loop, and if so, does
the change affect loop control?
echo "\n -3- \n";
$n = 10;
for($i = 0; $i < $n; $i += 1){
        $n = $n + 0.5;
        // echo "i is : " . $i . " \n";
        // echo "n is : " . $n . " \n ";
}

echo $i; // 20
```

4. Are the loop parameters evaluated only once, or once for every iteration?

Every iteration.

Python

1. What are the types of loop control variables?
In Python, counter control variable is iterated through a specific list such as range(). If the range object is used then the counter variable is consisted of integers. However you use an array such as [“asd”, “a”, 3] then the counter variable takes the values of these elements in the arraylist.

```
# 1. What are the types of loop control variables?
for i in range(1, 10):
        print( i)

for element in ["asd", "a", 1]:
        print(element)
```

2. What are the scopes of loop control variables?

It depends on where they are declared, if they are declared in a function then the scope of the variable is restricted to that function’s scope, however, we can think that counter variables in for loops can be thought as if they are declared out of the functions.

```
# 2. What are the scopes of loop control variables?
print("-2-")
i = 1
j = 2
for i in range(10):
        for j in range(10):
                print( str(i) + " , " + str(j) )

print("i's final state is " + str(i) ) # gives 9
print("j's final state is " + str(j) ) # gives 9
```

3. Is it legal for the loop control variable or loop parameters to be changed in the loop, and if so, does the change affect loop control?

Since, in Python, counter loop variables are enumerated via given list, even if you change the variable in the loop, after it finishes the reading statements, it continues from the next element in the list. In this manner, Python works differently from the ones that I have shown above. (JavaScript and PHP).

```
# 3. Is it legal for the loop control variable or loop parameters to be changed in the loop, and if so, does the change affect loop control?
print("-3-")
i = 2
for i in range(1, 10):
        i = i + 2
        print( i) # prints 1 to 11
```

4. Are the loop parameters evaluated only once, or once for every iteration?

They are evaluated only once, in the following code, even end changes in each iteration, the range does not change so we only get numbers from 0 to 4 printed.

```
# 4. Are the loop parameters evaluated only once, or once for every iteration?
print("-4-")
end = 5
for i in range( end):
        end = end + 1
        print( i)

print("i is iterated " + str(i+1) + " times although end's final value is : " + str(end))
```

Another confirmation to my claim is shown in the following code example, if loop parameters were evaluated for every iteration the following code would not finish:

```
test_arr = [1,1,1]
for i in range( len(test_arr)):
        test_arr.append( i)

print( test_arr) # [1, 1, 1, 0, 1, 2]
```

Dart

1. What are the types of loop control variables?

Like in JavaScript and PHP, it seems like we can use any primitive type we want, however it is good to use numbers conventionally in for-loops, for the sake of readability. For example in the following code we declared a variable called isDivisibleBy7 which is of boolean type.

```
// 1. What are the types of loop control variables?
for(var i = 9, isDivisibleBy7 = false; !isDivisibleBy7; i++){
        print( i);
        if( i % 7 == 0){
                isDivisibleBy7 = true;
                print("${i} is divisible by 7");
        }
}
```

2. What are the scopes of loop control variables?

In Dart, unlike the other programming languages we have shown above, the variables declared in the for function can only be accessed in the for function as the following code shows:

```
// 2. What are the scopes of loop control variables?
for(var i = 9; i < 10; i++){
    // do nothing
}

// print( i); -> Error: Getter not found: 'i'.
```

3. Is it legal for the loop control variable or loop parameters to be changed in the loop, and if so, does the change affect loop control?

Yes, it is legal. Furthermore as the loop parameters change in the iteration process, the way how the loop checks the conditions also affected like in the case of JavaScript and PHP above.

```
// 3. Is it legal for the loop control variable or loop parameters to be changed in the loop, and if so, does
the change affect loop control?
var n = 10.0;
for(var i = 0; i < n; i++){
    n = n + 0.5;
    i = i + 1;
    print( i); // prints 0 2 4 ... 14
}

print( n); // prints 13.0
```

4. Are the loop parameters evaluated only once, or once for every iteration?
They are evaluated for every iteration.

Rust

1. What are the types of loop control variables?
In order to iterate in Rust, a variable and an iterator expression is used. The way how for loops works in Rust is somehow familiar to Python. But unlike Python, Rust is more strict. For example it does not allow us to change the control variables inside the loop.

```
println!("-1-");
// here our counter variable i is just a number
for i in 1..10{
    println!( "{}", i );
}

// here our counter variables are of types string and number
let strings = ["hi", "ugur", "TA"];
for (index, string) in strings.iter().enumerate(){
    println!("At index : {} string : {} is found! ", index, string);
}
```

2. What are the scopes of loop control variables?
Control variables are only accessible/available in the scope of the loop that is declared. If we try to access the respective control variable outside of the for loop we get an error.

```
// 2. What are the scopes of loop control variables?
for i in 1..10{
        // do nothing
}
// println!("{}", i); -> i is not found in this scope
```

3. Is it legal for the loop control variable or loop parameters to be changed in the loop, and if so, does the change affect loop control?

Rust compiler does not allow for programmers to change control variables in the loop unlike all of the programming languages discussed above.

```
println!("-3-");
for i in 1..10{
        // i = i + 2; we get an error here
        println!( "{}", i );
}
```

4. Are the loop parameters evaluated only once, or once for every iteration?

They are evaluated only once like in the case of Python. Besides, we also do not have the chance to change the value of control variables as it has shown above so we can feel safe about this issue.

Best language for counter-controlled loop?

Now as we can see from the examples above, in JavaScript, PHP and Dart, type checking for control variables is kinda loose. In those languages, variables of different types such as Boolean or string can be initialized inside of the loop's counter expression declaration part. This is something that I don't like since I expect to a "counting" kind of process when I see the for-loop, besides, instead of putting Boolean variables in declaration of the for loops, while loops can be used (which makes the usage of different type of variables in the for loop unnecessary).

I think, JavaScript, Dart, Python and PHP also fail in terms of scoping the control variables. I do not like control variables are being accessible from outside of the loop (since they are irrelevant in the outside of the scope). Rust is the only one that suffices my expectations on this particular issue as well.

Now, for the last two properties, I think there are both good advantages and disadvantages of being able to change the control variables inside the loop and control variables being checked every iteration. However, for that issue, I would also prefer Rust instead of the other ones since I give more importance to reliability rather than flexibility.

So overall, as I explained above, I think Rust overcomes the other ones in terms of counter loops.

My Learning Strategy

My learning strategy were not different from my strategy in the first homework, which is to say again, that I used repl.it in order to speed up the process of dealing with different languages and not worrying about their respective compilation/run-time environment setup. Since I already knew JavaScript and Python and also had some acquaintance with other languages too, I did not need to look up for documentations this time. I wrote the stuff from my heart and just tried to see how things work. Again, I started with the programming languages that I was more familiar so I can find the best approach for to make my tests in the other languages too, so because of this, I started with JavaScript, PHP and Python.

For each question, I tried to come up with non-trivial but direct experiments that would lead me to a conclusion (which would also speed up my progress). If I see a compilation or run time error in a specific question, I could directly make inferences about what kind of features the programming language supports and what kind of features it does not support. So if there is a problem, I could also directly see it. In my opinion, this kind of an approach also probably speed-ed up my learning progress since several questions were also connected to the other ones in this way.

Lastly, in the homework, there are some important things that I realized. For example, I can confidently say that the process of learning a new programming language is mostly, is not about the remembering certain syntax or memorizing stuff but mostly about already knowing or being familiar with the concepts underneath of the language that is being learned. In this sense, I can say that the particular or specific things such as syntax of a programming language is not important that much. As long as the person understood the fundamental intentions of a language, it becomes very easy to adapt to the specialties in a language.

Furthermore I thing being comfortable on using the documentations provided by a specific language is also a very important part of a learning a new programming languages. However, since this homework did not require that much of a specific knowledge, I did not benefited from any online documentations this time.