LINKS - LEX

[Programming Fundamentals using Python - Part 1](https://lex.infosysapps.com/app/toc/lex_auth_0125409616243425281061/overview)

[Programming Fundamentals using Python - Part 2](https://lex.infosysapps.com/app/toc/lex_auth_012734003600908288382/overview)

[Programming Fundamentals using Python - Science Graduates - Foundation Program](https://lex.infosysapps.com/app/toc/lex_auth_0127412552654028801068/overview)

<https://lex.infosysapps.com/app/toc/lex_auth_012807761472348160416/overview>

Program – Set of instructions to perform a specific task by computer.

Level of program

1. High Level
2. Low Level – directly interact with hardware

Searching

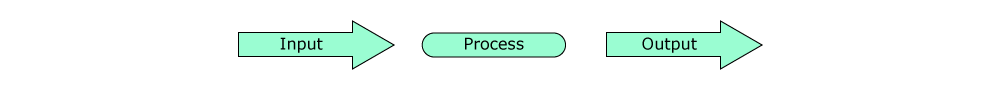
Counting

Optimization – limited resources due to which optimization needed

Sorting

Decision

How to solve problem

The solution to any problem involves the below three aspects:

Algorithm & its Representation

In mathematics and computer science, an algorithm is a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of problems or to perform a computation.

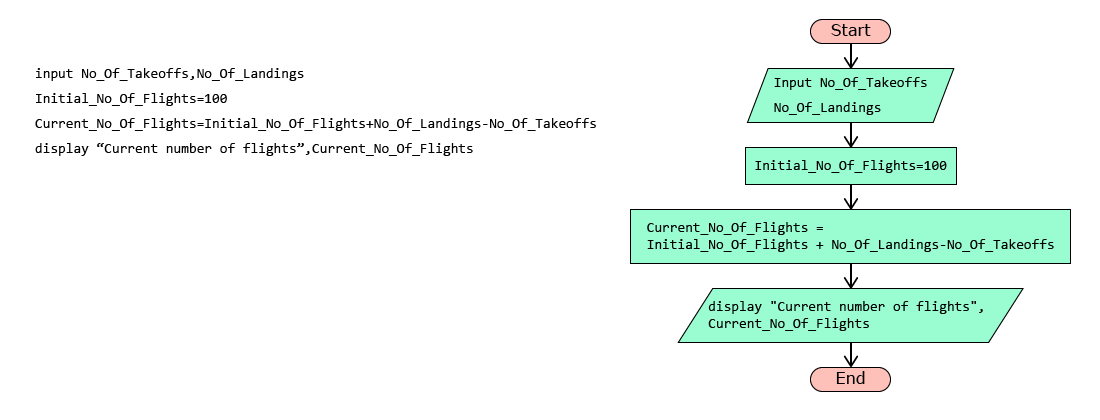
Representation of an algorithm

Algorithm should be represented from our mental thoughts into a form which others can understand. There are primarily two ways of representing an algorithm:

* Flow chart: Diagrammatic way of representing the algorithm.
* Pseudo-code: Representation of the algorithm in a way that is in between a program and normal English.

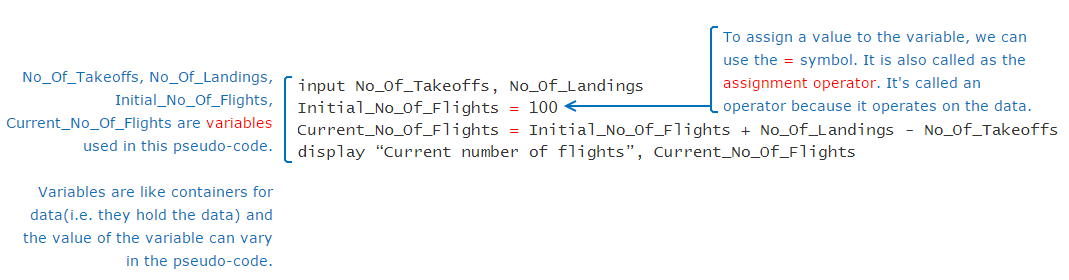
For example, here is a pseudo-code and a flowchart.

**Pseudo-code:**                                                                                                                                             **FlowChart:**



Pseudo-code cannot be executed by a computer. It is just a representation of an algorithm for us to understand.

Variables & Operators



Variable - Container for any value

Like assignment operator, there are other operators also which can be used to perform various operations.  
Arithmetic operators: Used for performing arithmetic operations

|  |  |
| --- | --- |
| **Operators** | **Description** |
| **+** | Addition |
| **-** | Subtraction |
| **\*** | Multiplication |
| **/** | Division |
| **%** | Modulus |

Relational operators: Also known as comparison operators, are used to compare values. Result of a relational expression is always either true or false.

|  |  |
| --- | --- |
| **Operators** | **Description** |
| **==** | Equal to |
| **<** | Less than |
| **>** | Greater than |
| **<=** | Less than or equal to |
| **>=** | Greater than or equal to |
| **!=** | Not equal to |

Logical operators are used to combine one or more relational expressions.

|  |  |
| --- | --- |
| **Operators** | **Description** |
| AND | Result will be true, if both the expressions are true. If any one or both the expressions are false, the result will be false |
| OR | Result will be true, even if one of the expression is true. If both the expressions are false, the result will be false |
| NOT | If the expression is true, result will be false and vice versa |

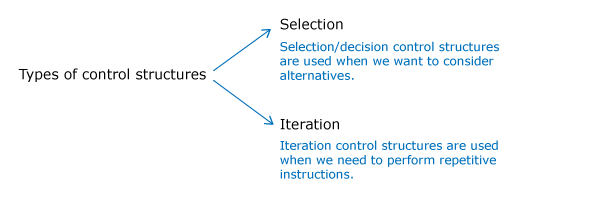
If A and B are two relational expressions, say A = (Num1>2000), B= (Num2>100), the result of combining A and B using logical operator is based on the result of A and B as shown below:

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **A AND B** |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

Decision Constructs

Flow of control

In a pseudo-code, typically the instructions are performed one by one or line by line. But there may be situations when all the statements in a pseudo-code are not performed. Parts of the pseudo-code which change the flow of instructions or in other word, change the flow of control are called as control structures.

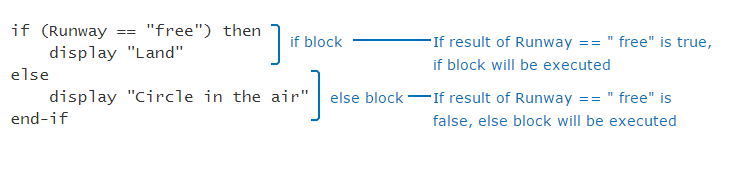


Selection using if statement

ATC takes lot of decisions as part of its air traffic control operations.

For example, if a flight is approaching the runway, ATC has to check if the runway is free. If the runway is not free, then the flight should not land immediately. It should circle in the air and wait for further instructions from the ATC.

Such decision making process can be conveniently represented in a pseudo-code using an if statement.



Sometimes, ATC may have more than one alternatives for a given situation. For example, if the runway is free, the flight can land. But if the flight has less fuel, then it should be allowed an emergency landing. Otherwise, it should circle in the air.

if (Runway=="free") then

   display "Land"

else if (Fuel\_Status=="low") then

   display "Emergency landing"

else

   display "Circle in the air"

end-if

Iteration Constructs

Immigration check needs to be done for all the passengers in the flight. Suppose the flight had only 5 passengers, the pseudo-code can be written as follows:

display "Flight has landed"

display "Proceed for Immigration Check"

Passenger\_Count=1

display "Immigration check done for passenger,", Passenger\_Count

Passenger\_Count=Passenger\_Count+1

display "Immigration check done for passenger,", Passenger\_Count

Passenger\_Count=Passenger\_Count+1

display "Immigration check done for passenger,", Passenger\_Count

Passenger\_Count=Passenger\_Count+1

display "Immigration check done for passenger,", Passenger\_Count

Passenger\_Count=Passenger\_Count+1

display "Immigration check done for passenger,", Passenger\_Count

Sometimes, we may want to execute some statements specific (known) number of times as in the case of immigration check. In such cases we can use a **for loop**as shown below.

No\_Of\_Passengers=5

for(Passenger\_Count=1,Passenger\_Count<=No\_Of\_Passengers,Passenger\_Count=Passenger\_Count+1)

     display "Immigration check done for passenger,", Passenger\_Count

end-for

When we want to repeatedly execute a statement as long as a condition is met, we can use the iteration statement called as **while loop**.

display "The flight has landed"

display "Immigration check done"

display " Collect the baggage from the conveyor belt"

Baggage\_Count=150

while(Baggage\_Count>0) do

    input No\_Of\_Baggage\_Picked

    Baggage\_Count=Baggage\_Count-No\_Of\_Baggage\_Picked

end-while

Go through the below two pseudo-codes and guess the output.

|  |  |
| --- | --- |
| **Pseudo-code 1 :** | **Pseudo-code 2 :** |
| 1. Counter=5 2. while(Counter>=5) do 3. display Counter 4. Counter = Counter + 1 5. end-while | 1. input Limit 2. for(Counter=5, Counter<=Limit, Counter=Counter-1) 3. display Counter 4. end-for   Assume that the input value provided to variable Limit is |

Beware of infinite loops. The logic that we are writing in loops should ensure that the loop will terminate in finite number of iterations.

What is the outcome of the following pseudo-code?

input Counter

while(Counter<5) do

  Counter=Counter+1

  display Counter

end-while

Assume that the input value provided to variable, Counter is 1.

2,3,4,5

2,3,4

1,2,3,4

1,2,3,4,5

Write a pseudo-code to move from the start (S) to the end (E) in the maze.

**Note**: You can drag and drop the pseudo-code magnets to the pseudo-code box and create the appropriate pseudo-code.

**Estimated Time: 10 minutes**

#### Pseudo-code Magnets:

for(counter=            , counter<=                            , counter=          )

end-for

move  

#### Pseudo-code:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **S** |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | E |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |