**Kaan Gönç**

**21602670**

**Section 2**

**CS 315 – Homework 2**

1. **What are the types of loop control variables?**

In **Python**, the structure of ‘for’ loop is as follows

*for* <control\_variable> *in* <object>:

Here, the control variable gets the values of the elements in the objects. Therefore, the type of the control variable is the type of the element in the object.

#Control Variable gets integer values

for x in [0, 1, 2]:

print (x)

#Control Variable gets char values

for x in ['a', 'b', 'c']:

print (x)

#Control Variable gets string values

for x in ["ab", "cd", "ef"]:

print (x)

#Control Variable gets boolean values

for x in [True, False, True]:

print (x)

#Control Variable gets mixed values

for x in [0, 'a', "ab", True]:

print (x)

This example program compiles properly and prints all the elements in the arrays.

In **Javascript**, control variables can be either integer or floating numbers.

for (var x = 0; x < 3; x++) {

document.write(x + "<br>");

}

for (var x = 0.1; x < 1.0; x += 0.1) {

document.write(x + "<br>");

}

This example code compiles properly and executes the following output:

0  
1  
2  
0.1  
0.2  
0.30000000000000004  
0.4  
0.5  
0.6  
0.7  
0.7999999999999999  
0.8999999999999999  
0.9999999999999999

As it is seen form the output, the control variable with the value of floating number doesn’t work accurately because of the instability of the floating number addition.

In **PHP**, the control variable can be integers, floating numbers, or strings(chars are also counted as string in PHP)

for ($x = 0; $x <3; $x++) {

var\_dump($x);

print "<br>";

}

for ($x = 0.1; $x <1.0; $x = $x + 0.1) {

var\_dump($x);

print "<br>";

}

for ($x = 'a'; $x <'e'; $x++) {

var\_dump($x);

print "<br>";

}

This example program compiles properly and executes the following output:

int(0)   
int(1)   
int(2)   
float(0.1)   
float(0.2)   
float(0.3)   
float(0.4)   
float(0.5)   
float(0.6)   
float(0.7)   
float(0.8)   
float(0.9)   
float(1)   
string(1) "a"   
string(1) "b"   
string(1) "c"   
string(1) "d"

In **Perl**, the control variable can be integer, floating number, or string (chars are also counted as string in Perl).

#Control Variable is integer

for (my $x = 0; $x < 3; $x++)

{

print "$x\n";

}

#Control Variable is float

for (my $x = 0.1; $x < 1.0; $x += 0.1)

{

print "$x\n";

}

#Control Variable is string

for (my $x = 'aa'; $x lt 'ae'; $x++)

{

print "$x\n";

}

This example program compiles properly and executes the following output:

0

1

2

0.1

0.2

0.3

0.4

0.5

0.6

0.7

0.8

0.9

1

aa

ab

ac

ad

1. **What are the scopes of loop control variables?**

In **Python**, the scope of the control variable isn’t restricted with the body of the loop and can be accessible outside the scope of the loop. For example;

for x in range(3):

print("Inside the loop scope", x)

print("Outside the loop scope", x)

This example program compiles properly and executes the following output:

Inside the loop scope 0                                             Inside the loop scope 1                                             Inside the loop scope 2                                             Outside the loop scope 2

This output proves that the control variable ‘x’ is the same out of the loop scope as its last value in the loop is 2.

In **Javascript**, there are two options for the scope of the control variable. If the variable is declared with the ‘var’ keyword, this means it isn’t restricted with the scope of the loop (such as Python), although the variable becomes special for the loop and can’t be accessible out of it if it is declared with ‘let’ keyword.

//Control variable is declared with 'var'

for (var x = 0; x < 3; x++) {

document.write("Inside the loop scope: " + x + "<br>");

}

document.write("Outside the loop scope: " + x + "<br>");

This example program compiles properly and executes the following output:

Inside the loop scope: 0  
Inside the loop scope: 1  
Inside the loop scope: 2  
Outside the loop scope: 3

This output proves that the control variable ‘x’ is the same out of the loop scope as its value is 3 when the loop ends. Its reason i that the control variable is declared with ‘var’ keyword when the loop starts.

//Control variable is declared with 'let'

for (let x2 = 0; x2 < 3; x2++) {

document.write("Inside the loop scope: " + x2 + "<br>");

}

document.write("Outside the loop scope: " + x2 + "<br>");

This example program doesn’t compile properly and executes the following output:

Inside the loop scope: 0  
Inside the loop scope: 1  
Inside the loop scope: 2

As it is seen from the output, the last statement can’t be executed because the variable ‘x2’ is unknown at that point. Its reason is that the control variable was declared with ‘let’ keyword and is usable for only inside the loop scope.

In **PHP**, the scope of the control variable isn’t restricted with the body of the loop and can be accessible outside the scope of the loop. For example;

for ($x = 0; $x <3; $x++) {

var\_dump($x);

print "<br>";

}

var\_dump($x);

This example program compiles properly and executes the following output:

int(0)   
int(1)   
int(2)   
int(3)

This output proves that the control variable ‘x’ is the same out of the loop scope as its value is 3 when the loop ends.

In **Perl**, the control variable is local for the loop and can’t be accessible outside the loop if they are declared with ‘my’ keyword. Otherwise, they can be used outside the loop.

#Declaration with 'my'

for (my $x = 0; $x < 3; $x++)

{

print "Inside the loop: $x\n";

}

print "Outside the loop: $x\n";

This example program compiles properly and executes the following output:

Inside the loop: 0

Inside the loop: 1

Inside the loop: 2

Outside the loop:

As it is seen from the output, the last statement is executed without printing the ‘$x’ value because the variable ‘$x’ is unkonwn at that point. Its reason is that the control variable can’t be used outside the loop scope if it is declared when the loop starts.

#Declaration without 'my'

for ( $x = 0; $x < 3; $x++)

{

print "Inside the loop: $x\n";

}

print "Outside the loop: $x\n";

This example program compiles properly and executes the following output:

Inside the loop: 0

Inside the loop: 1

Inside the loop: 2

Outside the loop: 3

This outout shows that the control variable can be accessible outside the loop when it is declared wtihout the ‘my’ keyword.

1. **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?**

In **Python**, the control variable can’t be changed in the loop. The parameters can be changed they are specified with a list, however they can’t be changed if they are specified using range().

#Trying to change the control Variable

for x in range(3):

print ("Original: ", x)

x = x + 2

print ("After change:", x)

This example program compiles properly and executes the following output:

Original:  0

After change: 2                                                     Original:  1                                                        After change: 3

Original:  2

After change: 4

This output shows that even if the control variable is tried to be changed in the loop, the variable acts like it was never changed in the next iteration.

#Trying to change the parameter specified with range()

integers = [0, 1, 2]

for x in range(len(integers)):

print (x)

integers.pop()

This example program compiles properly and executes the following output:

0

1

2

This output shows that even if the integers list is changed in the loop, its length is remains same and the loop iterates according to its first value.

#Trying to change the parameter specified with a list

integers = [0, 1, 2]

for x in integers:

print(x)

integers.pop()

This example program compiles properly and executes the following output:

0

1

This output proves that the change of the list parameter affects the loop control because the loop couldn’t do its last iteration after the element of 2 is popped.

In **Javascript**, both the control variable and loop parameters can be changed in the loop.

//Trying to change control variable

for (var x = 0; x < 5; x++) {

document.write(x + "<br>");

x = x + 2;

}

This example program compiles properly and executes the following output:

0

3

This output shows that the control variable can be changed in the loop and the condition check occurs according to the new ‘x’ value.

//Trying to change loop parameter

var limit = 3;

for (var x = 0; x < limit; x++)

{

document.write(x + "<br>");

limit = 5;

}

This example program compiles properly and executes the following output:

0  
1  
2  
3  
4

This example shows that terminal value of the loop variable can be changed in the loop and affects the loop control.

In **PHP**, both the control variable and loop parameters can be changed in the loop.

//Trying to change control variable

for ($x = 0; $x < 5; $x++) {

var\_dump($x);

print "<br>";

$x = $x + 2;

}

This example program compiles properly and executes the following output:

int(0)   
int(3)

This output shows that the control variable can be changed in the loop and the condition check occurs according to the new ‘$x’ value.

//Trying to change loop parameter

$limit = 3;

for ($x = 0; $x < $limit; $x++) {

var\_dump($x);

print "<br>";

$limit = 5;

}

This example program compiles properly and executes the following output:

int(0)   
int(1)   
int(2)   
int(3)   
int(4)

This example shows that terminal value of the loop variable can be changed in the loop and affects the loop control.

In **Pearl**, both the control variable and loop parameters can be changed in the loop.

#Trying to change control variable

for (my $x = 0; $x < 5; $x++)

{

print "$x\n";

$x = $x + 2;

}

This example program compiles properly and executes the following output:

0

3

This output shows that the control variable can be changed in the loop and the condition check occurs according to the new ‘$x’ value.

#Trying to change loop parameter

$limit = 3;

for (my $x = 0; $x < $limit; $x++)

{

print "$x\n";

$limit = 5;

}

This example program compiles properly and executes the following output:

0

1

2

3

4

This example shows that terminal value of the loop variable can be changed in the loop and affects the loop control.

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

(The results of the example programs in the section 3 also proves the answers of the questions in section 4. Therefore, I thought it is unnecessary to give extra example programs in the section 4.)

In **Python**, the loop parameters are evaluated only once if the parameters specified with range() as the change of loop parameters doesn’t affect the loop control. However, the loop parameters are evaluated for every iteration once if the parameters specified with a list because the changes of loop parameters affect the loop control. They have to be evaluated in each iteration in order to affect the loop control.

In **Javascript, PHP, and Perl**, the loop parameters are evaluated once for every iteration because any change of loop parameters affect the loop control. They have to be evaluated in each iteration in order to affect the loop control.

**Discussion**

In my opinion, Perl provides the best counter-controlled loop. The reasons are that using these loops are more flexible in Perl because the loop parameters are evaluated in each iteration so the control variable and loop parameters can be changed in loop and this affects the loop control. Also, the counter-controlled loops of Perl gives the user two options for the scoping of the control variable which is very important to avoid some unwanted situations after the loop ends. Finally, the control variable can be string and controlling string in the loop can be very useful sometimes according to the program.

**References**

Sebesta, R. W., Mukherjee, S., & Bhattacharjee, A. K. (2016). *Concepts of programming languages*. Boston: Pearson Prenitce Hall.