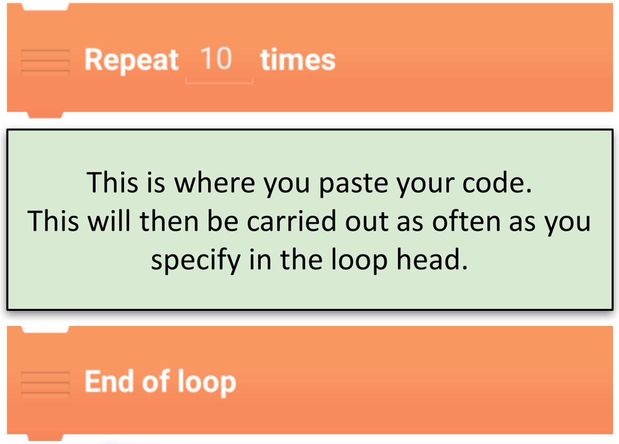
Loops

There are loops in every programming language. These help you do certain things over and over again.

In Pocket Code, for example, there is the "Repeat" brick, which helps you to execute your code more often.

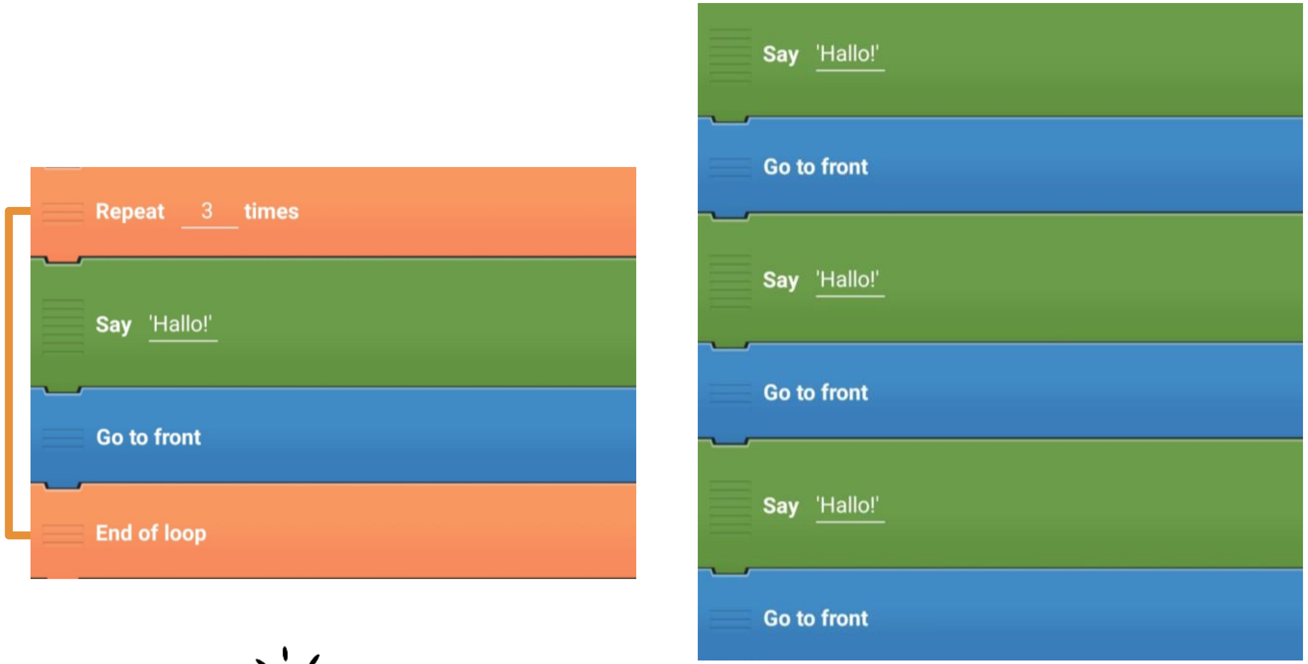
Repeat X times executes your code exactly X times within the loop (until the end of the loop), because with larger numbers it becomes very long-winded to copy the code over and over again (imagine you want to execute something 100 times) .

This is the beginning of the loop, also called the loop head. There you specify how often this loop should be executed.



This is where you paste your code.
This will then be carried out as often as you specify in the loop head.

That's the end of the loop. This is automatically created for you in Pocket Code.



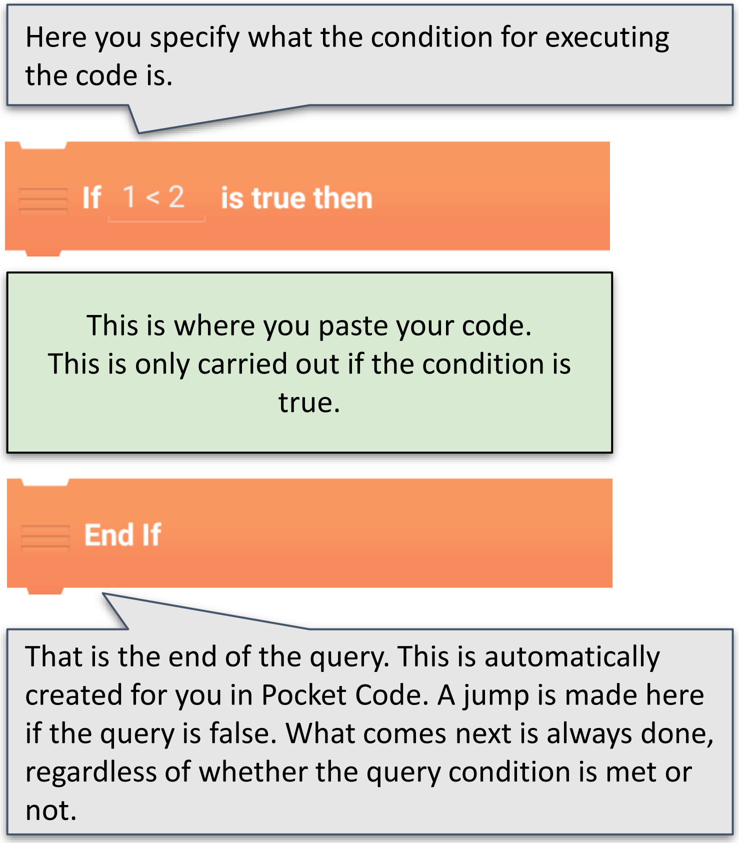
These two scripts do exactly the same thing.
Once with a loop and once without.

Query (If)

Queries help to only execute certain parts of your script if a condition is true.

If the queried is true, the code in between is executed - if it is not, this part is simply skipped.

If you need 50 points in a test to pass, you have to check how many points have been achieved and can than determine whether the test was passed or not.



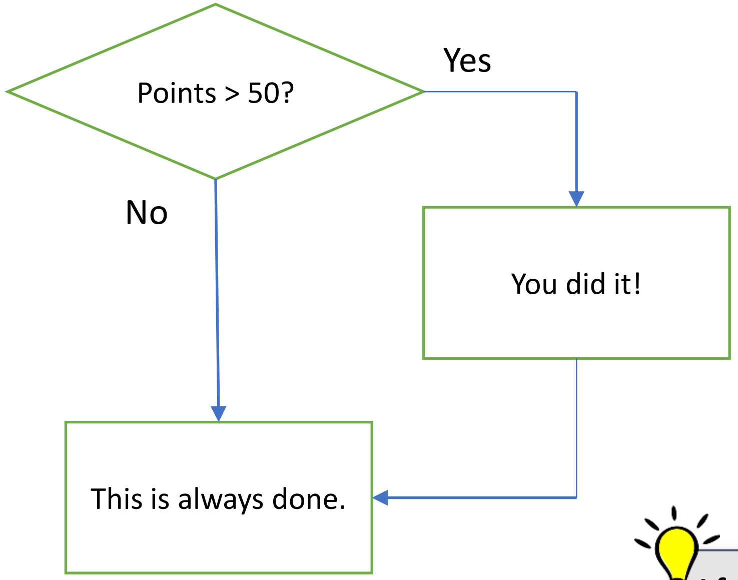
Here you specify what the condition for executing the code is.

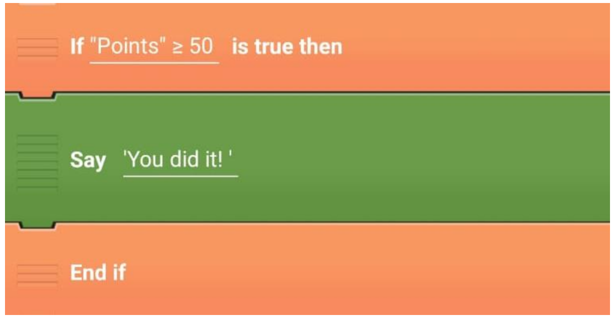
If 1 < 2 **is true then**

This is where you paste your code.
This is only carried out if the condition is true.

End If

That is the end of the query. This is automatically created for you in Pocket Code. A jump is made here if the query is false. What comes next is always done, regardless of whether the query condition is met or not.





If "Points" ≥ 50 **is true then**

Say 'You did it! '

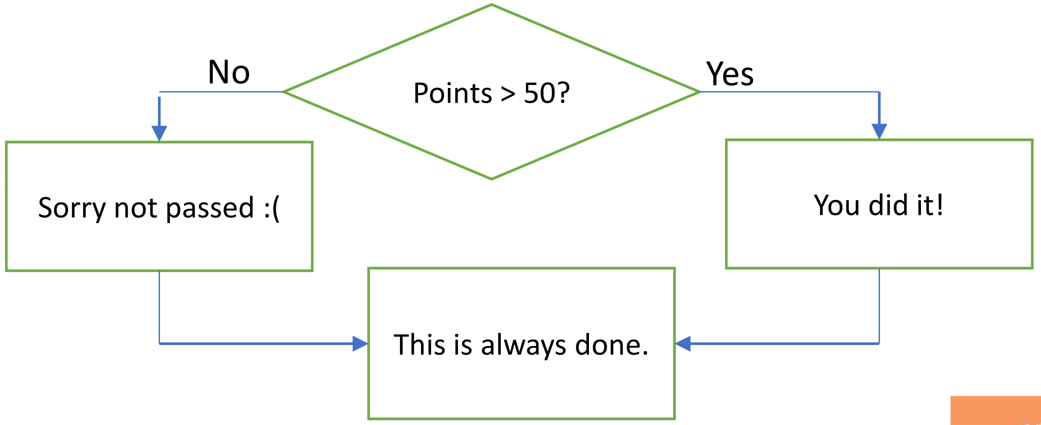
End if

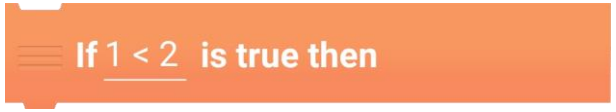
After the query, the code branches. If that in the diamond (the query) is correct, the right path is chosen - if it is not correct, it continues without detours.

Query (If - Else)

In contrast to the normal if-query, something is done here if the query is not fulfilled. Otherwise it works exactly the same.

There are two different paths to the if-else query - the query condition decides which path is taken.





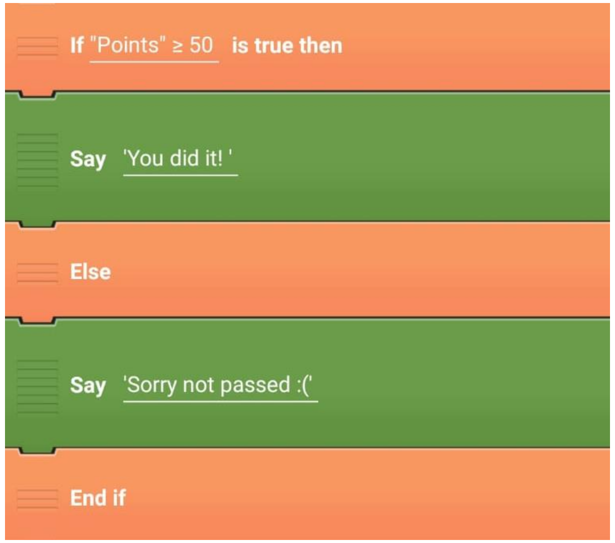
```
If 1 < 2 is true then
```

This part of the script is executed when the query condition is true.

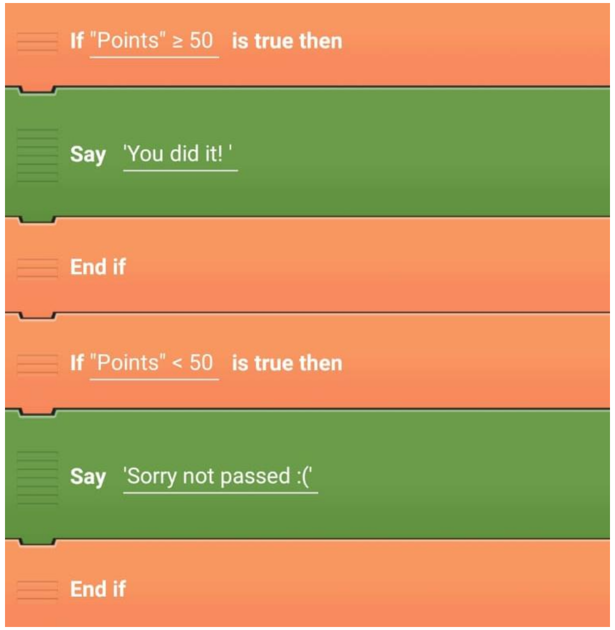
```
Else
```

This part of the script is executed when the query condition is **NOT** true.

```
End If
```



```
If "Points" ≥ 50  is true then
Say 'You did it! '
Else
Say 'Sorry not passed :('
End if
```



```
If "Points" ≥ 50  is true then
Say 'You did it! '
End if
If "Points" < 50  is true then
Say 'Sorry not passed :('
End if
```

These two scripts do exactly the same thing. Left with a combined if-else query and right with two if queries.

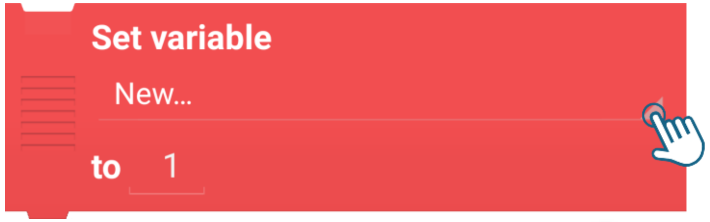
VARIABLES (create & use)

There are variables in every programming language. These help to save certain things and values and allow to change and retrieve them at any time.

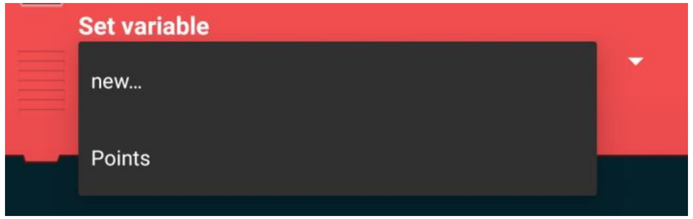
Unlike a fixed number, as the name suggests, a variable is variable. This means that it can be changed at any time.

How to create variables and work with your variables in Pocket Code:

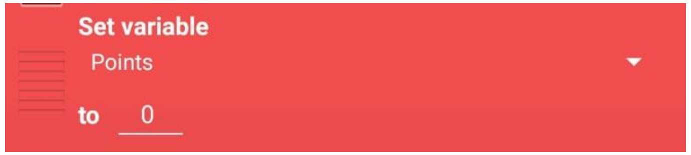
You can find everything about variables in the pink/red area **data**.


Set variable
New...
to 1

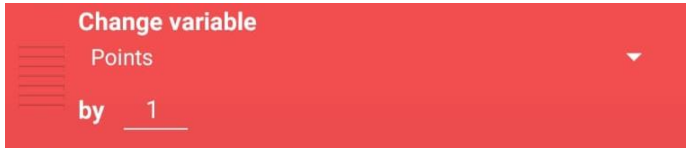
To create a new variable or to work with an existing one, press the little arrow in your selected pink/red block.



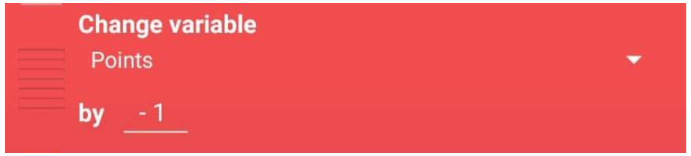
There you can choose whether you want to create a new variable (press new ...) or use an existing variable


Set variable
Points
to 0

This is how you give a variable a fixed value.


Change variable
Points
by 1

This is how you increase a variable (new value = value that was saved before + fixed value).


Change variable
Points
by - 1

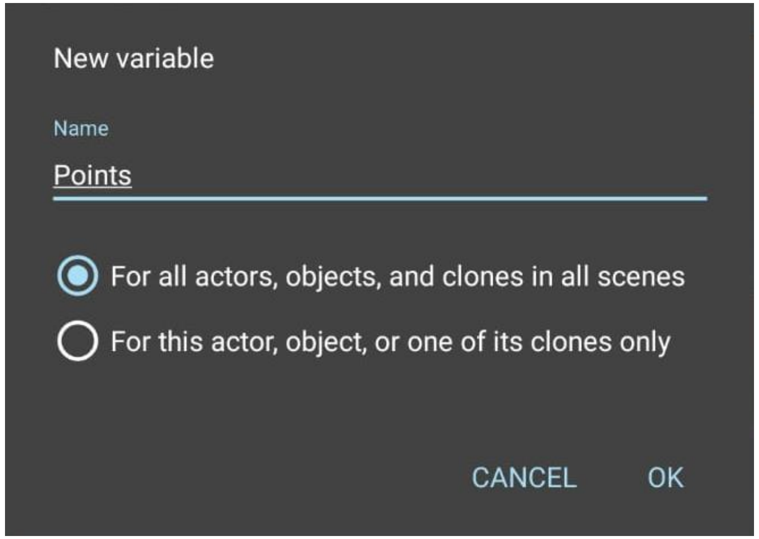
This is how you decrease a variable (new value = value that was saved before - fixed value).

VARIABLES (local & global)

You may have noticed that when you create a new variable, Pocket Code not only asks you for the name the variable should have, you also have to select whether the variable is “for all actors, objects and clones in all scenes” or “only for this actor, this object or one of its clones only”.

This is about visibility. So you have to decide whether the variable is only visible for this object (i.e. a charater or something similar) or for the whole project (your whole game) and can therefore also be edited.

In this context global means: visible everywhere in your project and local: only visible in your object / charater



New variable

Name

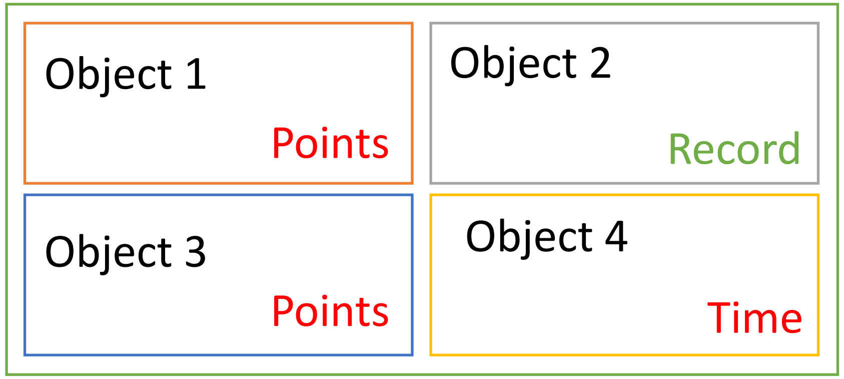
Points

● For all actors, objects, and clones in all scenes

○ For this actor, object, or one of its clones only

CANCEL OK

Your Project



Object 1 Points

Object 2 Record

Object 3 Points

Object 4 Time

red Variable: local
green Variable: global

The Record variable can be accessed from all objects

To the variable Time only in object 4

In the variable Points, other values can be stored in objects 1 and 3, as these are independent and invisible from one another and can therefore also have the same names