# Algorithms

An **algorithm** is a sequence of ordered instructions that are followed step-by-step to solve a problem. This does *not* need to be on a computer.

**Decomposition** is the breaking down of a complex problem into smaller more manageable problems that are easier to solve.

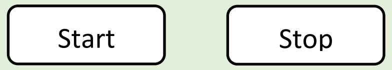
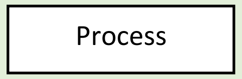
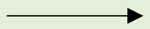
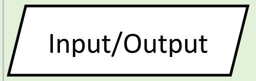
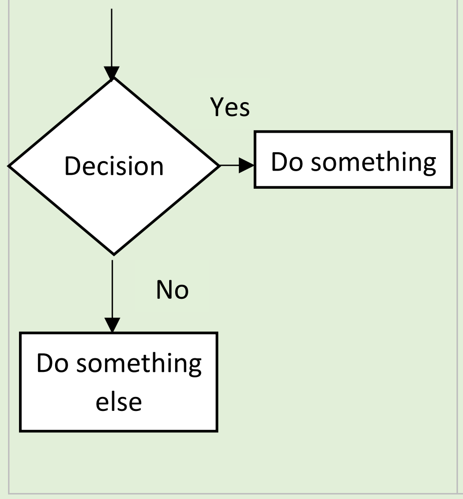
**Abstraction** allows us to remove unnecessary detail from a problem leaving us with only the relevant parts of a problem thereby making it easier to solve.

**Algorithm Efficiency** More than one algorithm can be used to solve the same problem. Normally we use the algorithm that solves the problem in the quickest time with the fewest operations or makes use of the least amount of memory.

**Dry run testing** is carried out using **trace tables**. The purpose of the trace tables is for the programmer to track the value of the variables and outputs at each step of the program and to track how they change throughout the running of the program.

## Flowchart Symbols

We can represent algorithms using flowcharts

| | |
|---|---|
| **Start and Stop**<br><br> | **Process – An operation that the algorithm performs**<br><br> |
| **Connector – Links all the other symbols together**<br> | **Input and Output of data that is read in and written out**<br><br> |
| **Decision is the same as a selection (if then … else)**<br><br> | IF answer is “yes” THEN<br>      do something<br>ELSE IF answer is “no”<br>      do something else<br>ENDIF |

## Pseudocode

We can represent algorithms using pseudocode

| | Example | Python equivalent |
|---|---|---|
| **Variable assignment** | a ← 10 | a = 10 |
| **Constant assignment** | constant PI ← 3.142 | PI = 3.142 |
| **Input** | a ← USERINPUT | a = input() |
| **Output** | OUTPUT “Bye” | print(“Bye”) |
| **Arithmetic Operators**<br><br>Add<br>Multiply<br>Divide<br>Subtract<br>Integer division<br>Modulus (remainder) | <br><br>+<br>*<br>/<br>–<br>a ← 7 DIV 2<br>a ← 7 MOD 2 | <br><br>+<br>*<br>/<br>–<br>a= 7 // 2<br>a = 7 % 2 |
| **Relational Operators**<br><br>Less than<br>Greater than<br>Equal to<br>Not equal to<br>Less than or equal to<br>Greater than or equal to | <br><br><<br>><br>=<br>≠ or <><br>≤<br>≥ | <br><br><<br>><br>==<br>!=<br><=<br>>= |
| **Boolean Operators**<br><br>AND<br>OR<br>NOT | <br><br>AND<br>OR<br>NOT | <br><br>AND<br>OR<br>NOT |
| **Selection**<br><br>if .. | <br><br>IF i > 2 THEN<br> j ← 10<br>ENDIF | <br><br>if i > 2:<br> j=10 |
| if .. else … | IF i > 2 THEN<br> j ← 10<br>ELSE<br> j ← 3<br>ENDIF | if i > 2:<br>  j=10<br>else:<br>  j=3 |
| if … else if … else | IF i ==2 THEN<br> j ← 10<br>ELSE IF i==3<br>THEN<br> j ← 3<br>ELSE<br> j ← 1<br>ENDIF | if i ==2:<br> j=10<br>elif i==3:<br> j=3<br>else:<br> j=1 |
| **Iteration**<br><br>While loops | <br><br>a ← 1<br>WHILE a < 4<br>  OUTPUT a<br>  a ← a + 1<br>ENDWHILE | <br><br>while a<4:<br> print(a)<br> a=a+1 |
| For loops | FOR a ← 0 TO 3<br>    OUTPUT a<br>ENDFOR<br>a ← 1 | for a in<br>range(3):<br> print(a) |
| Repeat loops | REPEAT<br>  OUTPUT a<br>  a ← a + 1<br>UNTIL a←4 | |
| **Subroutines**<br><br>procedure | <br><br>SUB hello()<br> OUTPUT “hello”<br>ENDSUB | <br><br>def hello():<br> print(“hello”) |
| Function (with paramerters and return) | SUB add(n)<br> a ← 0<br> FOR a ← 0 TO n<br>  a ← a + n<br> ENDFOR<br> RETURN a<br>ENDSUB | def add(n):<br> a=0<br> for a in<br>range(n+1):<br>  a=a+n<br> return a |
| **Built-in functions**<br><br>Length of array | <br><br>LEN(a) | <br><br>len(a) |
| Random integer | RANDOM_INT(0, 9) | import random<br>random.randint(0,9) |