CS 315 Programming Languages

Homework 2 Report

Iteration Based on Data Structures in C, Go, Javascript, PHP, Python, and Rust

İlke Doğan 21702215 Section 1

TABLE OF CONTENTS

Python Language

Code Explanation:

In Part 1, the iterations in Python language are tested one by one. In Python language; While, For, and Nested for algorithms are designed to learn their syntax and working behaviour. In a while, until turning false the loop continues working. In for and nested for, same condition with while but it continues looping until the condition ‘i’ in range is satisfied. In Part 2, set, tuple, dictionary, and List are tested. In tuple, firstly, tuple is printed, then the behaviour for negative array numbers is examined. List is printed, then append is tested by using a nested loop. Dictionary is printed. Set is printed, then tuple is added by using update() in nested for. In part 3, In python for, works like for-each and changing the value does not change the value inside for argument ‘ i ’, however, unlike for, while is affected by it.

Compiler URL:
https://www.programiz.com/python-programming/online-compiler/

1. *Iteration statements provided,*

In python, the iterations are tested one by one. In Python language; While, For, and Nested For loop are available and these iterations are examined under different algorithms.

Code:

```
x = 1
boolvar = True
while ( boolvar ):
    if( x == 6 ):
        boovar = False
        print(" Reached  '6'")
        break
    x = x + 1
    print(" Print Until Finding '6' and now the value is: ", x)

# For Loop
print( "\n")
print( " ----     FOR LOOP IN PYTHON     ---- \n")


for i in range(1,10):
    if( i == 6 ):
        print(" Reached  '6'")
    print(" Print Until Finding '6' and now the value is: ", i)

# Nested For Loop
print( "\n")
print( " ----  NESTED FOR LOOP IN PYTHON  ---- \n")

x = 0
k = 0
for i in range(1,20,2):
    for k in range(k,30):
        x = x + 1
print(" Reached  '16' for", x, "times \n" )
```

Output:

```
1. ITERATION STATEMENTS
 ----     WHILE LOOP IN PYTHON  ----
 Print Until Finding '6' and now the value is:  2
 Print Until Finding '6' and now the value is:  3
 Print Until Finding '6' and now the value is:  4
 Print Until Finding '6' and now the value is:  5
 Print Until Finding '6' and now the value is:  6
 Reached  '6'
----     FOR LOOP IN PYTHON     ----
 Print Until Finding '6' and now the value is:  1
 Print Until Finding '6' and now the value is:  2
 Print Until Finding '6' and now the value is:  3
 Print Until Finding '6' and now the value is:  4
```

Print Until Finding '6' and now the value is: 5
Reached '6'
Print Until Finding '6' and now the value is: 6
Print Until Finding '6' and now the value is: 7
Print Until Finding '6' and now the value is: 8
Print Until Finding '6' and now the value is: 9

---- NESTED FOR LOOP IN PYTHON ----
Reached '16' for 39 times

2. Data structures suitable for iteration

In python, data structures with suitable iterations are tested. Built-in data structures in python are set, tuple, dictionary, and List which helps programmers to resolve the problems more faster and in an efficient way .

Code:

```
print( " ----    1. TUPPLE IN PYTHON  ---- \n")
print(" Printing the tupple in tupple")
tuppleExample = ("22", '06', "1999",('Happy', 'Birthday'))
for i in tuppleExample:
    print(i)
print(tuppleExample)

print(" Finding the behaviour for negative array numbers, and the
sequence is:")
num = -4
num2 = -7
x = 1
while ( num < 2 ):
    num = num + 1
    num2 = num2 + 1
    x = x + 1
    print( x,". try: ",tuppleExample[num])
    print( "The couple: ",tuppleExample[num: ])

print( " \n----    2. LIST IN PYTHON  ---- \n")
print(" Printing the List in List")
listExample = ["22", '06', "1999",['Happy', 'Birthday'],['i','l', 'k',
'e']]
for i in listExample:
    print(i)
print(listExample)
print(" Finding every elements in the list by using Nested For ")

listExample2 = []
print("\n✈ Before Seperating every item list in list: ")
print(listExample)
for i in listExample:
    for k in i:
        listExample2.append(k)
```

```
print(" \n✈ After Seperating every item list in list: ")
print(listExample2)

print( "\n----    3. DICTONARY IN PYTHON  ---- \n")
dictionaryExample = {"Day":22, "Mounth":'06', "Year":1999}
for i in dictionaryExample:
    print(i, " = ",dictionaryExample[i] )
print( "\n----    4. SET IN PYTHON  ---- \n")
print(" ✈ Before adding tupple into Set: ")

setExample = {"22", '06', "1999"}
for i in setExample:
    print(i)
print(setExample)

print(" \n✈  After Adding tupple into Set: ")
for i in listExample:
    for k in i:
        setExample.update(k)
print(setExample)

tempSet = setExample
print(" \n✈  After Deleting  first 10 numbers from Set: ")
b = 0
while ( b < 10 ):
    b = b + 1
    tempSet.pop()
print(tempSet)
```

Output:

---- 1. TUPPLE IN PYTHON ----

Printing the tupple in tupple
22
06
1999
('Happy', 'Birthday')
('22', '06', '1999', ('Happy', 'Birthday'))
Finding the behaviour for negative array numbers, and the sequence is:
2 . try: 06
The couple: ('06', '1999', ('Happy', 'Birthday'))
3 . try: 1999
The couple: ('1999', ('Happy', 'Birthday'))
4 . try: ('Happy', 'Birthday')
The couple: (('Happy', 'Birthday'),)
5 . try: 22
The couple: ('22', '06', '1999', ('Happy', 'Birthday'))
6 . try: 06
The couple: ('06', '1999', ('Happy', 'Birthday'))
7 . try: 1999
The couple: ('1999', ('Happy', 'Birthday'))

---- 2. LIST IN PYTHON ----

Printing the List in List
22
06
1999
['Happy', 'Birthday']
['i', 'l', 'k', 'e']
['22', '06', '1999', ['Happy', 'Birthday'], ['i', 'l', 'k', 'e']]
Finding every elements in the list by using Nested For

✈ Before Seperating every item list in list:
['22', '06', '1999', ['Happy', 'Birthday'], ['i', 'l', 'k', 'e']]

✈ After Seperating every item list in list:
['2', '2', '0', '6', '1', '9', '9', '9', 'Happy', 'Birthday', 'i', 'l', 'k', 'e']

---- 3. DICTONARY IN PYTHON ----

Day = 22
Mount = 06
Year = 1999

---- 4. SET IN PYTHON ----

✈ Before adding tupple into Set:
22
1999
06
{'22', '1999', '06'}

✈ After Adding tupple into Set:
{'r', 'i', 'l', 'a', '22', '6', 'H', 'e', 'y', '1999', '06', '2', 'h', 'B', 't', '9', 'd', 'p', '0', '1', 'k'}

✈ After Deleting first 10 numbers from Set:
{'06', '2', 'h', 'B', 't', '9', 'd', 'p', '0', '1', 'k'}

3. The way the next item is accessed

In python, to learn the way the next item, While and For loop is used. Inside for loop has no effect on expression in for, however, while condition is dependent inside of it.

Code:

```
x = 0
while ( x < 11 ):
    if( x == 6):
        print(" Reached  '6'")
        x += 2
    x = x + 1
    print(" Turn ", x, " times")
print("After execution x's value:", x)
```

```
x = 0
for x in range (0,10):
    if x == 6:
        print(" Reached  '6'")
        x += 2
    print(" Turn ", x, " times")
print("After execution x's value: ", x)
```

Output:

```
 Turn  1  times
 Turn  2  times
 Turn  3  times
 Turn  4  times
 Turn  5  times
 Turn  6  times
 Reached  '6'
 Turn  9  times
 Turn  10  times
 Turn  11  times
After execution x's value:  11

 Turn  0  times
 Turn  1  times
 Turn  2  times
 Turn  3  times
 Turn  4  times
 Turn  5  times
 Reached  '6'
 Turn  8  times
 Turn  7  times
 Turn  8  times
 Turn  9  times
After execution x's value:  9
```

Javascript Language

Code Explanation:

In Part 1, the iterations in JavaScript language are tested one by one. In JavaScript language; While, For, Do-while, For/in, and For/of loop algorithms are designed to learn their syntax and working behaviour. In a while, until turning false the loop continues working. In for and nested for, same condition with while but it continues looping until the condition ‘i’ in range is satisfied. In Do-While Loop, until k < 17 that are incremented by 2, the code is printed the values. In for/in&of Loop,

array is printed. In Part 2, set, Array,and map are tested. In set, elements are added in for loop and by using has() method some of elements in set check whether available or not. In Array, elements are added in a for loop. In part 3, in JavaScript; for works like for-each and changing the value does not change the value inside for argument ‘ i ’, however, unlike for, while is affected by it.

Compiler URL: https://js.do/

1. Iteration statements provided,

In JavaScript, the iterations are tested one by one. In JavaScript language; While, For, Do-while, For/in, and For/of loop are available and these iterations are examined under different algorithms.

Code:

```
x = 1
        var boolvar = true;
        while( boolvar ){
            if( x == 6 ){
                boolvar = false
                subheading += " Reached  '6'<br>"
                break
            }
            x = x + 1
            subheading += " Print Until Finding '6' and now the value
is: " + x + "<br>"
        }
        // For Loop
        subheading += "<br> ☯ FOR LOOP IN JAVASCRIPT ☯ <br>"
        i = 0;
        for( i= 0; i < 11 ; i++){
            if( i == 6 ){
                subheading += " Reached  '6'<br>"
            }
            subheading += " Print Until Finding '6' and now the value
is: " + i + "<br>"
        }
        // Do-While Loop
        subheading += " <br> ☯ DO-WHILE LOOP IN JAVASCRIPT ☯ <br>"
        k = 0;
        do{
            k += 2;
            subheading +="Print Until Finding '16' for "+ k + " times
increment by 2 <br>"
        } while( k < 17 );
        // for/in Loop
        subheading += "<br> ☯ FOR-IN LOOP IN JAVASCRIPT ☯<br>"
        var arrayNum = {day:22, month:06, year:1999};
        var x;
        for (x in arrayNum) {
            subheading += "Print array: "+ arrayNum[x] + "<br>";
```

```
    }
    // for/of Loop

    subheading += "<br> ☯ FOR-OFF LOOP IN JAVASCRIPT ☯ <br>"
    var birthday = ["22", "june", "1999"];
    for (var x of birthday) {
        subheading += "Print array: "+ x + "<br>";
    }
```

Output:

□□□□□□□□□□□□ 1. ITERATION STATEMENTS □□□□□□□□□□□□
☯ WHILE LOOP IN JAVASCRIPT ☯
Print Until Finding '6' and now the value is: 2
Print Until Finding '6' and now the value is: 3
Print Until Finding '6' and now the value is: 4
Print Until Finding '6' and now the value is: 5
Print Until Finding '6' and now the value is: 6
Reached '6'

☯ FOR LOOP IN JAVASCRIPT ☯
Print Until Finding '6' and now the value is: 0
Print Until Finding '6' and now the value is: 1
Print Until Finding '6' and now the value is: 2
Print Until Finding '6' and now the value is: 3
Print Until Finding '6' and now the value is: 4
Print Until Finding '6' and now the value is: 5
Reached '6'
Print Until Finding '6' and now the value is: 6
Print Until Finding '6' and now the value is: 7
Print Until Finding '6' and now the value is: 8
Print Until Finding '6' and now the value is: 9
Print Until Finding '6' and now the value is: 10

☯ DO-WHILE LOOP IN JAVASCRIPT ☯
Print Until Finding '16' for 2 times increment by 2
Print Until Finding '16' for 4 times increment by 2
Print Until Finding '16' for 6 times increment by 2
Print Until Finding '16' for 8 times increment by 2
Print Until Finding '16' for 10 times increment by 2
Print Until Finding '16' for 12 times increment by 2
Print Until Finding '16' for 14 times increment by 2
Print Until Finding '16' for 16 times increment by 2
Print Until Finding '16' for 18 times increment by 2

☯ FOR-IN LOOP IN JAVASCRIPT ☯
Print array: 22
Print array: 6

Print array: 1999

☯ FOR-OFF LOOP IN JAVASCRIPT ☯
Print array: 22
Print array: june
Print array: 1999

2. Data structures suitable for iteration

In JavaScript, data structures with suitable iterations are tested. Built-in data structures in JavaScript are set, Array,and map which helps programmers to resolve the problems more faster and in an efficient way. Array and set data structures are experienced at the given code below.

Code:

```
subheading += "<br> ☯ 1. ARRAY IN JAVASCRIPT ☯ <br>"
    var arrayExample = [22, 06, 1999];
    for( i= 0; i < 4 ; i++){
            subheading +=  i + ". element is: " + arrayExample[i] +"
<br>";
        }
    subheading += "Array length is: " + arrayExample.length +" <br>";
    subheading += "<br> ☯ 2. SET IN JAVASCRIPT ☯ <br>"
    let setExample = new Set()
    for( i= 1; i < 10 ; i++){
            subheading +=  i + ". element is added <br>"
            setExample.add(i)
        }
    subheading += "The set is created: [ " + [...setExample] +
"]<br><br>"
    x = 0
    subheading +=" Checking availability of the elements in the set:
<br>"
    while ( x < 15){
        if( setExample.has(x) == true ){
            subheading += x +" :)  is available <br>"
        }
        else{
            subheading += x +" :(  not available <br>"
        }
        x+=2
    }
```

Output:

☯ 1. ARRAY IN JAVASCRIPT ☯
0. element is: 22
1. element is: 6
2. element is: 1999
3. element is: undefined
Array length is: 3

☯ 2. SET IN JAVASCRIPT ☯
1. element is added
2. element is added
3. element is added
4. element is added
5. element is added
6. element is added
7. element is added
8. element is added
9. element is added
The set is created: [1,2,3,4,5,6,7,8,9]

Checking availability of the elements in the set:
0 :(not available
2 :) is available
4 :) is available
6 :) is available
8 :) is available
10 :(not available
12 :(not available
14 :(not available

3. The way the next item is accessed

In JavaScript, to learn the way the next item, While and For loop is used. Inside for loop has no effect on expression in for, however, while condition is dependent inside of it.

Code:

```
subheading += "<br> ☯ WHILE LOOP FOR NEXT ITEM TEST IN JAVASCRIPT ☯
<br>"
x = 0
        while( x < 11  ){
            if( x == 6 ){
                subheading += " Reached  '6'<br>"
                x += 2
            }
            subheading += "Turn " + x + " times<br>"
            x = x + 1
        }
        subheading += " It should be printed 10 times if not next item
related with what is written inside loop: " + x + "<br>"
        // For Loop
        subheading += "<br> ☯ FOR LOOP FOR NEXT ITEM TEST IN
JAVASCRIPT ☯ <br>"
        for( i= 0; i < 11 ; i++){
            if( i == 6 ){
                subheading += " Reached  '6'<br>"
                x += 2
```

```
            }
            subheading += " Turn " + i + " times<br>"
        }
        subheading += " It should be printed 10 times if not next item
related with what is written inside loop: " + x + "<br>"
```

Output:

☯ WHILE LOOP FOR NEXT ITEM TEST IN JAVASCRIPT ☯
Turn 0 times
Turn 1 times
Turn 2 times
Turn 3 times
Turn 4 times
Turn 5 times
Reached '6'
Turn 8 times
Turn 9 times
Turn 10 times
It should be printed 10 times if not next item related with what is written inside loop: 11

☯ FOR LOOP FOR NEXT ITEM TEST IN JAVASCRIPT ☯
Turn 0 times
Turn 1 times
Turn 2 times
Turn 3 times
Turn 4 times
Turn 5 times
Reached '6'
Turn 6 times
Turn 7 times
Turn 8 times
Turn 9 times
Turn 10 times
It should be printed 10 times if not next item related with what is written inside loop: 13

C Language

Code Explanation:

In Part 1, the iterations in C language are tested one by one. In C language; While, For, and Do-while algorithms are designed to learn their syntax and working behaviour. In a while, until turning false the loop continues working. In for and nested for, same condition with while but it continues looping until the condition ‘i’ in range is

satisfied. In Do-While Loop, until k < 17 that are incremented by 2, the code is printed the values.In Part 2, Array is tested for int and char arrays. One of them do-while & int Array is used and sum is calculated. The other one is used and char Array is used and printed. In part 3, In C language; for works like for-each and changing the value does not change the value inside for argument ‘ i ’, however, unlike for, while is affected by it.

Compiler URL:
https://www.programiz.com/c-programming/online-compiler/

1. Iteration statements provided,

In C Language, the iterations are tested one by one. In C Language; While, For, and Do-while are available and these iterations are examined under different algorithms.

Code:

```
int x = 1;
bool boolvar = true;
while( boolvar ){
    if( x == 6 ){
        boolvar = false;
        printf("Reached  '6'");
        break;
    }
    x = x + 1;
    printf("Print Until Finding '6' and now the value is: %d", x);
    printf("\n");
}

// For Loop
printf( "\n");
printf( " ----     FOR LOOP IN C     ---- \n");

for( int i= 0; i< 11 ; i++){
    if( i == 6 ){
        printf("Reached  '6'");
    }
    printf("Print Until Finding '6' and now the value is: %d", i);
    printf("\n");
}

// Do-While Loop
printf( " ----   DO-WHILE LOOP IN C  ---- \n");
int k = 0;
do{
    k+=2;
    printf("Print Until Finding '16' for %d", k);
```

```
    printf("times increment by 2 \n" );
} while( k < 17 );
```

Output:

1. ITERATION STATEMENTS
---- WHILE LOOP IN C ----
Print Until Finding '6' and now the value is: 2
Print Until Finding '6' and now the value is: 3
Print Until Finding '6' and now the value is: 4
Print Until Finding '6' and now the value is: 5
Print Until Finding '6' and now the value is: 6
Reached '6'
---- FOR LOOP IN C ----
Print Until Finding '6' and now the value is: 0
Print Until Finding '6' and now the value is: 1
Print Until Finding '6' and now the value is: 2
Print Until Finding '6' and now the value is: 3
Print Until Finding '6' and now the value is: 4
Print Until Finding '6' and now the value is: 5
Reached '6'Print Until Finding '6' and now the value is: 6
Print Until Finding '6' and now the value is: 7
Print Until Finding '6' and now the value is: 8
Print Until Finding '6' and now the value is: 9
Print Until Finding '6' and now the value is: 10
---- DO-WHILE LOOP IN C ----
Print Until Finding '16' for 2times increment by 2
Print Until Finding '16' for 4times increment by 2
Print Until Finding '16' for 6times increment by 2
Print Until Finding '16' for 8times increment by 2
Print Until Finding '16' for 10times increment by 2
Print Until Finding '16' for 12times increment by 2
Print Until Finding '16' for 14times increment by 2
Print Until Finding '16' for 16times increment by 2
Print Until Finding '16' for 18times increment by 2

2. Data structures suitable for iteration

In C Language, data structures with suitable iterations are tested. Built-in data structures in C language is the Array that helps programmers to resolve the problems more faster and in an efficient way. Array is experienced at the given code below.

Code:

```
printf( " ----       ARRAY IN C  ---- \n");
// For Loop
char arrayExample[] = {'2', '6', '9','i', 'd'};
for( int i= 0; i < 5 ; i++){
```

```
    printf("%d. element is:  %c",i+1, arrayExample[i]);
    printf("\n");
}
// Do-While Loop
printf( " ----   DO-WHILE LOOP IN C  ---- \n");
int arrayExample2[] = {2, 2, 6, 9};
int sum = 0;
k= 0;
do{
    sum += arrayExample2[k];
    k++;
} while( k < 4 );
printf("Sum for the every item {2, 2, 6, 9} in the array= %d", sum);
```

Output:

---- ARRAY IN C ----
1. element is: 2
2. element is: 6
3. element is: 9
4. element is: i
5. element is: d

---- DO-WHILE LOOP IN C ----
Sum for the every item {2, 2, 6, 9} in the array= 19

3. The way the next item is accessed

In C Language, to learn the way the next item, While and For loop is used. Inside for loop has no effect on expression in for, however, while condition is dependent inside of it.

Code:

```
printf( " ----     WHILE LOOP IN C     ---- \n");
// While Loop
x = 0;
while ( x < 11 ){
    if( x == 6 ){
        printf("Reached  '6'\n");
        x += 2;
    }
    x = x + 1;
    printf(" Turn %d times", x);
    printf("\n");
}
// For Loop
printf( "\n");
printf( " ----     FOR LOOP IN C     ---- \n");
for( int i= 0; i< 11 ; i++){
    if( i == 6 ){
        printf("Reached  '6'\n");
        x += 2;
    }
    printf(" Turn %d times", i);
    printf("\n");
```

```
}
```

Output:

```
----     WHILE LOOP IN C     ----
 Turn 1 times
 Turn 2 times
 Turn 3 times
 Turn 4 times
 Turn 5 times
 Turn 6 times
Reached  '6'
 Turn 9 times
 Turn 10 times
 Turn 11 times

 ----     FOR LOOP IN C     ----
 Turn 0 times
 Turn 1 times
 Turn 2 times
 Turn 3 times
 Turn 4 times
 Turn 5 times
Reached  '6'
 Turn 6 times
 Turn 7 times
 Turn 8 times
Turn 9 times
 Turn 10 times
```

Go Language

Code Explanation:

In Part 1, the iterations in Go language are tested one by one. In Go language; While, For, and Do-while algorithms are designed to learn their syntax and working behaviour. In a while, until turning false the loop continues working and the difference here from the other languages is while is named as for in Go. In for and for-each, same condition with while but it continues looping until the condition ‘i’ in range is satisfied, run-out statement provided with break. In Part 2, Array and Slice are tested. In array, by using for, int Array is printed. In slice, by using for, String Array is printed. In part 3, In Go; for works like for-each and changing the value does

not change the value inside for argument ‘ i ’, however, unlike for, while is affected by it.

Compiler URL:
https://www.onlinegdb.com/online_go_compiler

1. Iteration statements provided,

In Go Language, the iterations are tested one by one. In Go Language; While, For, and Do-while are available and these iterations are examined under different algorithms.

Code:

```
// While Loop
    fmt.Println(" 1. ITERATION STATEMENTS \n")
    fmt.Println(" ----   WHILE LOOP IN GO  ---- \n")
    var boolvar bool  = true
    n:= 1
    for boolvar {
        if  n == 6 {
            fmt.Println("Reached  '6' and program is broken")
            break
        }
        n = n + 1
        fmt.Println("Print Until Finding '6' and now the value is: ",
n)
    }
    fmt.Println("\n")

    // THREEE COMPONENT LOOP

    fmt.Println(" ----   THREE COMPONENT LOOP IN GO  ---- \n")
    for i := 0; i < 11; i++ {
        if  i == 6 {
            fmt.Println("Reached  '6' and program is broken")
            break
        }
        fmt.Println("Print Until Finding '6' and now the value is:", i)
    }
    fmt.Println("\n")

    // For-each range LOOP

    fmt.Println(" ----   FOR-EACH RANGE LOOP IN GO  ---- \n")
    arrayNum := []int64{ 22, 06, 1999}
    for i, s := range arrayNum {
        fmt.Println("Print array: (Array Place:",i,") (Number: ", s,
")")
    }
    fmt.Println("\n")
```

Output:

```
1. ITERATION STATEMENTS

 ----   WHILE LOOP IN GO  ----

Print Until Finding '6' and now the value is:  2
Print Until Finding '6' and now the value is:  3
Print Until Finding '6' and now the value is:  4
Print Until Finding '6' and now the value is:  5
Print Until Finding '6' and now the value is:  6
Reached  '6' and program is broken


 ----   THREE COMPONENT LOOP IN GO  ----

Print Until Finding '6' and now the value is: 0
Print Until Finding '6' and now the value is: 1
Print Until Finding '6' and now the value is: 2
Print Until Finding '6' and now the value is: 3
Print Until Finding '6' and now the value is: 4
Print Until Finding '6' and now the value is: 5
Reached  '6' and program is broken


 ----   FOR-EACH RANGE LOOP IN GO  ----

Print array: (Array Place: 0 ) (Number:  22 )
Print array: (Array Place: 1 ) (Number:  6 )
Print array: (Array Place: 2 ) (Number:  1999 )
```

2. Data structures suitable for iteration

In Go Language, data structures with suitable iterations are tested. Built-in data structures in Go language are the Array, Slice, Map, and Struct that helps programmers to resolve the problems more faster and in an efficient way. Array and Slice are experienced at the given code below.

Code:

```
fmt.Println(" ----   ARRAY IN GO  ---- \n")
    arrayExample := [6]int{ 22, 6 , 1 , 9 , 9 , 9}
    for i := 0; i < 6; i++ {
        fmt.Println(i+1, ". element is:", arrayExample[i])
    }
    fmt.Println("Whole array is:", arrayExample)
```

```
    fmt.Println("\n")
    fmt.Println(" ----   SLICE IN GO  ---- \n")
    stringExample := [4]string{"Happy", "Birthday", "ilke", "dogan"}
    for i := 0; i < 3; i++ {
        fmt.Println(stringExample[i])
    }
```

Output:

```
----   ARRAY IN GO  ----
1 . element is: 22
2 . element is: 6
3 . element is: 1
4 . element is: 9
5 . element is: 9
6 . element is: 9
Whole array is: [22 6 1 9 9 9]

 ----   SLICE IN GO  ----
Happy
Birthday
ilke
```

3. The way the next item is accessed

In Go Language, to learn the way the next item, While and For loop is used. Inside for loop has no effect on expression in for, however, while condition is dependent inside of it.

Code:

```
a:= 0
    for a < 11 {
        if  a == 6 {
            fmt.Println("Reached  '6'")
            a = a + 2
        }
        a= a + 1
        fmt.Println(" Turn ", a , " times")
    }
    fmt.Println("\n")

    // THREEE COMPONENT ( FOR ) LOOP

    fmt.Println(" ----   THREE COMPONENT LOOP IN GO  ----" )
    for i := 1; i < 11; i++ {
        if  i == 6 {
            fmt.Println("Reached  '6'")
            n = n + 2
        }
        fmt.Println(" Turn ", i , " times")
    }
```

Output:

```
----   WHILE LOOP IN GO  ----
 Turn  1  times
 Turn  2  times
 Turn  3  times
 Turn  4  times
 Turn  5  times
 Turn  6  times
Reached  '6'
 Turn  9  times
 Turn  10  times
 Turn  11  times


 ----   THREE COMPONENT LOOP IN GO  ----
 Turn  1  times
 Turn  2  times
 Turn  3  times
 Turn  4  times
 Turn  5  times
Reached  '6'
 Turn  6  times
 Turn  7  times
 Turn  8  times
 Turn  9  times
 Turn  10  times
```

Rust Language

Code Explanation:

In Part 1, the iterations in Rust language are tested one by one. In Rust language; While, For, and Loop are designed to learn their syntax and working behaviour. In a while, until turning false the loop continues working and the difference here from the other languages is while is named as for in Go. In loop, same condition with while but it continues looping until the condition ‘i’ in range is satisfied, run-out statement provided with break. In for, it goes 1 to 10 like python language (x in range) and prints all the numbers. In Part 2, Arrays and vectors are tested. In array, by using iter(), array elements are reached and printed. In part 3, in Rust ; while condition x value is affected by inside x change.

Compiler URL: https://play.rust-lang.org/

1. Iteration statements provided,

In Rust Language, the iterations are tested one by one. In Rust Language; While, For, and Loop are available and these iterations are examined under different algorithms.

Code:

```
let mut x = 0;
     while x < 6 {
       x = x + 1;
       println!("{}= now the value and Print Until Finding '6' ", x);
       }
       println!("Reached  '6'");
     println!( "\n");
     // Loop
     println!( "\n");
     println!( " 1. ITERATION STATEMENTS \n");
     println!( " ----   LOOP IN RUST  ---- \n");
     let mut k = 0;
     loop {
       if k == 16 {
           println!("'16' is found!");
           break;
       }
       println!("{}= Print Until Finding '16' for", k);
       k = k + 1;
     }
       // FOR Loop
   println!( "\n");
   println!( " 1. ITERATION STATEMENTS \n");
   println!( " ----   FOR LOOP IN RUST  ---- \n");

   for x in 1..10 {
       println!("{}= Print Until reaching 11", x);
     }
```

Output:

```
1. ITERATION STATEMENTS

 ----   WHILE LOOP IN RUST  ----

1= now the value and Print Until Finding '6'
2= now the value and Print Until Finding '6'
3= now the value and Print Until Finding '6'
4= now the value and Print Until Finding '6'
5= now the value and Print Until Finding '6'
6= now the value and Print Until Finding '6'
```

```
Reached  '6'
 ----   LOOP IN RUST  ----

0= Print Until Finding '16' for
1= Print Until Finding '16' for
2= Print Until Finding '16' for
3= Print Until Finding '16' for
4= Print Until Finding '16' for
5= Print Until Finding '16' for
6= Print Until Finding '16' for
7= Print Until Finding '16' for
8= Print Until Finding '16' for
9= Print Until Finding '16' for
10= Print Until Finding '16' for
11= Print Until Finding '16' for
12= Print Until Finding '16' for
13= Print Until Finding '16' for
14= Print Until Finding '16' for
15= Print Until Finding '16' for
'16' is found!
 ----   FOR LOOP IN RUST  ----

1= Print Until reaching 11
2= Print Until reaching 11
3= Print Until reaching 11
4= Print Until reaching 11
5= Print Until reaching 11
6= Print Until reaching 11
7= Print Until reaching 11
8= Print Until reaching 11
9= Print Until reaching 11
```

2. Data structures suitable for iteration

In Rust Language, data structures with suitable iterations are tested. Built-in data structures in Rust language help programmers to resolve the problems more faster and in an efficient way. Array and Tupple are experienced at the given code below.

Code:

```
x = 0;
let arrayExample = [22, 6, 1, 9, 9, 9];
  for i in arrayExample.iter() {
     x += i;
    println!("Array element is: {}", i);
  }
  println!("Array elements' sum is : {}", x);
```

Output:

```
Array element is: 22
```

```
Array element is: 6
Array element is: 1
Array element is: 9
Array element is: 9
Array element is: 9
Array elements' sum is : 56
```

3. The way the next item is accessed

In Rust Language, to learn the way the next item, While loop is used and while condition is dependent inside of it.

Code:

```
let mut x = 0;
   while x < 11 {
      if x == 6 {
        println!(" Reached  '6'");
        x = x + 2;
    }
    println!(" Turn {}", x);
    x = x + 1;
  }
   println!( "It should be printed 10 times if not next item related
with what is written inside loop: {}", x);
```

Output:

```
Turn 0
Turn 1
Turn 2
Turn 3
Turn 4
Turn 5
Reached  '6'
Turn 8
Turn 9
Turn 10
It should be printed 10 times if not next item related with what is
written inside loop: 11
```

PHP Language

Code Explanation:

In Part 1, the iterations in PHP language are tested one by one. In PHP language; While, Do-While, For, and For-each are designed to learn their syntax and working behaviour. In a while, until turning false the loop continues working. In Do-While Loop, until k < 17 that are incremented by 2, the code is printed the

values. In for, same condition with while but it continues looping until the condition ‘i’ in range is satisfied. In for-each, by using array array elements reached and incremented by 4, however, there is a big advantage in for-each statements compared to other languages. It can iterate data structures directly. In Part 2, Arrays are tested. In the first example for array, array(22, 06, 1, 9, 9, 9), is printed. In the second example, the key and value are tested like an array ("day" => 22, "month" => 6, "year"   => 1999), and this array is printed. In part 3, in PHP; for works like for-each and changing the value does not change the value inside for argument ‘ i ’, however, unlike for, while is affected by it.

Compiler URL:
https://paiza.io/projects/H4yZEt7JgEp2onRociF6Gw

1. Iteration statements provided,

In PHP Language, the iterations are tested one by one. In PHP Language; While, Do-While, For, and For-each are available and these iterations are examined under different algorithms.

Code:

```
// While Loop
    $boolTrue1 = true;
    printf( "\n");
    printf( " 1. ITERATION STATEMENTS \n");
    printf( " ----   WHILE LOOP IN PHP  ---- \n");
    $x = 1;
   while($boolTrue1) {
   if ( $x == 6){
       printf( "Reached  '6'\n");
       break;
       }
   $x++;
   printf("Print Until Finding '6' and now the value is: %d", $x);
   printf("\n");
   }

    // Do-While Loop
    printf( "\n");
    printf( " ----   DO-WHILE LOOP IN PHP  ---- \n");
    $k = 0;
    do{
        $k+=2;
        printf("Print Until Finding '16' for %d", $k);
        printf("times increment by 2 \n" );
    } while( $k < 17 );
    // For Loop
```

```
    printf( "\n");
    printf( " ----   FOR LOOP IN PHP  ---- \n");
    for( $i= 0; $i< 11 ; $i++){
       if( $i == 6 ){
           printf("Reached  '6'");
       }
       printf("Print Until Finding '6' and now the value is: %d", $i);
       printf("\n");
   }
    // For-each Loop
    printf( "\n");
    printf( " ----   FOR-EACH LOOP IN PHP  ---- \n");
    $arrayNum = array(22, 06, 1999);
    foreach ($arrayNum as $value) {
       $value = $value + 4;
       printf("Print array: %d", $value);
       printf( "\n");
    }
```

Output:

```
1. ITERATION STATEMENTS
 ----   WHILE LOOP IN PHP  ----
Print Until Finding '6' and now the value is: 2
Print Until Finding '6' and now the value is: 3
Print Until Finding '6' and now the value is: 4
Print Until Finding '6' and now the value is: 5
Print Until Finding '6' and now the value is: 6
Reached  '6'

 ----   DO-WHILE LOOP IN PHP  ----
Print Until Finding '16' for 2times increment by 2
Print Until Finding '16' for 4times increment by 2
Print Until Finding '16' for 6times increment by 2
Print Until Finding '16' for 8times increment by 2
Print Until Finding '16' for 10times increment by 2
Print Until Finding '16' for 12times increment by 2
Print Until Finding '16' for 14times increment by 2
Print Until Finding '16' for 16times increment by 2
Print Until Finding '16' for 18times increment by 2

 ----   FOR LOOP IN PHP  ----
Print Until Finding '6' and now the value is: 0
Print Until Finding '6' and now the value is: 1
Print Until Finding '6' and now the value is: 2
Print Until Finding '6' and now the value is: 3
Print Until Finding '6' and now the value is: 4
Print Until Finding '6' and now the value is: 5
Reached  '6'Print Until Finding '6' and now the value is: 6
Print Until Finding '6' and now the value is: 7
Print Until Finding '6' and now the value is: 8
Print Until Finding '6' and now the value is: 9
Print Until Finding '6' and now the value is: 10

 ----   FOR-EACH LOOP IN PHP  ----
```

```
Print array: 26
Print array: 10
Print array: 2003
```

2. Data structures suitable for iteration

In PHP Language, data structure with suitable iterations are tested. Built-in data structure in Rust language is Array that helps programmers to resolve the problems more faster and in an efficient way. Array is experienced at the given code below.

Code:

```
//Second example
    $arrayNum = array(22, 06, 1, 9, 9, 9);
    foreach ($arrayNum as $value) {
        printf("Print array: %d", $value);
        printf( "\n");
     }
    printf( "\n");
    //First example
     $birth = array ("day"  => 22, "month" => 6, "year"   => 1999 );
     foreach ($birth as $name => $value) {
        printf("Print array: %d", $value);
        printf( "\n");
     }
```

Output:

```
Print array: 22
Print array: 6
Print array: 1
Print array: 9
Print array: 9
Print array: 9

Print array: 22
Print array: 6
Print array: 1999
```

3. The way the next item is accessed

In PHP Language, to learn the way the next item, While and For loop is used. Inside for loop has no effect on expression in for, however, while condition is dependent inside of it.

Code:

```
$x = 1;
     while( $x < 11 ) {
     if ( $x == 6){
```

```
            printf("Reached  '6'\n");
            $x += 2;
        }
     $x++;
     printf(" Turn %d times", $x);
    printf("\n");
     }
     // For Loop
     printf( "\n");
     printf( " ----   FOR LOOP IN PHP  ---- \n");
     for( $i= 0; $i< 11 ; $i++){
        if( $i == 6 ){
            printf("Reached  '6'");
            $i += 2;
        }
        printf(" Turn %d times", $i);
        printf("\n");
    }
```

Output:

```
Turn 2 times
Turn 3 times
Turn 4 times
Turn 5 times
Turn 6 times
Reached  '6'
Turn 9 times
Turn 10 times
Turn 11 times

----   FOR LOOP IN PHP  ----
Turn 0 times
Turn 1 times
Turn 2 times
Turn 3 times
Turn 4 times
Turn 5 times
Reached  '6'
Turn 8 times
Turn 9 times
Turn 10 times
```

Learning Methodology

While I was working on iterations and data structures homework, I have read mostly articles and websites from the internet. I looked at the lecture notes as there are similar examples that homework requires. I examined similar code examples and tried to understand the working steps behind the iterations and finding the next item.

For the 3rd question, I searched next item logic for different applications because I think it needs debugging.

Conclusion (Readability & Writability)

In conclusion, most languages have a for and while loop. In terms of writability, Python is much more comfortable than other languages. However, it has one downside that I perceived is that it is hard to read because of brackets. One of the most unreadable languages is PHP because while I am writing variables, iterations I struggled a little bit. In part 2, while examining Built-in data structures of the 6 languages, it was realized that PHP and C languages have just Arrays as data structures. It is argued that having just one built-in data structure makes languages more easy to read, however, in the long term, it may have downsides to cover all algorithm cases. For the writability of built-in data structures, it is thought that the most convenient language is Python as it has 4 built-in data structures and writability is quite easy to understand and write. The languages who have for-each has an extra advantage, I think since there is a big advantage in for-each statements compared to other languages. It can iterate data structures directly that can be found from the examples given above.