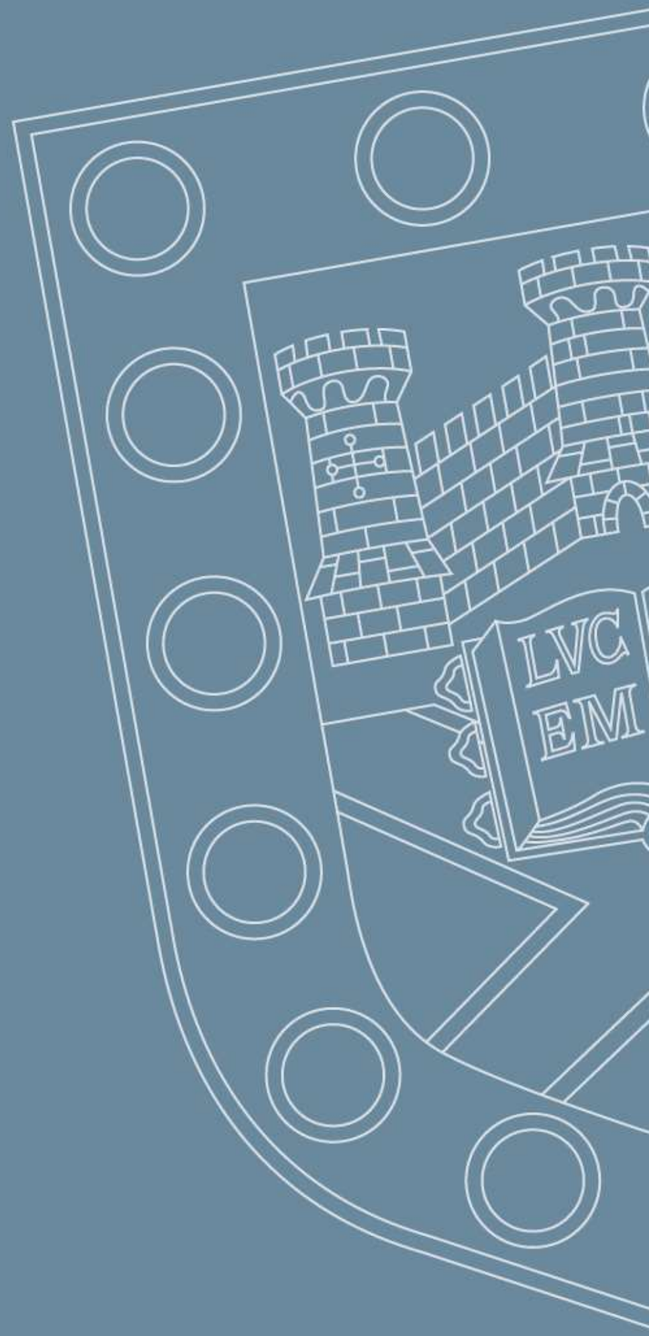
UNIVERSITY OF EXETER

# Python programming

## Algorithms



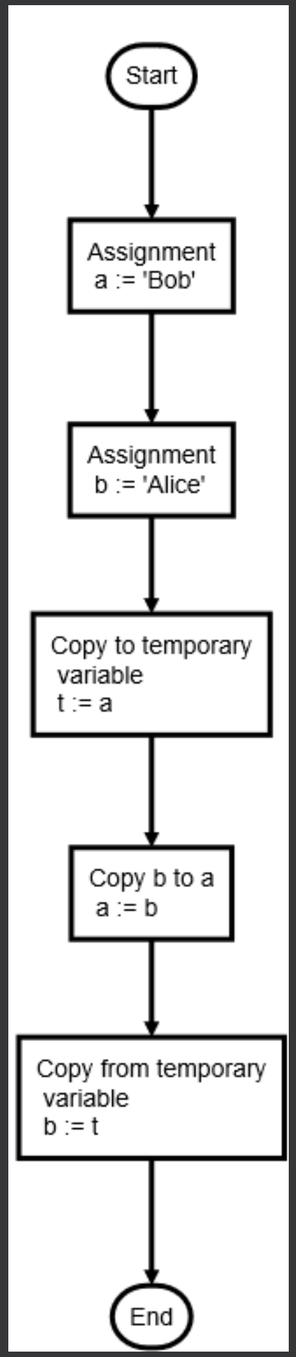

institute of CODING

# ALGORITHMS

## A set of instructions to solve a problem

Computer programs are texts that instruct the computer to manipulate data in a particular way.

For example swapping the contents of two variables.

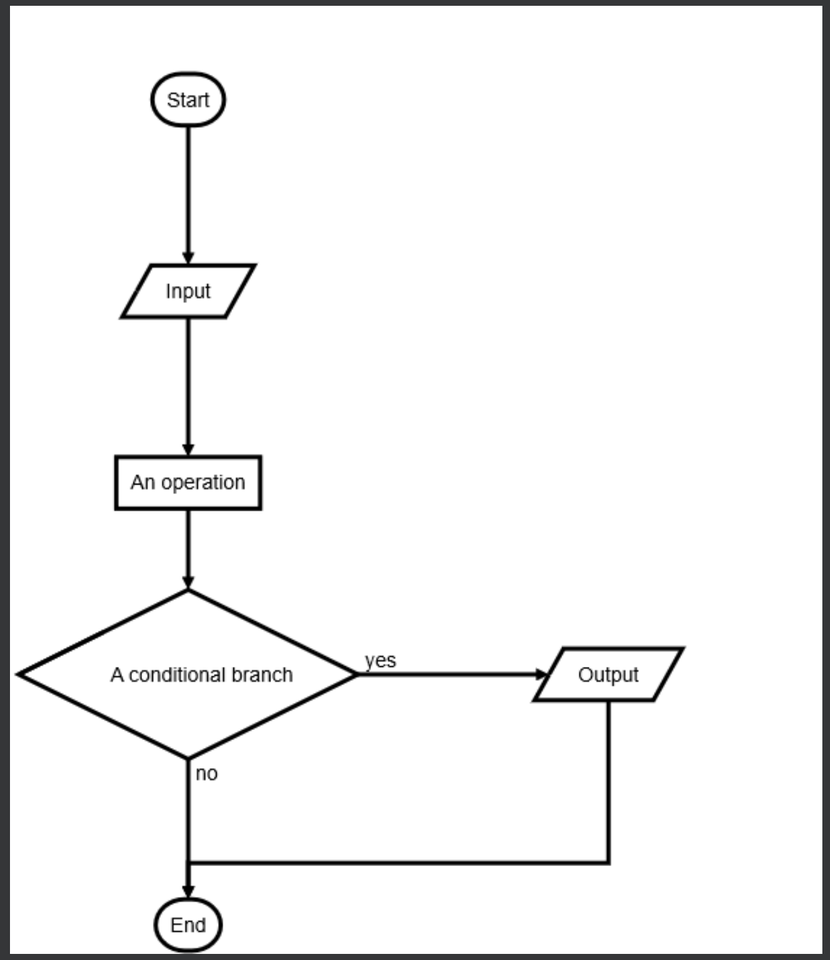


# ALGORITHMS

## Flowcharts

Flowcharts are not as widely used in computing as they once were.

Programming languages like Python make it easier to express algorithms in text form.



# ALGORITHMS

## Assembler language

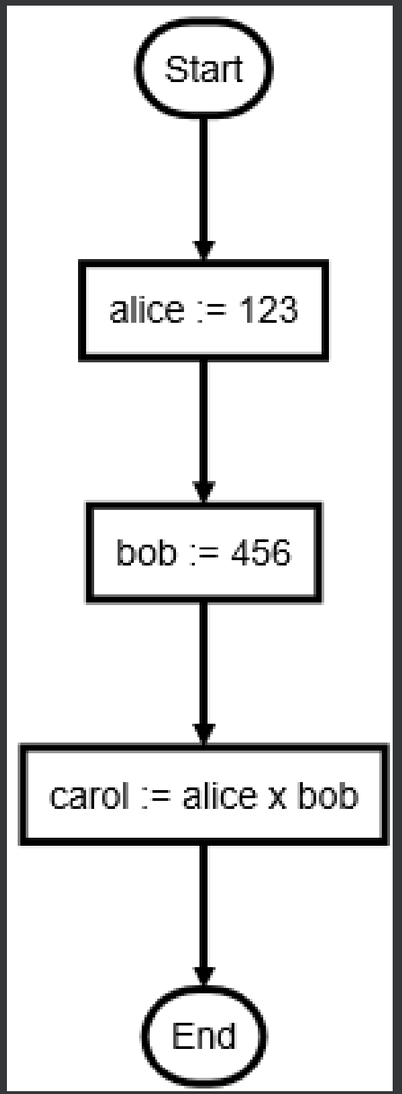
```
main:
        pushq   %rbp
        movq    %rsp, %rbp
        movl    alice(%rip), %edx
        movl    bob(%rip), %eax
        imull   %edx, %eax
        movl    %eax, carol(%rip)
        movl    $0, %eax
        leave
        ret
alice:
        .long   123
bob:
        .long   456
```

Computer programs when expressed in computer instructions, assembler code, look like this.

# ALGORITHMS

## Assembler language

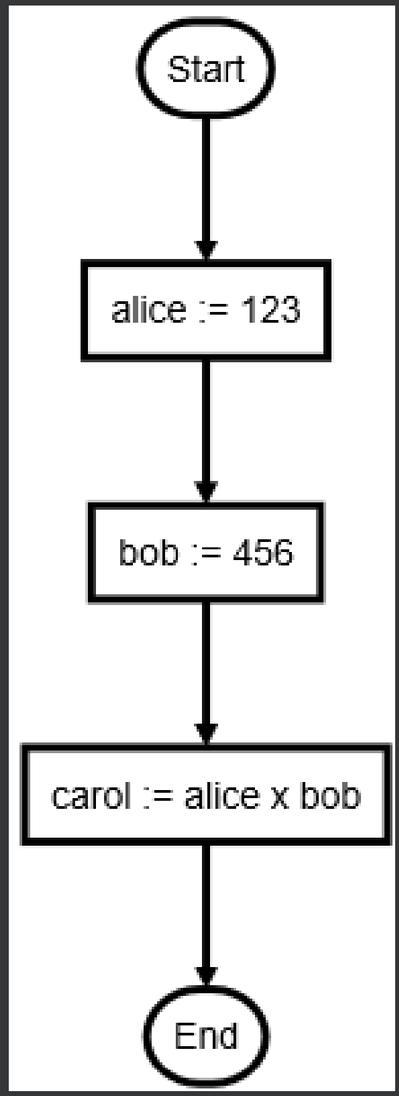
```
main:
        pushq   %rbp
        movq    %rsp, %rbp
        movl    alice(%rip), %edx
        movl    bob(%rip), %eax
        imull   %edx, %eax
        movl    %eax, carol(%rip)
        movl    $0, %eax
        leave
        ret
alice:
        .long   123
bob:
        .long   456
```



# ALGORITHMS

## Python

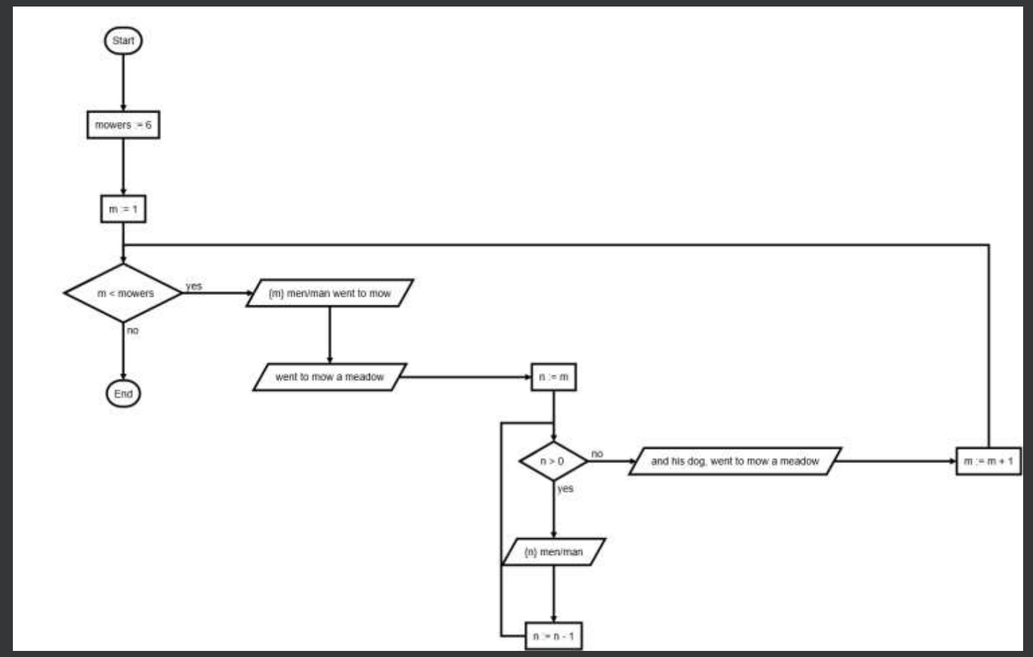
```
alice = 123

bob = 456

carol = alice * bob
```



# ALGORITHMS

## A non-trivial program

So what if we have a problem that’s more complicated than swapping two names, or multiplying two numbers?



# ALGORITHMS

## Went to Mow a Meadow

One man went to mow, Went to mow a meadow,

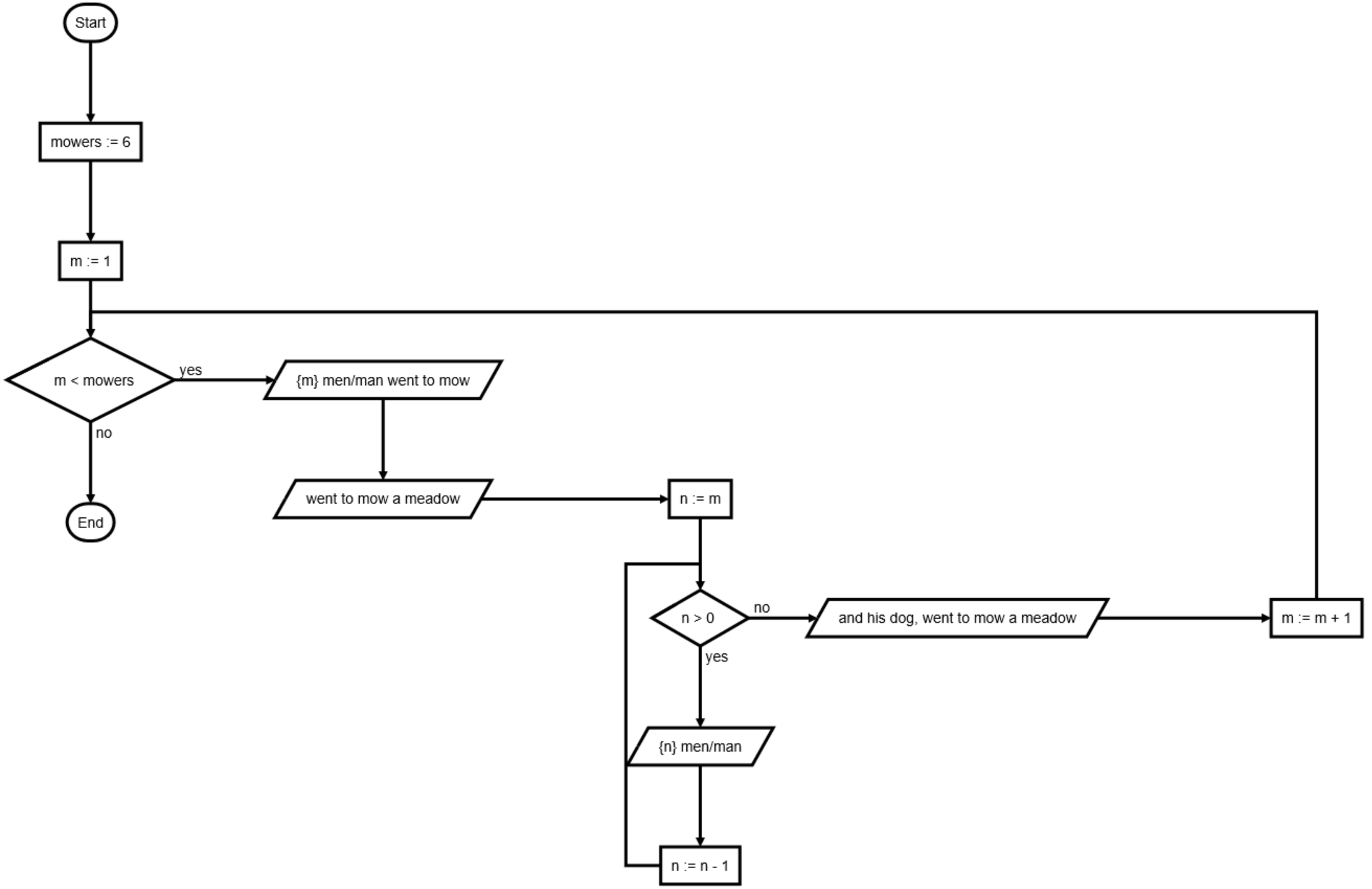
One man and his dog, Went to mow a meadow.

Two men went to mow, Went to mow a meadow,

Two men, one man and his dog, Went to mow a meadow.

Three men went to mow, Went to mow a meadow,

Three men, two men, one man and his dog, Went to mow a meadow.


Start
mowers := 6
m := 1
m < mowers
yes
no
{m} men/man went to mow
went to mow a meadow
n := m
n > 0
no
yes
and his dog, went to mow a meadow
m := m + 1
{n} men/man
n := n - 1
End

# ALGORITHMS

## Flowchart symbols

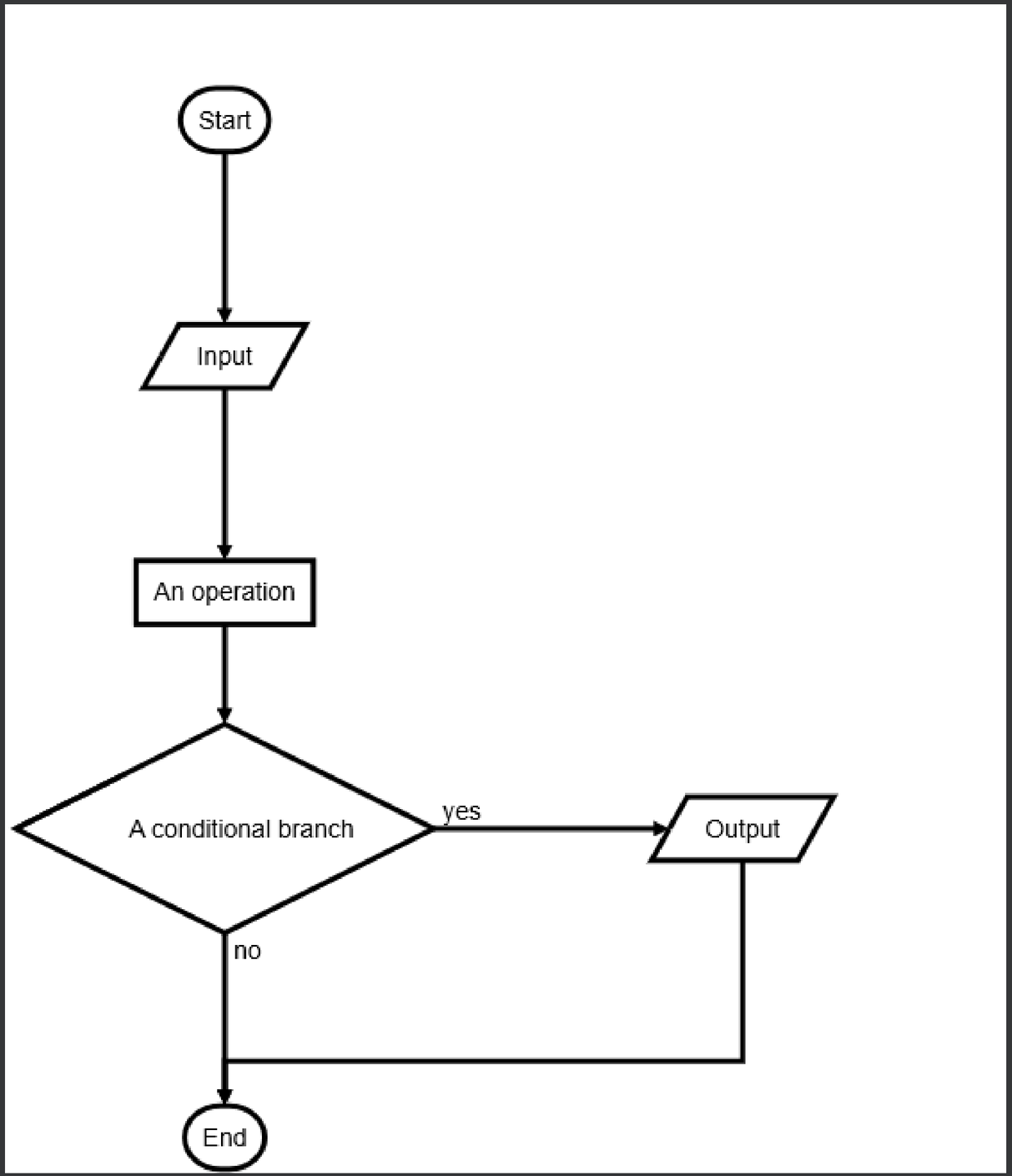
Start and end – circle or rounded

Input and output – parallelogram

Operation – rectangle

Conditional branch – diamond

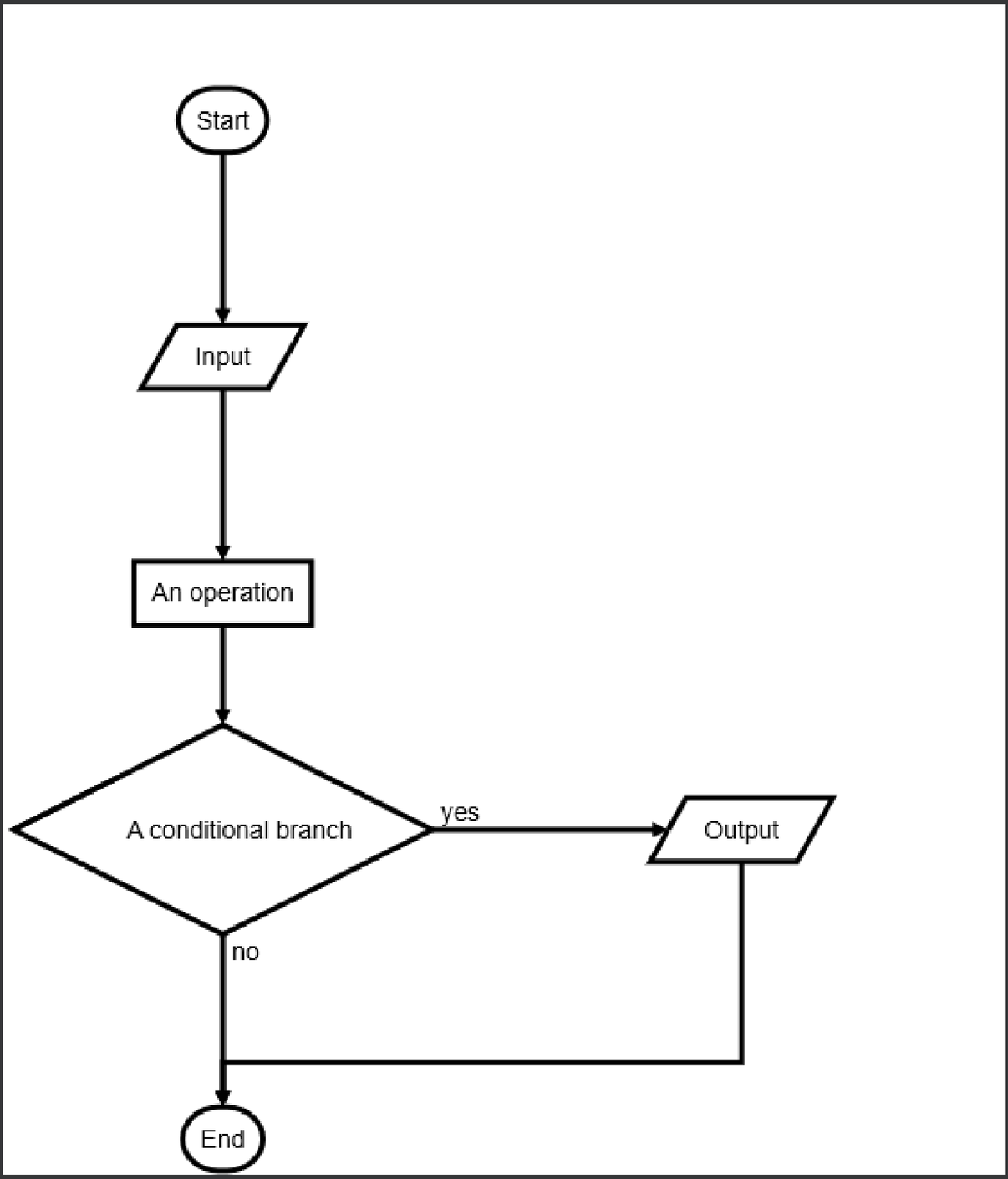
You don’t have to use these, but it can’t hurt to have a standard.



# ALGORITHMS

## As Python

```
value = input()
value = value.upper()

if value is 'BYE':
    print('Goodbye!')
```



# ALGORITHMS

## Turing Completeness

Alan Turing, mathematician and pioneer of computer science.

A set of rules is Turing complete, or computationally universal, if those rules can be used to simulate any Turing machine.

# ALGORITHMS

## Turing Completeness

It might surprise you to know that we've already encountered enough Python to have a Turing complete set of rules.

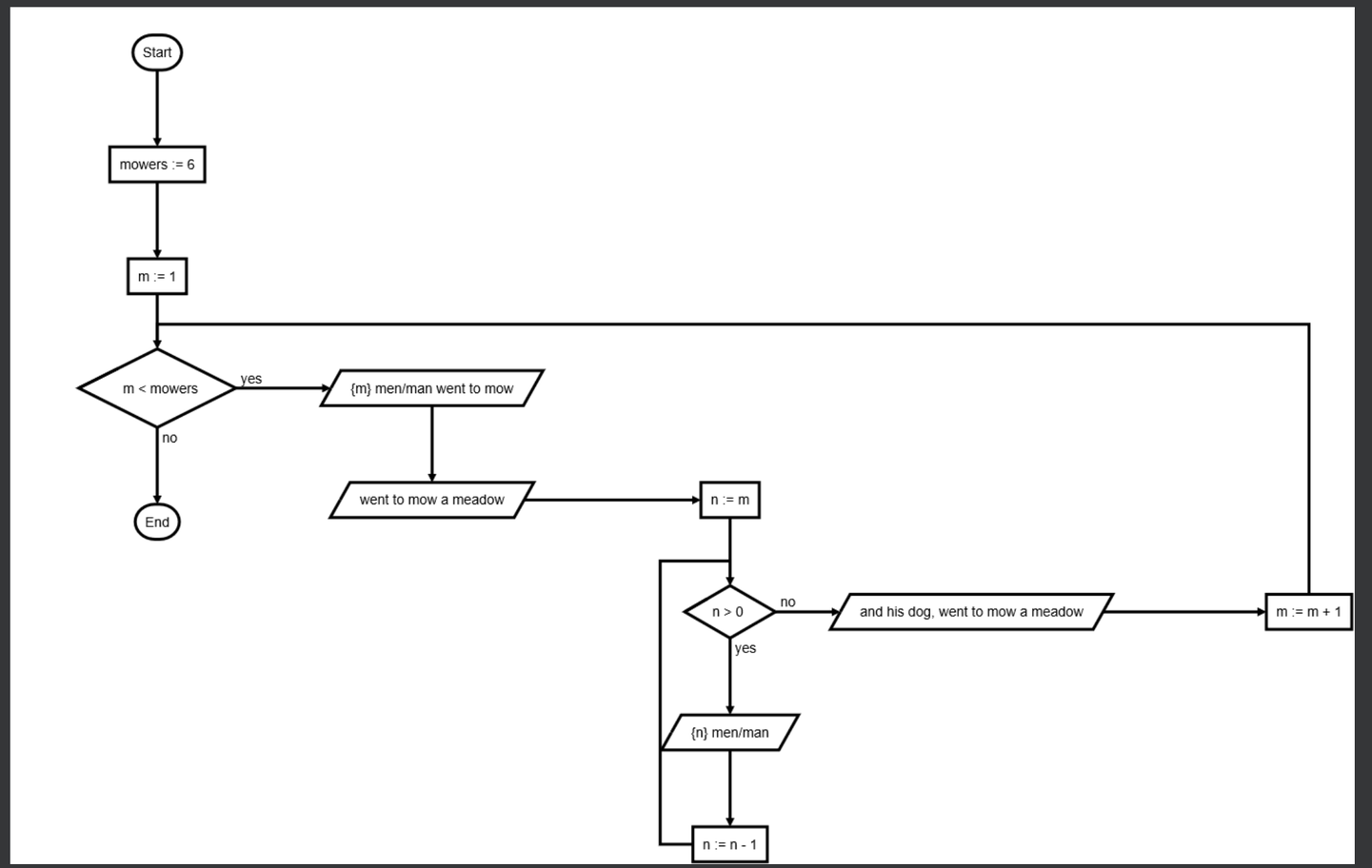
- Conditional branching
- Ability to change memory (variables)

# ALGORITHMS

## Exercise – Went to mow a meadow

What do we need?

- Output (print) text
- Count up
- Count down

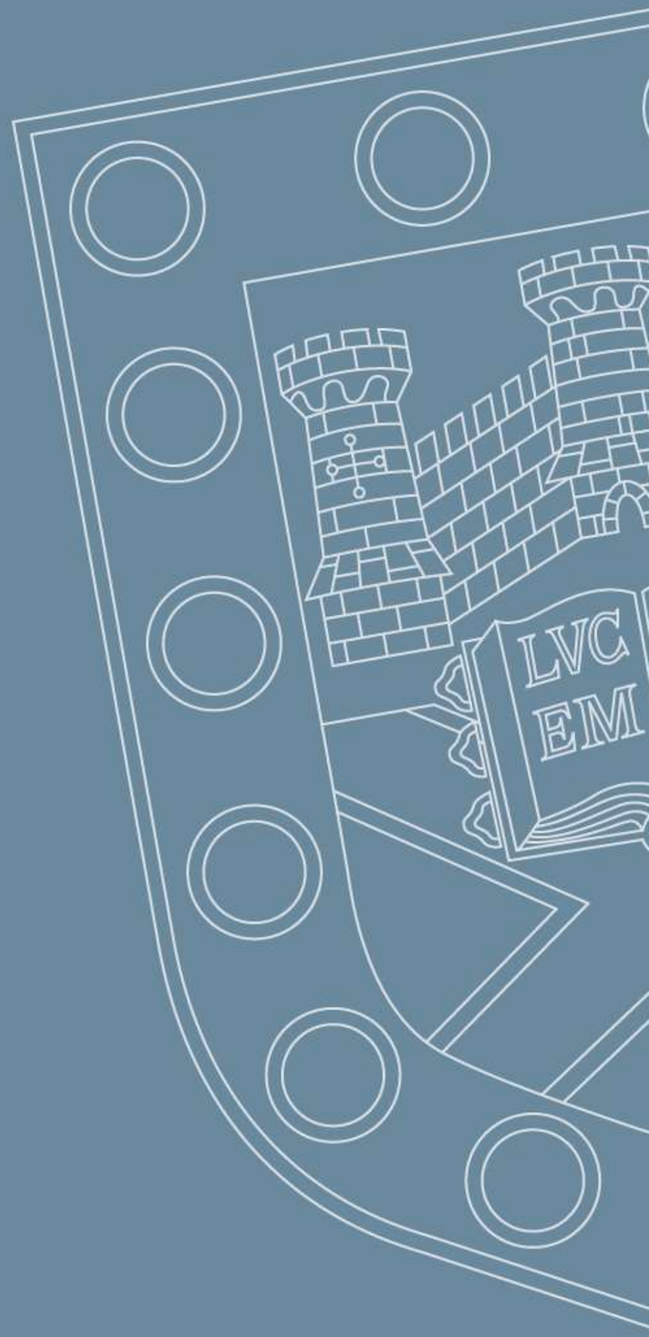


# ALGORITHMS

## Counting and printing

```
for m in range(4):
    print("m is", m)
    print("m plus 1 is", m + 1)
    print("4 minus m is", 4 - m)
```

UNIVERSITY OF EXETER




institute of CODING