

Data Types & Structures

INTEGER (whole numbers)	1, 2, 45, 231, -2, -213
REAL (decimal numbers)	1.23, 938.232312, -23.233
STRING (words or sentences)	"Hello World", "123 bipbop"
CHARACTER (one character or alphabet)	'A', 'b', 'Y', 'o', '1', '8', 'd'
BOOLEAN (true or false)	TRUE, FALSE, TRUE, TRUE
ARRAY (many words or numbers)	["Hi", "me not smart", 174, 23, "12 bipbop", -1234]

Rules:

INT and **REAL** are typed as numbers only
STR must be in double quotation marks i.e. "Hello"

CHAR must be in single quotation marks i.e. 'H'

BOOL must be in either TRUE or FALSE

Array can be assigned to variables and must be inside braces. i.e. names = ["Andy", "Cindy", "Bobby"]

Operators (Arithmetic)

Operators	Examples	It Will Give
+	5 + 5	10
-	8 - 3	5
*	6*2	12
/	8/4	2
^	2^3	8
DIV (dividend)	12 DIV 5	2
MOD (modulo)	10 MOD 5	0

Note: MOD will give you the remainder of the number

Operators (Assignment)

Operators	Examples
> (greater than sign)	100 > 5
< (lesser than sign)	60 < 150
<> (not equal sign)	7 <> 25
= (equal sign)	12 = 12

The >= operator means greater than or equal to

The <= operator means lesser than or equal to

Operators (boolean)

Operators	Examples
AND	TRUE AND TRUE = TRUE
OR	TRUE OR FALSE = TRUE
NOT	NOT FALSE = TRUE

Boolean Operators are also called Logical Operators. AND means conjunction, OR means disjunction, NOT means negation

The FOR Loop

FOR LOOP

Use a FOR loop when you know how many times it'll run

The syntax:

```
FOR variable = something to something
do something
```

NEXT

The example:

```
FOR number = 1 to 3:
    PRINT number
```

NEXT

Running this on raptor/ other engines will output:

1
2
3

REPEAT Loop

THE REPEAT LOOP

Use a REPEAT loop when you know how many times it'll run

The syntax:

```
REPEAT
do something
UNTIL something
```

The example:

```
number = 1
REPEAT
    PRINT number
    number = number + 1
UNTIL number = 3
```

Running this on raptor/ other engines will output:

1
2
3

WHILE Loop

THE WHILE LOOP

Use WHILE loop when you don't know how many times it'll run

The syntax:

```
WHILE something DO
do something
ENDWHILE
```

The example:

```
number = 1
WHILE number < 3 DO
    PRINT number
    number = number + 1
ENDWHILE
```

If you put this in raptor/other engines they will output:

1
2
3

Writing An Algorithm Example

You are tasked to find out what is the 16th number in the Fibonacci sequence, how do you do it?

```
fib_0 = 0
fib_1 = 1
find_num = 16
FOR i = 0 to find_num
  — fib_2 = fib_0 + fib_1
  — fib_0 = fib_1
  — fib_1 = fib_2
PRINT "The 16th number is: ", fib_2
```

Answer of Output = 987

Fibonacci Sequence is a sequence where each number is the sum of the two preceding ones. For example: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Conditional Statements (IF)

Syntax:

```
IF condition
  THEN do something
  ELSE do something else
ENDIF
```

Example:

```
khaisa_r_height = 100
IF khaisa_r_height < 120
  THEN PRINT "Please enter
the kids room"
  ELSE PRINT "Please enter
the teenagers room"
ENDIF
```

Description:

Conditional Statements are IF statements. The IF statement check if a condition is TRUE or FALSE.

In the English language, we say:

If you come to the party, I'll buy one pizza.

In the computer language. We say it like this:

IF answer = "YES" THEN pizza = 1 ENDIF
As simple as that!

Extended IF statement

```
IF grade > 90 AND behavior > 80
  THEN PRINT "Excellent"
  ELSE IF grade > 80 AND behavior > 70
    THEN PRINT "Well Done"
  ELSE IF grade > 70 AND behavior > 60
    THEN PRINT "Good Job"
  ELSE PRINT "Improvement is needed!"
ENDIF
```

IF and ELIF (else if) are used for a choice between several different values. You can either use ELIF or CASE, it is up to you.

CASE OF

Syntax:

```
CASE ... OF
  something
  OTHERWISE something
ENDCASE
```

Example:

```
CASE grade OF
  'A' : PRINT "Excellent"
  'B' : PRINT "Well Done"
  'C' : PRINT "Good Job"
  'D' : PRINT "Do Better"
  'F' : PRINT "Find Another"
  OTHERWISE PRINT "Improve next time"
ENDCASE
```

CASE is another conditional statement that is use for several different values

Functions

The syntax:

```
FUNCTION do_something (Parameters)
  statements
  RETURN something
END PROCEDURE
```

Example:

```
FUNCTION count_pizza_slices (number_of_pizza)
  number_of_pizza_slices = 0
  number_of_pizza_slices = number_of_pizza * 8
  RETURN number_of_pizza_slices
END PROCEDURE
```

Running this on a engine/ide:

count_pizza_slices(5)

Output:

40