LANGUAGE REFERENCE

VECTORS

FINITE VECTORS

In order to create a vector, with certain numbers, the user types as following.

```
V = |2 3 4 5 7 9 0|
```

If the user wants to assign a vector a specific sequence of numbers (implicit step is 1):

```
A = |1..10|
```

Which is the same as:

```
A = |1 2 3 4 5 6 7 8 9 10|
```

And is also the same as:

```
A = |1..10:1|
```

However, the user might also want to get each second, or each third element of this sequence:

```
B = |1..10:2|
```

This expression basically generates a list, by taking each second term in the given sequence, i.e. the same as following expression:

```
B = |1 3 5 7 9|
```

However, the user might also set a rule that a vector should follow:

```
C = |1..10|.even
```

Which is the same as:

```
C = |2 4 6 8|
```

And if you want to add even more rules:

```
D = |1..20|.even.prime
```

Which will output following:

```
D = |2|
```

The sum of the elements in a finite vector is calculated by adding a plus sign in front of a vector identifier:

```
E = +|1..10|
```

This will result:

E = 55

INFINITE VECTORS

It is also possible to generate infinite vectors, i.e., a list of infinite amount of elements. A simple list of numbers $n \to \infty, n \in Z^+$, would look like:

Z = |n: 0 < n < {infinite}| or Z = |0..{infinite}|

FUNCTIONS

Sometimes, it's good to now a type of a variable, which can be found using the function *type*.

X = 2

X.type // returns "number"

But if it is a string:

Y= "Hello World"

Y.type // returns "string"

We can also check if the number is following a certain rule:

Z = (5).isprime // returns "1"

A comparison can also be performed using equal signs:

Q = +|1..10| == +|1..20| // returns "0"

And if we want to execute a statement:

Q ? "it's true" : "it's not" // which in this case returns "it's not"

CONDITIONAL OPERATIONS

Conditional operations are good to use if you plan to create a program.

```
if 5 == 2 then
        "this  can't be true"
else
        "as it should be!"
end
```

FOR LOOPS

If you want to repeat an operation a given amount of times, for loops will be the best solution.

```
for i = 0..2
    x
end
```

or

```
for i=0..2, x, end          // the same as above
```

You can also iterate through an already declared vector.

```
primes =  |0..10|.prime
for x = primes, x, end
// outputs |2 3 5 7|
```

If you for some reason want to output letters in order:

```
for x = |“a” “b” “c” “d”|
    x
end
// outputs a b c d
```