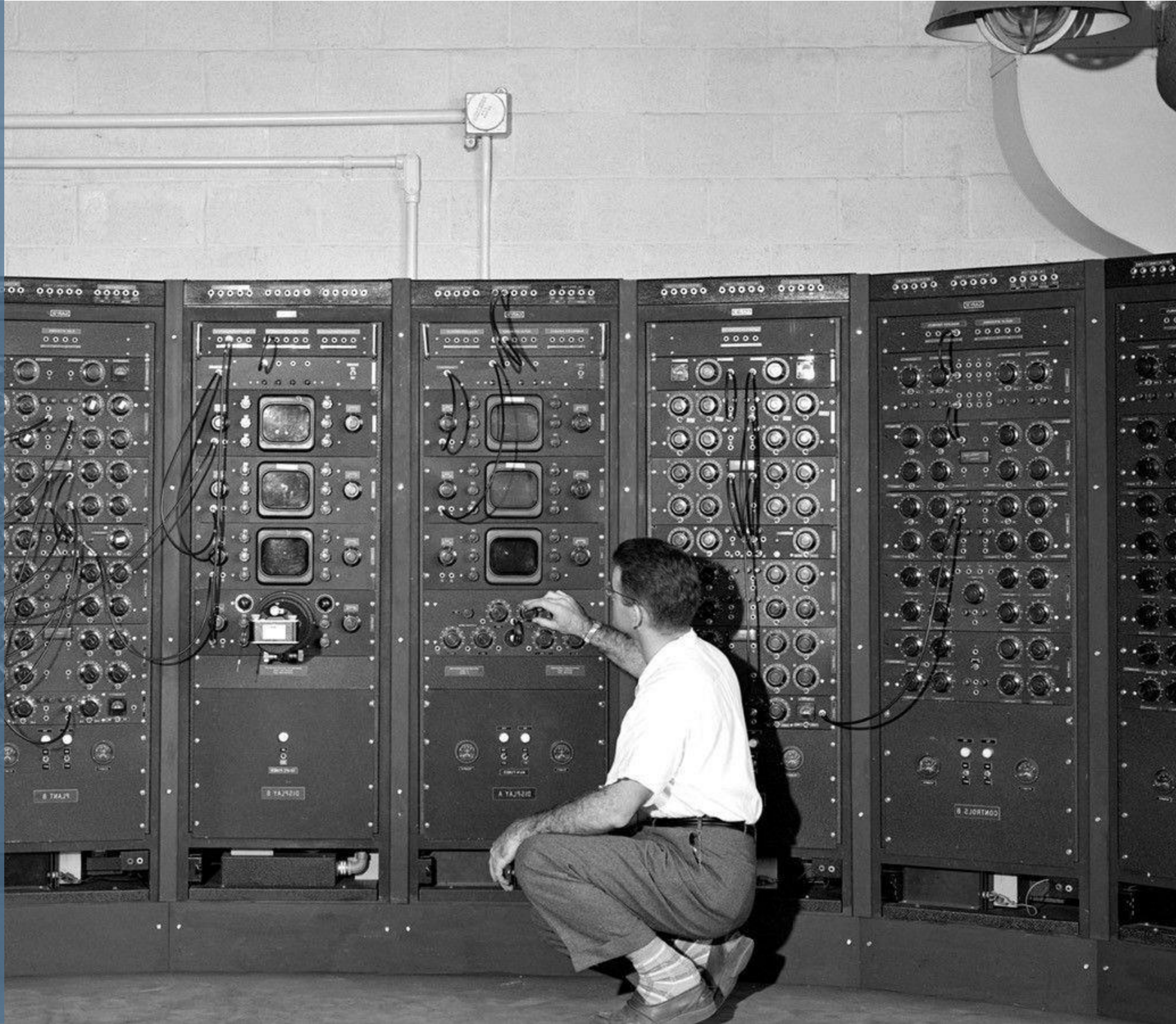


IF, LOOPS, FUNCTIONS

SESSION 2

FREDERIK MORTIER



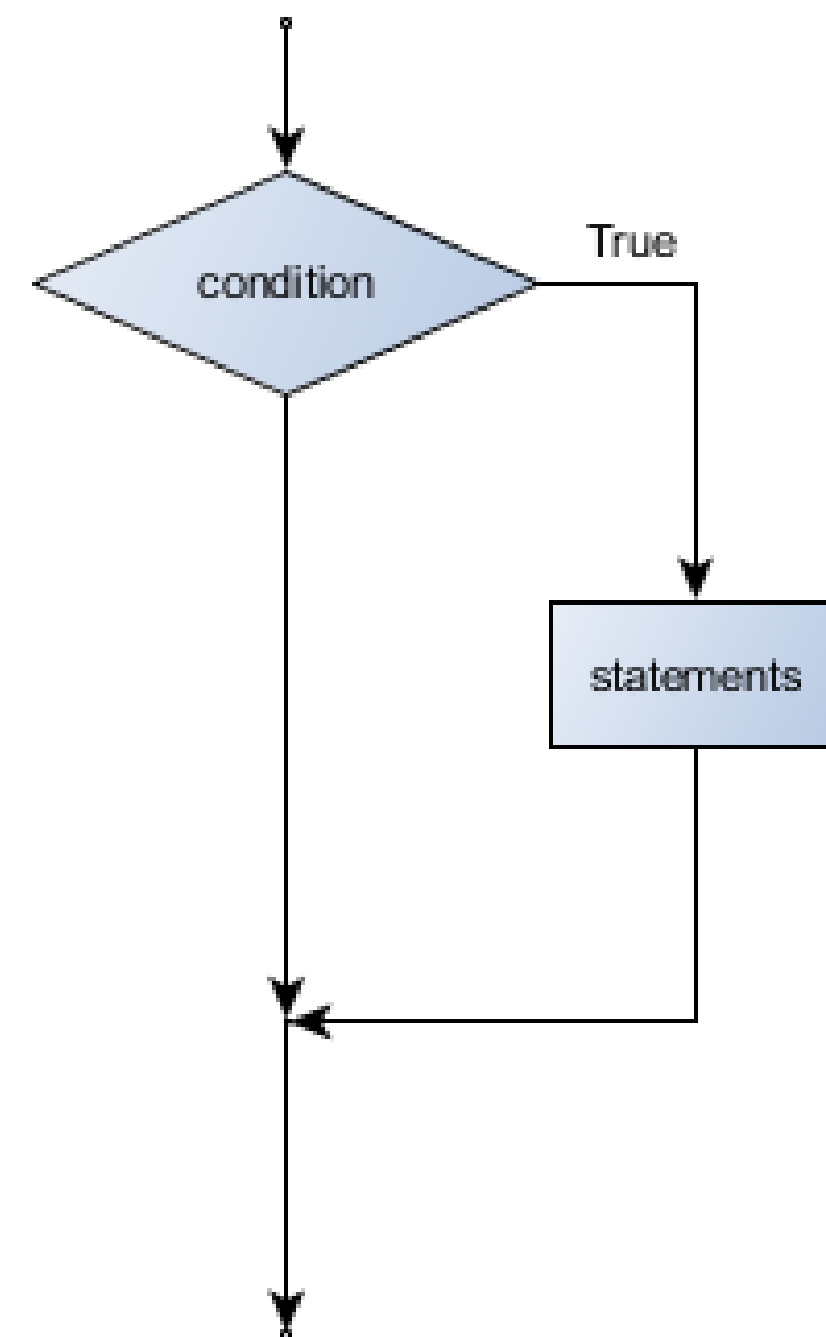
IF – THEN

CONDITIONAL STATEMENT

```
if (condition) {statement}
```

IF **condition** is TRUE

THEN perform **statements**



condition

- Evaluation: >, ==, %in%, ...
- Outcome:
TRUE or FALSE
1 or 0

2 > 1 → TRUE (or 1)

2 <= 1 → FALSE (or 0)

1 == 2 → FALSE

1 != 2 → TRUE

0 → FALSE

1 → TRUE

2 → TRUE

5-5 → FALSE

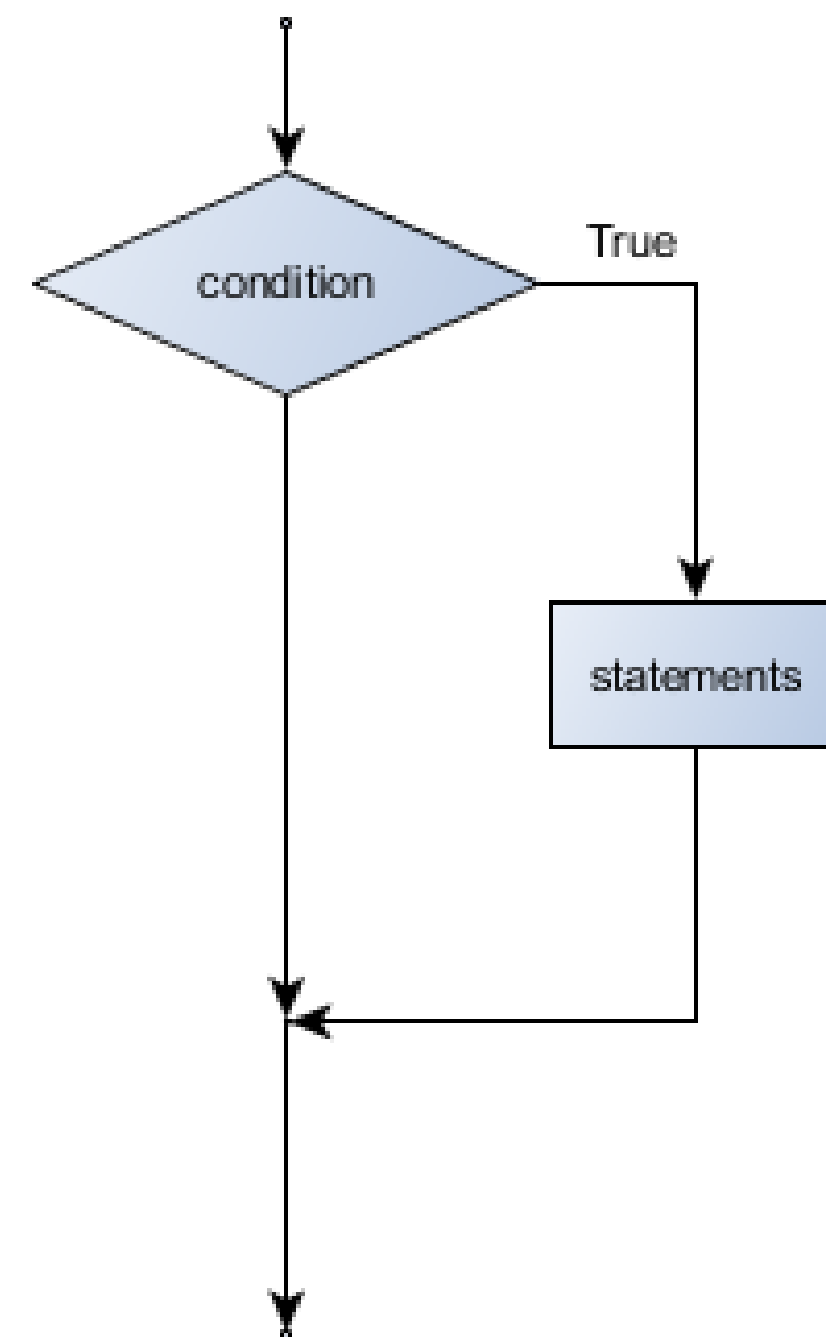
IF – THEN

CONDITIONAL STATEMENT

```
if (condition) {statement}
```

IF **condition** is TRUE

THEN perform **statements**



Statement

Print("Hello world")

Plot(x, y)

Data\$total <- Data\$d1 + Data\$d2

IF – THEN – ELSE

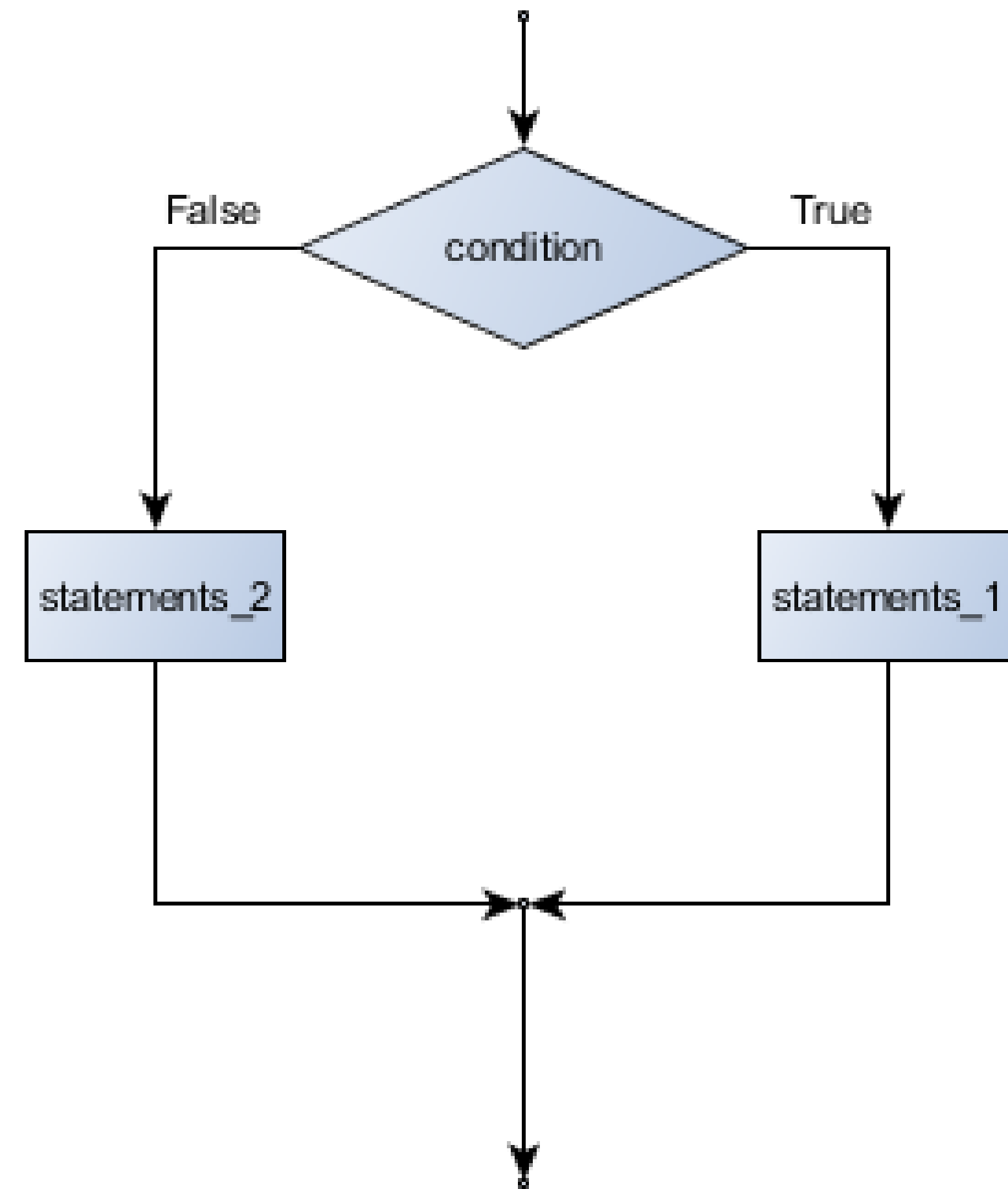
CONDITIONAL STATEMENT

```
if (condition) {statement 1}  
  else {statement 2}
```

IF **condition** is TRUE

THEN perform **statement 1**

ELSE perform **statement 2**



LOOPS

REPEATING ACTIONS

```
Print("Welcome")  
Print("Welcome")  
Print("Welcome")
```

- For a set amount of times:
→ For loop
- For as long as a condition is met:
→ While loop

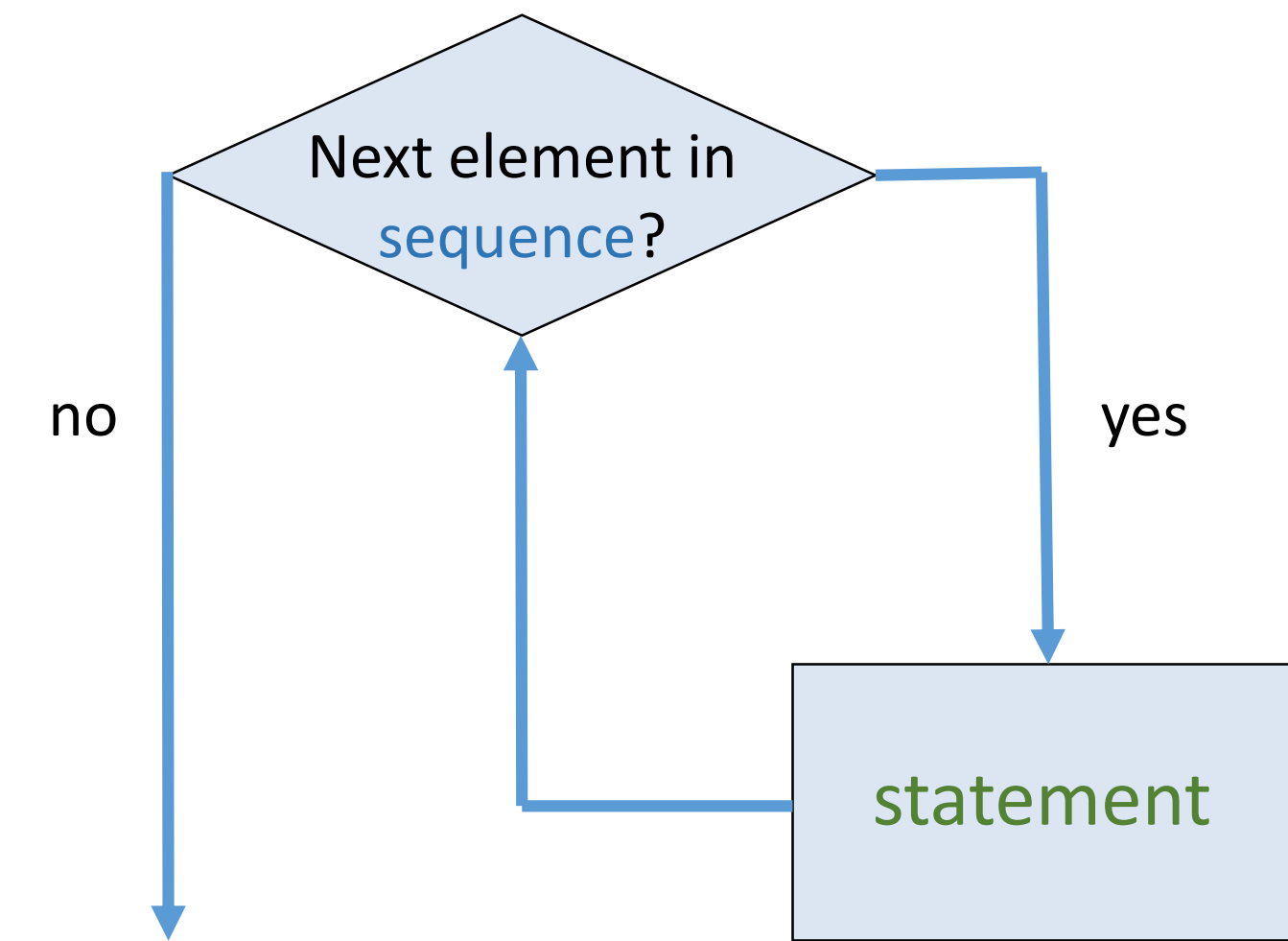
FOR LOOPS

SET AMOUNT OF LOOPS

```
for (sequence) {statement}
```

FOR each element in the sequence

Perform **statement**



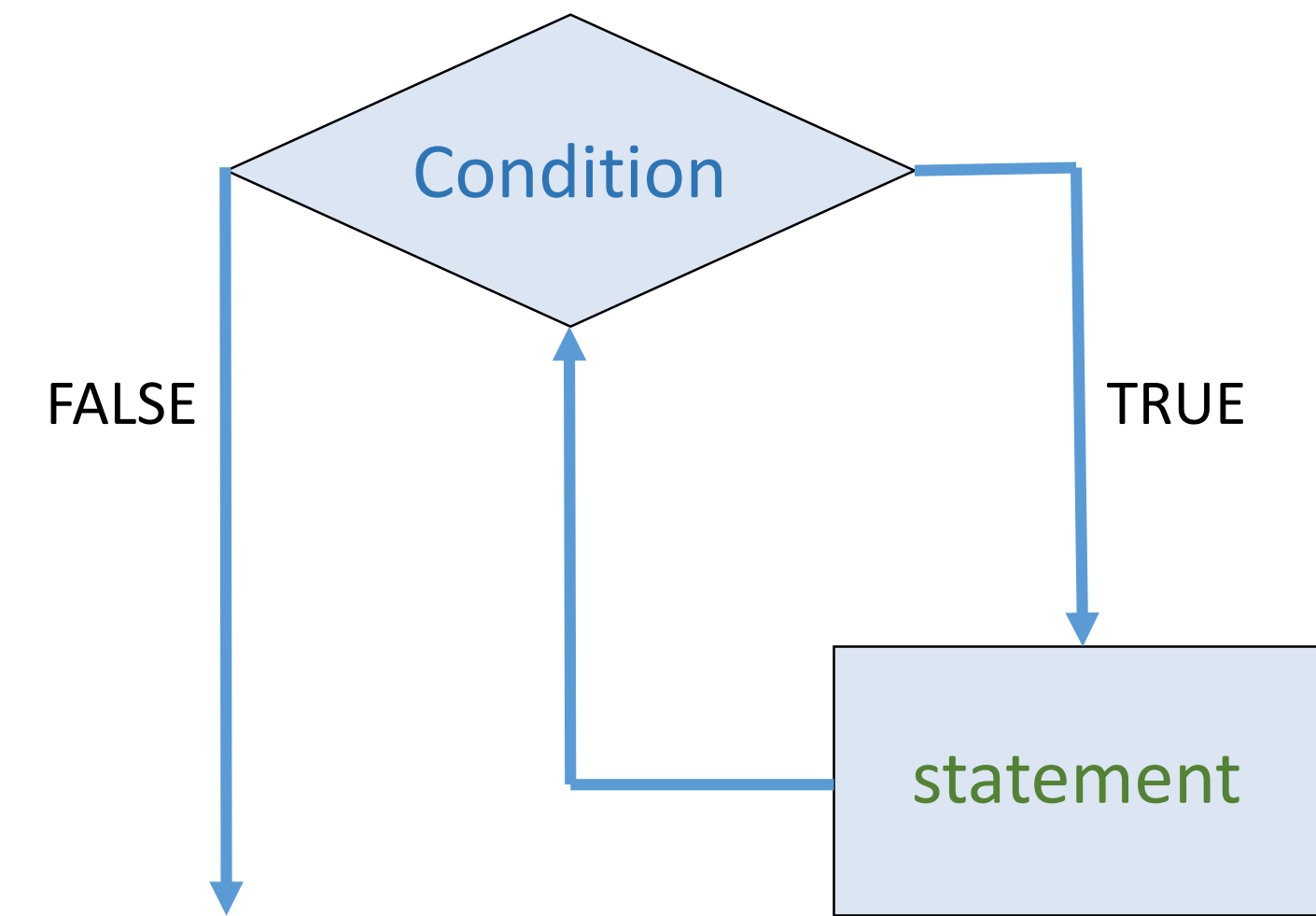
WHILE LOOPS

NUMBER OF LOOPS DEPENDENT OF A CONDITION

```
while (condition) {statement}
```

WHILE condition is TRUE

Perform **statement**



FUNCTION

PACKAGE A SET OF STATEMENTS IN ONE COMMAND

```
FuncName <- function(arguments) {statement}
```

Whenever the function is called

```
FuncName (arg1, arg2)
```

Perform **statement**

(return an output)

- Define a function once
- Call it as many times as you want (below its definition)

Arguments

Parameters that are **not fixed**

Passed when the function is called