Programming Concepts

Overview

Programs consist of a series of statements; a statement is a single instruction.

Programming constructs are used to control how the statements in a program are executed. There are three constructs:

- Sequence
- Selection
- Repetition

Sequence

This is the simplest programming construct. A sequence is a group of statements that are executed once in the order they appear.

This is an example program that takes the form of a sequence. It takes three numbers, calculates the average and outputs it.

```
Num1 ← INPUT
Num2 ← INPUT
Num3 ← INPUT
Total ← Num1+Num2+Num3
Average ← Total/3
PRINT Average
```

Num1	12
Num2	
Num3	
Total	
Average	

Output	

Sequence

This is the simplest programming construct. A sequence is a group of statements that are executed once in the order they appear.

This is an example program that takes the form of a sequence. It takes three numbers, calculates the average and outputs it.

```
Num1 ← INPUT
Num2 ← INPUT
Num3 ← INPUT
Total ← Num1+Num2+Num3
Average ← Total/3
PRINT Average
```

Num1	12
Num2	9
Num3	
Total	
Average	

Output	

Sequence

This is the simplest programming construct. A sequence is a group of statements that are executed once in the order they appear.

This is an example program that takes the form of a sequence. It takes three numbers, calculates the average and outputs it.

```
Num1 ← INPUT
Num2 ← INPUT
Num3 ← INPUT
Total ← Num1+Num2+Num3
Average ← Total/3
PRINT Average
```

Num1	12
Num2	9
Num3	9
Total	
Average	

Output	

Sequence

This is the simplest programming construct. A sequence is a group of statements that are executed once in the order they appear.

This is an example program that takes the form of a sequence. It takes three numbers, calculates the average and outputs it.

```
Num1 ← INPUT
Num2 ← INPUT
Num3 ← INPUT
Total ← Num1+Num2+Num3
Average ← Total/3
PRINT Average
```

Num1	12
Num2	9
Num3	9
Total	30
Average	

Output	

Sequence

This is the simplest programming construct. A sequence is a group of statements that are executed once in the order they appear.

This is an example program that takes the form of a sequence. It takes three numbers, calculates the average and outputs it.

```
Num1 ← INPUT
Num2 ← INPUT
Num3 ← INPUT
Total ← Num1+Num2+Num3
Average ← Total/3
PRINT Average
```

Num1	12
Num2	9
Num3	9
Total	30
Average	10

Output	

Sequence

This is the simplest programming construct. A sequence is a group of statements that are executed once in the order they appear.

This is an example program that takes the form of a sequence. It takes three numbers, calculates the average and outputs it.

```
Num1 ← INPUT
Num2 ← INPUT
Num3 ← INPUT
Total ← Num1+Num2+Num3
Average ← Total/3
PRINT Average
```

Num1	12
Num2	9
Num3	9
Total	30
Average	10

Output	10

Selection

When selection is used, the path taken through the program changes depending on the answer to a question. **IF statements** are an example.

This is an example program which takes two numbers and either outputs the largest, or outputs “They are equal”.

```
Num1 ← INPUT
Num2 ← INPUT
IF Num1 = Num2
    THEN
        PRINT "They are equal"
    ELSE
        IF Num1 > Num2
            THEN
                PRINT Num1
        ELSE
                PRINT Num2
        ENDIF
ENDIF
```

Num1	10
Num2	

Output	

Selection

When selection is used, the path taken through the program changes depending on the answer to a question. **IF statements** are an example.

This is an example program which takes two numbers and either outputs the largest, or outputs “They are equal”.

```
Num1 ← INPUT
Num2 ← INPUT
IF Num1 = Num2
    THEN
        PRINT "They are equal"
    ELSE
        IF Num1 > Num2
            THEN
                PRINT Num1
        ELSE
                PRINT Num2
        ENDIF
ENDIF
```

Num1	10
Num2	5

Output	

Selection

When selection is used, the path taken through the program changes depending on the answer to a question. **IF statements** are an example.

This is an example program which takes two numbers and either outputs the largest, or outputs “They are equal”.

```
Num1 ← INPUT
Num2 ← INPUT
IF Num1 = Num2
    THEN
        PRINT "They are equal"
    ELSE
        IF Num1 > Num2
            THEN
                PRINT Num1
        ELSE
                PRINT Num2
        ENDIF
ENDIF
```

Num1	10
Num2	5

Output	10

Comparison Operators

A number of different comparison operators can be used when writing IF statements.

=	Equal to
<>	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Repetition

This construct is used to repeat a group of statements, avoiding the need to manually type out statements multiple times.

There are two types of repetition:

- Count controlled
- Condition controlled

Count Controlled Loops

This construct is used to repeat a group of statements a set number of times.

This example program uses a **FOR loop** to output the text "Hello World" five times.

```
FOR i ← 1 TO 5
      PRINT "Hello World"
ENDFOR
```

Index	1

Output

Count Controlled Loops

This construct is used to repeat a group of statements a set number of times.

This example program uses a **FOR loop** to output the text "Hello World" five times.

```
FOR i ← 1 TO 5
      PRINT "Hello World"
ENDFOR
```

Index	1

Output
Hello World

Count Controlled Loops

This construct is used to repeat a group of statements a set number of times.

This example program uses a **FOR loop** to output the text "Hello World" five times.

```
FOR i ← 1 TO 5
      PRINT "Hello World"
ENDFOR
```

Index	2

Output

Hello World

Count Controlled Loops

This construct is used to repeat a group of statements a set number of times.

This example program uses a **FOR loop** to output the text "Hello World" five times.

```
FOR i ← 1 TO 5
      PRINT "Hello World"
ENDFOR
```

Index	2

Output

Hello World
Hello World

Count Controlled Loops

This construct is used to repeat a group of statements a set number of times.

This example program uses a **FOR loop** to output the text "Hello World" five times.

```
FOR i ← 1 TO 5
      PRINT "Hello World"
ENDFOR
```

Index	3

Output

Hello World
Hello World

Count Controlled Loops

This construct is used to repeat a group of statements a set number of times.

This example program uses a **FOR loop** to output the text "Hello World" five times.

```
FOR i ← 1 TO 5
      PRINT "Hello World"
ENDFOR
```

Index	3

Output

Hello World
Hello World
Hello World

Count Controlled Loops

This construct is used to repeat a group of statements a set number of times.

This example program uses a **FOR loop** to output the text "Hello World" five times.

```
FOR i ← 1 TO 5
      PRINT "Hello World"
ENDFOR
```

Index	4

Output

Hello World
Hello World
Hello World

Count Controlled Loops

This construct is used to repeat a group of statements a set number of times.

This example program uses a **FOR loop** to output the text "Hello World" five times.

```
FOR i ← 1 TO 5
      PRINT "Hello World"
ENDFOR
```

Index	4

Output

Hello World
Hello World
Hello World
Hello World

Count Controlled Loops

This construct is used to repeat a group of statements a set number of times.

This example program uses a **FOR loop** to output the text "Hello World" five times.

```
FOR i ← 1 TO 5
      PRINT "Hello World"
ENDFOR
```

Index	5

Output

Hello World
Hello World
Hello World
Hello World

Count Controlled Loops

This construct is used to repeat a group of statements a set number of times.

This example program uses a **FOR loop** to output the text "Hello World" five times.

```
FOR i ← 1 TO 5
      PRINT "Hello World"
ENDFOR
```

Index	5

Output

Hello World
Hello World
Hello World
Hello World
Hello World

Condition Controlled Loops

Repetition can also be used to repeat a group of statements until a condition is met. A **WHILE loop** is an example of a condition controlled loop.

This example program uses a WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
Counter ← 1
WHILE Counter <= 5
      PRINT "Hello World"
      Counter ← Counter + 1
ENDWHILE
```

Counter	1

Output

Condition Controlled Loops

Repetition can also be used to repeat a group of statements until a condition is met. A **WHILE loop** is an example of a condition controlled loop.

This example program uses a WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
Counter ← 1
WHILE Counter <= 5
      PRINT "Hello World"
      Counter ← Counter + 1
ENDWHILE
```

Counter	1

Output
Hello World

Condition Controlled Loops

Repetition can also be used to repeat a group of statements until a condition is met. A **WHILE loop** is an example of a condition controlled loop.

This example program uses a WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
Counter ← 1
WHILE Counter <= 5
      PRINT "Hello World"
      Counter ← Counter + 1
ENDWHILE
```

Counter	2

Output
Hello World

Condition Controlled Loops

Repetition can also be used to repeat a group of statements until a condition is met. A **WHILE loop** is an example of a condition controlled loop.

This example program uses a WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
Counter ← 1
WHILE Counter <= 5
      PRINT "Hello World"
      Counter ← Counter + 1
ENDWHILE
```

Counter	2

Output
Hello World Hello World

Condition Controlled Loops

Repetition can also be used to repeat a group of statements until a condition is met. A **WHILE loop** is an example of a condition controlled loop.

This example program uses a WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
Counter ← 1
WHILE Counter <= 5
      PRINT "Hello World"
      Counter ← Counter + 1
ENDWHILE
```

Counter	3

Output
Hello World Hello World

Condition Controlled Loops

Repetition can also be used to repeat a group of statements until a condition is met. A **WHILE loop** is an example of a condition controlled loop.

This example program uses a WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
Counter ← 1
WHILE Counter <= 5
      PRINT "Hello World"
      Counter ← Counter + 1
ENDWHILE
```

Counter	3

Output
Hello World Hello World Hello World

Condition Controlled Loops

Repetition can also be used to repeat a group of statements until a condition is met. A **WHILE loop** is an example of a condition controlled loop.

This example program uses a WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
Counter ← 1
WHILE Counter <= 5
      PRINT "Hello World"
      Counter ← Counter + 1
ENDWHILE
```

Counter	4

Output
Hello World Hello World Hello World

Condition Controlled Loops

Repetition can also be used to repeat a group of statements until a condition is met. A **WHILE loop** is an example of a condition controlled loop.

This example program uses a WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
Counter ← 1
WHILE Counter <= 5
      PRINT "Hello World"
      Counter ← Counter + 1
ENDWHILE
```

Counter	4

Output
Hello World Hello World Hello World Hello World

Condition Controlled Loops

Repetition can also be used to repeat a group of statements until a condition is met. A **WHILE loop** is an example of a condition controlled loop.

This example program uses a WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
Counter ← 1
WHILE Counter <= 5
      PRINT "Hello World"
      Counter ← Counter + 1
ENDWHILE
```

Counter	5

Output
Hello World Hello World Hello World Hello World

Condition Controlled Loops

Repetition can also be used to repeat a group of statements until a condition is met. A **WHILE loop** is an example of a condition controlled loop.

This example program uses a WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
Counter ← 1
WHILE Counter <= 5
      PRINT "Hello World"
      Counter ← Counter + 1
ENDWHILE
```

Counter	5

Output
Hello World
Hello World
Hello World
Hello World
Hello World

Condition Controlled Loops

Repetition can also be used to repeat a group of statements until a condition is met. A **WHILE loop** is an example of a condition controlled loop.

This example program uses a WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
Counter ← 1
WHILE Counter <= 5
      PRINT "Hello World"
      Counter ← Counter + 1
ENDWHILE
```

Counter	6

Output
Hello World
Hello World
Hello World
Hello World
Hello World

Counting

Repetition is useful for counting up or down to a given number. This algorithm counts up to an inputted number from 0.

```
INPUT Target
Counter ← 0
WHILE Counter <= Target
      PRINT Counter
      Counter ← Counter + 1
ENDWHILE
```

Target	5
Counter	

Output

Counting

Repetition is useful for counting up or down to a given number. This algorithm counts up to an inputted number from 0.

```
INPUT Target
Counter ← 0
WHILE Counter <= Target
      PRINT Counter
      Counter ← Counter + 1
ENDWHILE
```

Target	5
Counter	0

Output

Counting

Repetition is useful for counting up or down to a given number. This algorithm counts up to an inputted number from 0.

```
INPUT Target
Counter ← 0
WHILE Counter <= Target
      PRINT Counter
      Counter ← Counter + 1
ENDWHILE
```

Target	5
Counter	0

Output
0

Counting

Repetition is useful for counting up or down to a given number. This algorithm counts up to an inputted number from 0.

```
INPUT Target
Counter ← 0
WHILE Counter <= Target
      PRINT Counter
      Counter ← Counter + 1
ENDWHILE
```

Target	5
Counter	1

Output
0

Counting

Repetition is useful for counting up or down to a given number. This algorithm counts up to an inputted number from 0.

```
INPUT Target
Counter ← 0
WHILE Counter <= Target
      PRINT Counter
      Counter ← Counter + 1
ENDWHILE
```

Target	5
Counter	1

Output
0 1

Counting

Repetition is useful for counting up or down to a given number. This algorithm counts up to an inputted number from 0.

```
INPUT Target
Counter ← 0
WHILE Counter <= Target
      PRINT Counter
      Counter ← Counter + 1
ENDWHILE
```

Target	5
Counter	2

Output
0 1

Counting

Repetition is useful for counting up or down to a given number. This algorithm counts up to an inputted number from 0.

```
INPUT Target
Counter ← 0
WHILE Counter <= Target
      PRINT Counter
      Counter ← Counter + 1
ENDWHILE
```

Target	5
Counter	2

Output
0 1 2

Counting

Repetition is useful for counting up or down to a given number. This algorithm counts up to an inputted number from 0.

```
INPUT Target
Counter ← 0
WHILE Counter <= Target
      PRINT Counter
      Counter ← Counter + 1
ENDWHILE
```

Target	5
Counter	3

Output
0 1 2

Counting

Repetition is useful for counting up or down to a given number. This algorithm counts up to an inputted number from 0.

```
INPUT Target
Counter ← 0
WHILE Counter <= Target
      PRINT Counter
      Counter ← Counter + 1
ENDWHILE
```

Target	5
Counter	3

Output
0 1 2 3

Counting

Repetition is useful for counting up or down to a given number. This algorithm counts up to an inputted number from 0.

```
INPUT Target
Counter ← 0
WHILE Counter <= Target
      PRINT Counter
      Counter ← Counter + 1
ENDWHILE
```

Target	5
Counter	4

Output
0
1
2
3

Counting

Repetition is useful for counting up or down to a given number. This algorithm counts up to an inputted number from 0.

```
INPUT Target
Counter ← 0
WHILE Counter <= Target
      PRINT Counter
      Counter ← Counter + 1
ENDWHILE
```

Target	5
Counter	4

Output
0 1 2 3 4

Counting

Repetition is useful for counting up or down to a given number. This algorithm counts up to an inputted number from 0.

```
INPUT Target
Counter ← 0
WHILE Counter <= Target
      PRINT Counter
      Counter ← Counter + 1
ENDWHILE
```

Target	5
Counter	5

Output
0
1
2
3
4

Counting

Repetition is useful for counting up or down to a given number. This algorithm counts up to an inputted number from 0.

```
INPUT Target
Counter ← 0
WHILE Counter <= Target
      PRINT Counter
      Counter ← Counter + 1
ENDWHILE
```

Target	5
Counter	5

Output
0
1
2
3
4
5

Counting

Repetition is useful for counting up or down to a given number. This algorithm counts up to an inputted number from 0.

```
INPUT Target
Counter ← 0
WHILE Counter <= Target
      PRINT Counter
      Counter ← Counter + 1
ENDWHILE
```

Target	5
Counter	6

Output
0
1
2
3
4
5

Totalling

Repetition is also useful for calculating the sum of a set of numbers.

```
Total ← 0
FOR i ← 1 TO LENGTH(list)
    Total ← Total + list[i]
ENDFOR
PRINT Total
```

Total	0
Index	

Output

List	8	2	6	3

Totalling

Repetition is also useful for calculating the sum of a set of numbers.

```
Total ← 0
FOR i ← 1 TO LENGTH(list)
    Total ← Total + list[i]
ENDFOR
PRINT Total
```

Total	0
Index	1

Output

List	8	2	6	3

Totalling

Repetition is also useful for calculating the sum of a set of numbers.

```
Total ← 0
FOR i ← 1 TO LENGTH(list)
    Total ← Total + list[i]
ENDFOR
PRINT Total
```

Total	8
Index	1

Output

List	**8**	2	6	3

Totalling

Repetition is also useful for calculating the sum of a set of numbers.

```
Total ← 0
FOR i ← 1 TO LENGTH(list)
    Total ← Total + list[i]
ENDFOR
PRINT Total
```

List	8	2	6	3

Total	8
Index	2

Output

Totalling

Repetition is also useful for calculating the sum of a set of numbers.

```
Total ← 0
FOR i ← 1 TO LENGTH(list)
    Total ← Total + list[i]
ENDFOR
PRINT Total
```

Total	10
Index	2

Output

List	8	**2**	6	3

Totalling

Repetition is also useful for calculating the sum of a set of numbers.

```
Total ← 0
FOR i ← 1 TO LENGTH(list)
    Total ← Total + list[i]
ENDFOR
PRINT Total
```

Total	10
Index	3

Output

List	8	2	6	3

Totalling

Repetition is also useful for calculating the sum of a set of numbers.

```
Total ← 0
FOR i ← 1 TO LENGTH(list)
    Total ← Total + list[i]
ENDFOR
PRINT Total
```

Total	16
Index	3

Output

List	8	2	**6**	3

Totalling

Repetition is also useful for calculating the sum of a set of numbers.

```
Total ← 0
FOR i ← 1 TO LENGTH(list)
    Total ← Total + list[i]
ENDFOR
PRINT Total
```

Total	16
Index	4

Output

List	8	2	6	3

Totalling

Repetition is also useful for calculating the sum of a set of numbers.

```
Total ← 0
FOR i ← 1 TO LENGTH(list)
    Total ← Total + list[i]
ENDFOR
PRINT Total
```

Total	19
Index	4

Output

List	8	2	6	**3**

Totalling

Repetition is also useful for calculating the sum of a set of numbers.

```
Total ← 0
FOR i ← 1 TO LENGTH(list)
    Total ← Total + list[i]
ENDFOR
PRINT Total
```

Total	19
Index	4

Output

List	8	2	6	3

Totalling

Repetition is also useful for calculating the sum of a set of numbers.

```
Total ← 0
FOR i ← 1 TO LENGTH(list)
    Total ← Total + list[i]
ENDFOR
PRINT Total
```

List	8	2	6	3

Total	19
Index	4

Output
19