Basic looping techniques

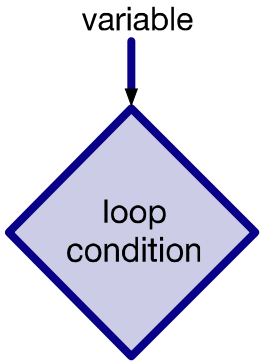
`while` loop: Only execute some code if a condition is met (1/6)

variable

The variable `x` takes the value 1

```
x <- 1
```

`while` loop: Only execute some code if a condition is met (2/6)

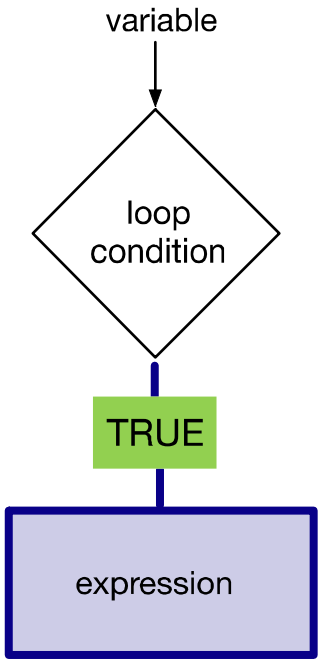



```
x <- 1

while (x <= 7){

}
```

1 The *loop condition* tests whether `x` is below or equal to 7

2 Use `{` and `}`

`while` loop: Only execute some code if a condition is met (3/6)



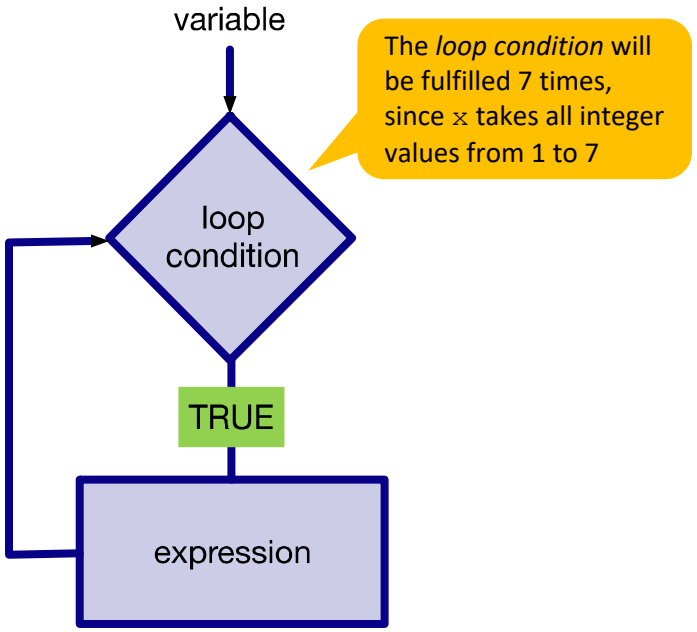
Code to be executed as long as the *loop condition* is fulfilled (1)

```
x <- 1

while (x <= 7){
   print(paste("x is set to ", x)
         x <- x + 1
}
```

Decide what happens while the *loop condition* is met, i.e. is `TRUE` (2)

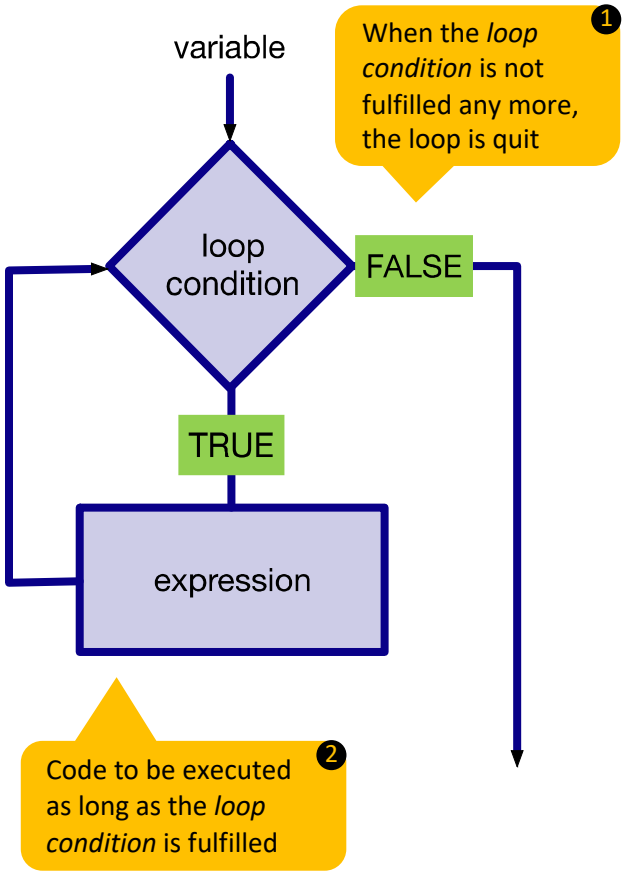
`x` increased by one for every loop the condition `x<=7` is fulfilled (3)

`while` loop: Only execute some code if a condition is met (4/6)




```
x <- 1

while (x <= 7){
   print(paste{"x is set to", x)
         x <- x + 1
}
```

`while` loop: Only execute some code if a condition is met (5/6)

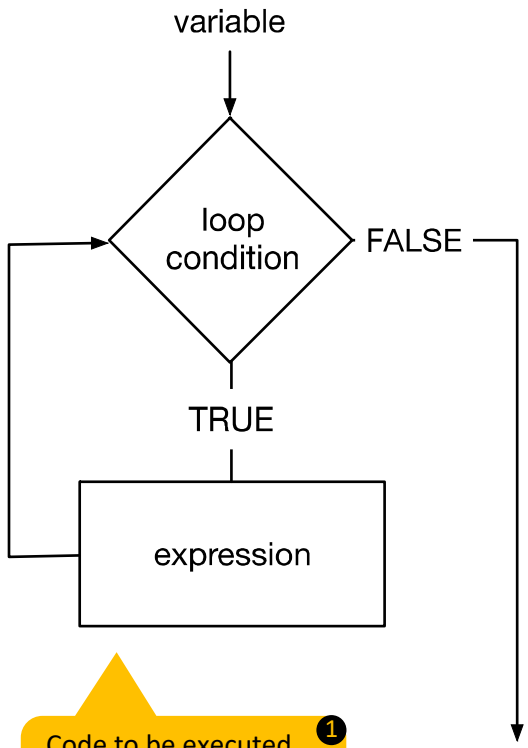



```
x <- 1

while (x <= 7){
   print(paste("x is set to", x))
         x <- x + 1
}
```

OUTPUT:

```
"x is set to 1"
"x is set to 2"
"x is set to 3"
"x is set to 4"
"x is set to 5"
"x is set to 6"
"x is set to 7"
```

`while` loop: Only execute some code if a condition is met (6/6)



General structure 2

```
while(condition){
   expr
}
```

Important

Make sure your loop condition is `FALSE` at some point.

Infinite loop

Loop condition is always `TRUE` and the while-loop is never quit, e.g.:

```
x <- 2
while (x == 2){
   print(x)
}
```

Infinite loops never stop! 3

Sidenote: use `paste()` to combine characters

Only used to combine **character strings**:

- E.g add a "%" to visual output

```
paste(variable, "%", sep="")
```

- E.g insert a variable into a sentence

```
paste("Customer spent", variable, "$")
```

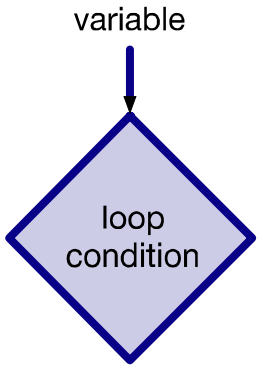
`for` loop: Repeat something as it belongs to a sequence (1/3)

variable

E.g. a sequence containing 1, 2, 3, 4, 5

```
cities <- c("New York", "Paris",
            "London", "Tokyo",
            "Cape Town")
```

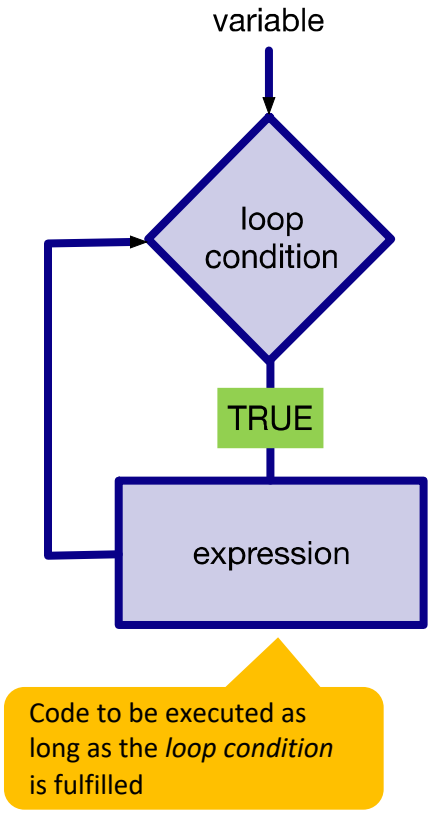
`for` loop: Repeat something as it belongs to a sequence (2/3)




```
cities <- c("New York", "Paris",
            "London", "Tokyo",
            "Cape Town")

for(i in 1:length(cities)){
    print(cities[i])
}
```

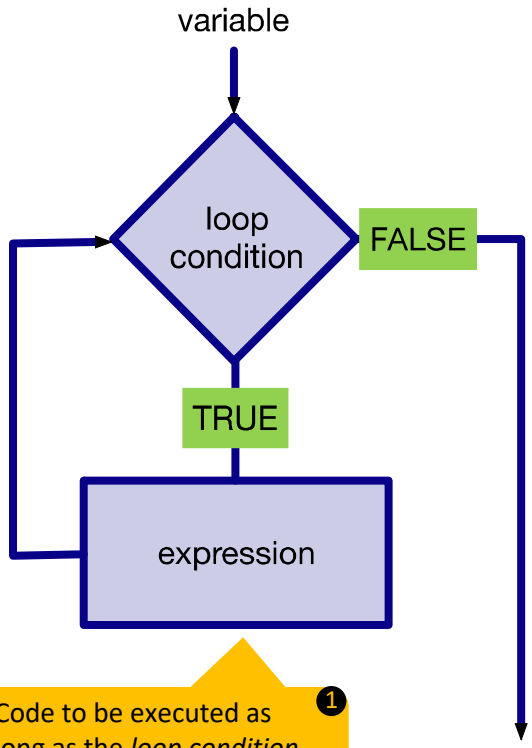
Loops over all elements in `cities`

`for` loop: Repeat something as it belongs to a sequence (3/3)




```
cities <- c("New York", "Paris",
            "London", "Tokyo",
            "Cape Town")

for(i in 1:length(cities)){
    print(cities[i])
}
```

`for` loop: Loop over elements



1 Code to be executed as long as the *loop condition* is fulfilled

```
cities <- c("New York", "Paris",
            "London", "Tokyo",
            "Cape Town")

for(i in 1:length(cities)){
    print(cities[i])
}
```

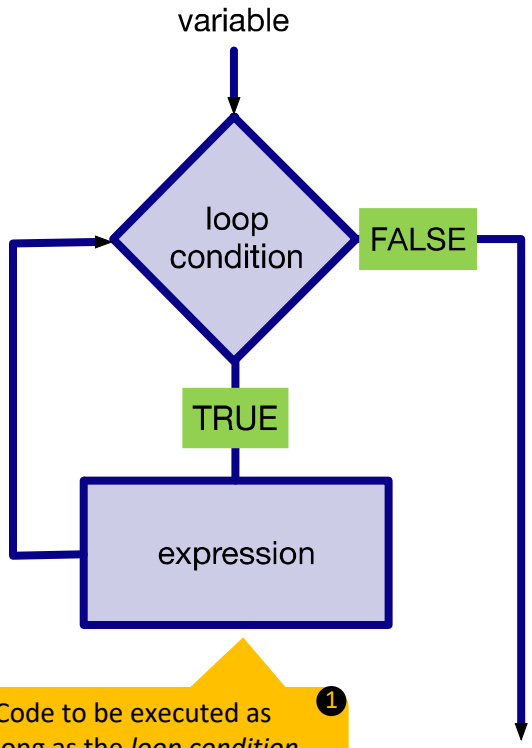
2 Loops over all elements in the list `cities`

OUTPUT:

```
"New York"
"Paris"
"London"
"Tokyo"
"Cape Town"
```

3 All elements are printed in the original order of the list

`for` loop: Loop over elements



1 Code to be executed as long as the *loop condition* is fulfilled

```
cities <- c("New York", "Paris",
            "London", "Tokyo",
            "Cape Town")

for(i in 1:length(cities)){
    print(cities[i])
}
```

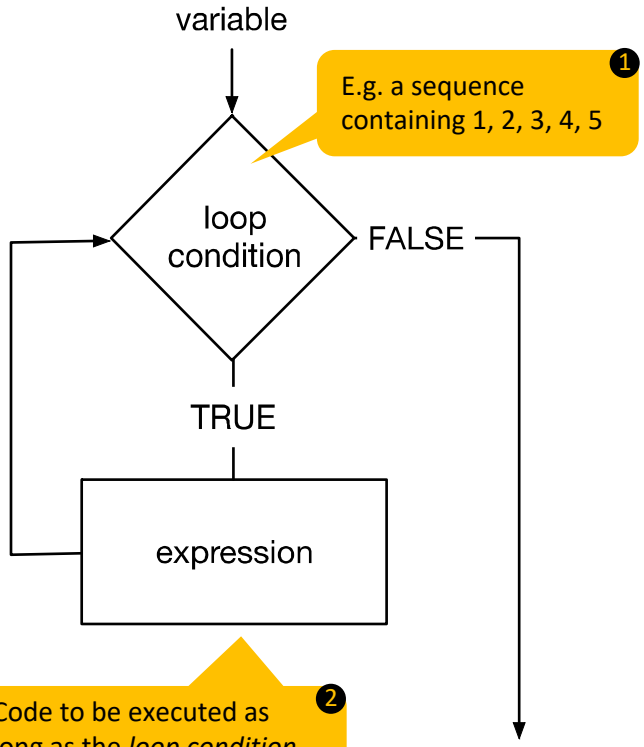
2 Loops over all elements in the list `cities`

OUTPUT:

```
"New York"
"Paris"
"London"
"Tokyo"
"Cape Town"
```

3 All elements are printed in the original order of the list

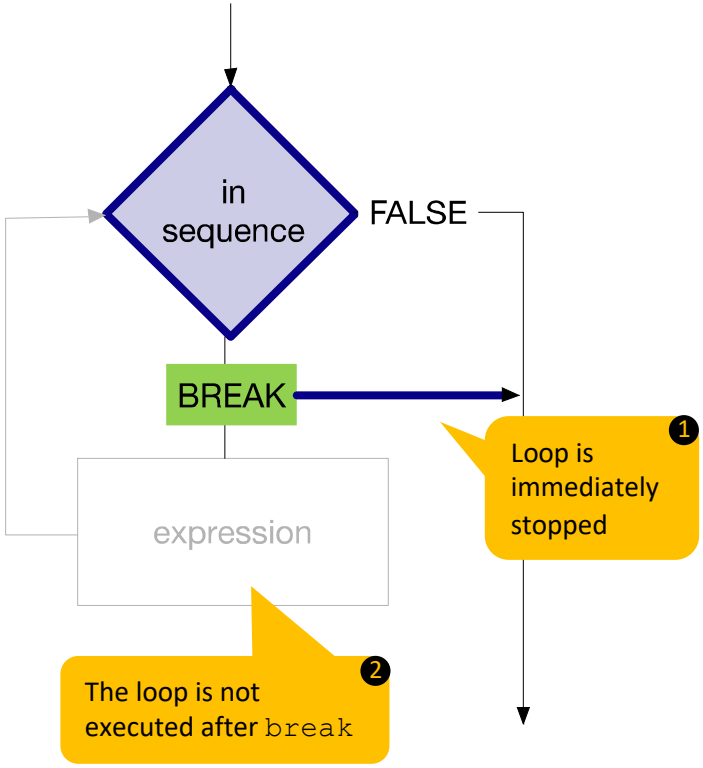
`for` loop: Repeat something as it belongs to a sequence



3 General structure

```
for(var in seq){
    expr
}
```

for loop: Take a break



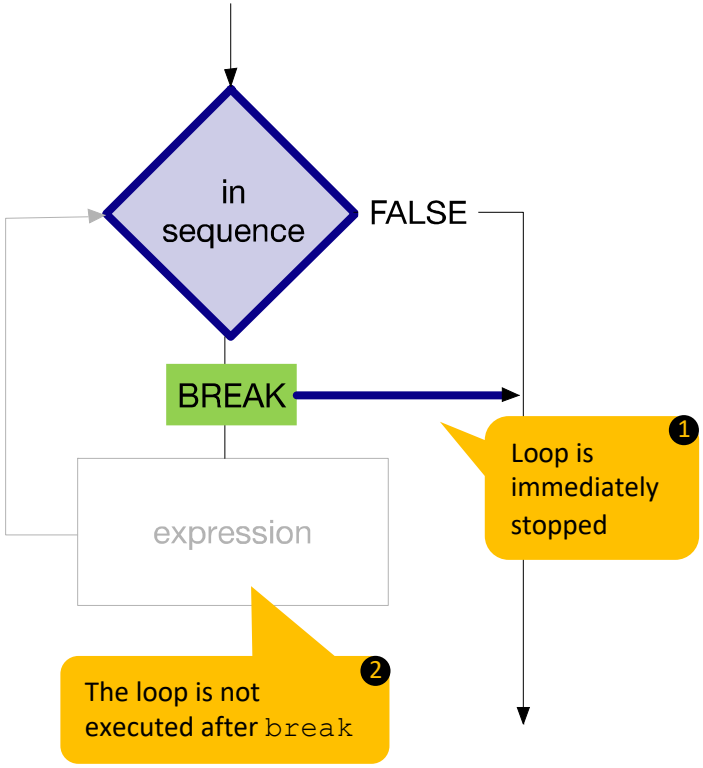

```
cities <- c("New York", "Paris",
            "London", "Tokyo",
            "Cape Town")

for(i in 1:length(cities)){
    if(cities[i] == "London"){
        break
    }
    print(cities[i])
}
```

OUTPUT:

```
"New York"
"Paris"
```

3 `London` is the third element in the list and the for loop is stops immediately, when `break` is executed

`for` loop: Take a `break`



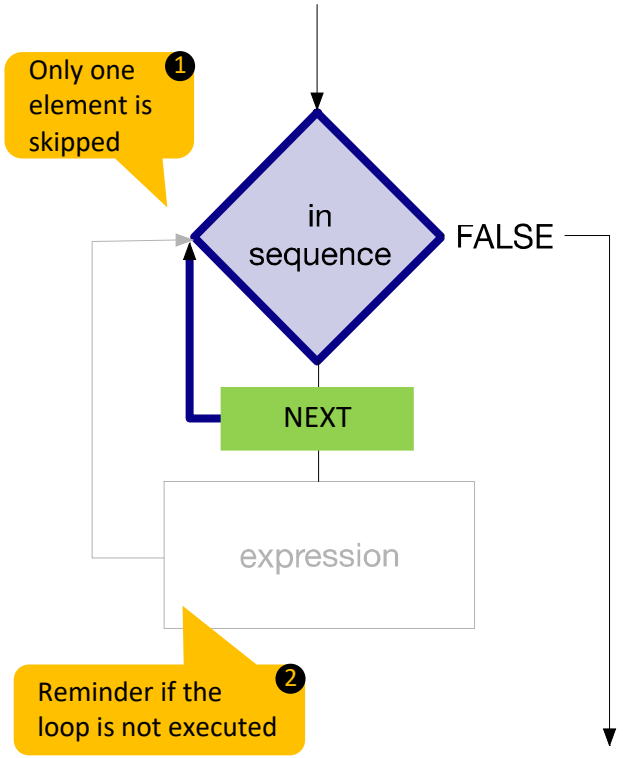

```
cities <- c("New York", "Paris",
            "London", "Tokyo",
            "Cape Town")

for(i in 1:length(cities)){
    if(cities[i] == "London"){
        break
    }
    print(cities[i])
}
```

OUTPUT:

```
"New York"
"Paris"
```

3 `London` is the third element in the list and the for loop is stops immediately, when `break` is executed

for loop: Skip with next



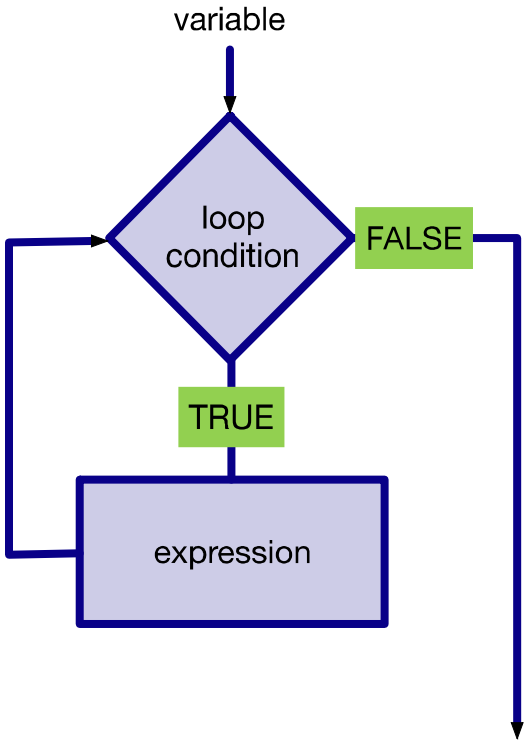

```
cities <- c("New York", "Paris",
            "London", "Tokyo",
            "Cape Town")

for(i in 1:length(cities)){
    if(cities[i] == "London"){
        next
    }
    print(cities[i])
}
```

3 For the list element `London`, the `if`-condition is `TRUE` and the `next`-label is reached

OUTPUT:

```
"New York"
"Paris"
"Tokyo"
"Cape Town"
```

`for` loop: Loop over vectors




```
cities <- c("New York", "Paris",
            "London", "Tokyo",
            "Cape Town")

for(city in cities){
    print(city)
}
```

To illustrate that we now loop over a vector, we no longer use the index name `i`, but change it to `city`

OUTPUT:

```
"New York"
"Paris"
"London"
"Tokyo"
"Cape Town"
```

`for` loop: Which one to choose?

Loop over vector

+ easy to code and read

- slower

- you can only use the elements in the vector
  (position of element in vector is not known)

Loop over numeric indices

+ more flexibility

+ position of element in vector is known

+ faster

- more difficult to code