Programming Constructs

Sequence

A sequence is simply a series of lines of code, where all lines are executed, in order, once.

Selection

A selection structure allows us to choose which code is executed, and which is skipped over. The most commonly used Selection structure is If..then...else.

In Python -

```
if x<10:
    print("Less than ten")
else:
    print("Bigger than ten")
```

In Pseudocode -

```
IF x<10 THEN
    SEND "Less than ten" TO DISPLAY
ELSE
    SEND "Bigger than ten" TO DISPLAY
END IF
```

Draw a flow diagram representation of the above If...then...else statement:

Selections are often 'nested' to allow for more options - i.e. an If...then...else is put inside another if...then...else. Some programming languages also allow the use of an 'Else if' statement that introduces a new condition. E.g. in Python:

```
if x<10:
    print("Less than ten")
elif x==10:
    print("Exactly ten")
```

```
else:
        print("Bigger than ten")
```

An 'else if' statement is never strictly necessary, as you can write the same code without it, e.g.

```
if x<10:
        print("Less than ten")
        if x==10:
                print("Exactly ten")
        else:
                print("Bigger than ten")
```

However, sometimes elif makes code more readable, as it doesn't require so much indentation and nesting.

Iteration

An iteration is the correct term for a loop - i.e. when code repeats.

There are two categories of iteration, **conditional controlled** and **count controlled**.

Conditional controlled

These loops repeat until a particular condition is met. The most commonly used loop of this kind is the while loop. For example, in Python:

```
X = 2
While X < 1000:
        X = X**2
        print (X)
```

Or in Pseudocode:

```
SET X TO 2
WHILE X < 1000 DO
        SET X TO X^2
        SEND X TO DISPLAY
END WHILE
```

We can also use a Repeat...Until loop. In a Repeat...Until loop the condition comes at the end of the loop, and the loop keeps going until this condition is met. Python does not use Until loops, but in Pseudocode we could re-write the above code as:

```
SET X TO 2
```

```
REPEAT
        SET X TO X^2
        SEND X TO DISPLAY
UNTIL X >= 1000 DO
```

Note how the condition changes in from a While loop to an Until loop.

We can always re-write an Until loop as a While loop, but sometimes an algorithm will read in a more intuitive way if written as an Until loop.

Count controlled Iteration

In a count controlled iteration we can specify exactly how many times the loop is performed. This involves the For loop.

In Python:

```
for i in range (1,11):
        print(i)
```

In Pseudocode:

```
FOR i FROM 1 TO 10 DO
        SEND i TO DISPLAY
END FOR
```

Both of these pieces of code do the same thing, but note the difference in the numbers used to control the loop.

We can also control the ‘step size’ in a loop, e.g. in Python:

```
for i in range (1,21,2):
        print(i)
```

This will count up in 2s.

In Pseudocode:

```
FOR i FROM 1 TO 20 STEP 2 DO
        SEND i TO DISPLAY
END FOR
```

This will do the same.

Finally we can use the ‘For each’ construct to traverse a string or an array. In Python:

```
MyArray = [1,3,5,7,9]
for i in MyArray:
        print(i)
```

Or in Pseudocode:

```
SET MyArray TO [1,3,5,7,9]
```

```
FOR EACH i FROM MyArray DO
        SEND i TO DISPLAY
END FOR
```

Arrays

An array is a data structure that allows us to store multiple pieces of data under a single variable name. In most programming languages all of the data stored in an array has to be of the same data type. In Python this is not the case (technically arrays in Python are lists), allowing us to put different types of data together.

Arrays can be one or two dimensional (or more in some programming languages). This just alters how many indexes we need to use to refer to a piece of data. An **index** is just the term we use for the number that refers to a piece of data, the piece of data itself is called the 'element'.E.g.

MyArray = ["Up","Down","Left","Right","Inside","Outside"]

Index	0	1	2	3	4	5
Element	Up	Down	Left	Right	Inside	Outside

Traversing an array means visiting each element in turn. This is often done to either search the array, print the array, sum the array, or some other function involving all elements of the array.

E.g. in Python:

```
for i in range (0,6):
        print(MyArray(i))
```

In Pseudocode:

```
SET MyArray TO ["Up","Down","Left","Right","Inside","Outside"]
FOR index FROM 0 TO LENGTH(MyArray) -1 DO
        SEND MyArray[index] TO DISPLAY
END FOR
```

We can also use the 'for each' construct discussed above to iterate through an array.

Question - Write code which would search My Array for the word 'Left' and will output the index of its location.