**ALIENS PROGRAMMING LANGUAGE**

Documentation

Version 0.1

Friday, August 28, 2020

//Content of documentation

1. Introduction:

we have made this language to make programming as easy as possible for new learners, who have no experience in coding.

we coined this language as ‘Aliens ' programming language and the abbreviation is "Als” which we will use a lot in our programming journey.

we also designed this language to be familiar to the user especially those who already have an experience in coding, we got inspired the structure and the syntax from the famous and successful languages such as (dart, python, C#, JavaScript ...), so it inherits many of the same statements and expressions form those.

* 1. The reason why we name our language "Aliens Programming Language":

before talking extensively about the syntax and the structure of the language we should put you in the ground and reasons behind choosing this name for our language :

On 10 August 2020, our team participated in one of the biggest events that Repit organized with a prize of 10000 $ for the winning team.

our team started the brainstorming on how our language should look like after hours of negotiation and discussion, we finally agreed that the language should present something that we all know, even those how never used a computer before .the idea is space.

you must be wondering what is the relevance of "space" with computer language, alright we will explain.

the first thing you should know is the Aliens language hierarchy, well.

imagine yourself you are cruising space; in your way you may discover new planets or new galaxies. how knows everything is possible over there, so the thing is our language converts all this word or space jargon if we could say to names that have a meaning in our language.

eg: a user could create a planet with its moons and of course inside a space which contains many galaxies, it is simple isn't it?

we think now it's become more obvious for you where the name of "Aliens programming language” came from.

1. Pre-requirements:

all you need to make this language work is <https://www.python.org/>

, and Linux.

“unfortunately, this language only works in Linux for the moment. “

1. Create new project:

now let create our first project which we will name as test :

>>>> #this is how we create our project

>>>> als createProject Test

>>>>cd Test

>>>>cat main.als

Space(){

Show(‘hello world’)

}

>>>>

As you see in the example above, we created our “Test” project successfully, inside Test directory we have the file main.als as well. you noticed the keyword space. this is our entry point which will be responsible for displaying "hello world".

1. Variables in Aliens

variables are important to store your information temporarily in the computer ram in order to use them again, there is a lot of types :

* 1. Numbers

we use this type of variables when we want to represent both integer and floating-point numbers.

a=12+20

show(‘the result of this addition is’+a)

#the result will be something like this:

-------------------------------------------------------------------------------------------------------------------------------------------

The result of the addition is 32

As you see in the example above we declare a variable which we called a to get the addition result of two integers, after that we used show function to display the result to the user.

**Note:**

We use **#**  this symbol to say this is only comment .

* 1. Strings

we use this type of variables when we want to represent any kind of text, but it must be surrounded by quotes. Or double quotes.

Myname=’Adam’

Show(‘my name is ‘+Myname)

#the result will be something like this.

-------------------------------------------------------------------------------------------------------------------------------------------

My name is Adam

What we did here is no difference from what we did in the example above of number so we declare a variable of type string, after that we display it .

* 1. Lists:

A list is a type of object used for storing multiple values in single variable ,Each value (also called an element or item ) in a list has a numeric position, known as its index, and it may contain data of any data type-numbers, strings, Booleans … and even other lists or dictionaries . The List index starts from 0, so that the first array element is lis[0] not lis[1].

Countries=[‘morocco’,‘usa’,’Canada’, ‘France’]

#this is a list of countries

Show(“I’m from “+Countries[0])

-------------------------------------------------------------------------------------------------------------------------------------------

I’m from morocco

As you see the result of this example is “Morocco”, this because we choose the first element .

1. Conditions

A condition is an expression that evaluates whether something is true or false. When the value of a condition is true, we say that this condition is satisfied.

* 1. If condition statement

If (true) {

Show( «hello i’m from earth »)

}

-------------------------------------------------------------------------------------------------------------------------------------------

Hello i’m from earth

In this case the condition is true, so it displays” hello I’m from earth” as a result .

* 1. If/else conditions statements

If (false) {

Show( «hello I’m from earth »)

} :{

Show(« hey I’m from mars »)

}

-------------------------------------------------------------------------------------------------------------------------------------------

hey I’m from mars

in this example we have two conditions, the first one is if (false) which is not true the second one represents “else” of the “if” condition which we symbolized it with “:” .and this means if the first condition not true than the second one will be true.

The result of this example is “hey I’m from mars”.

* 1. If/else if …./else conditions statement

we use usually this kind of conditions to check a chain of conditions if they are true or not.

It is very important to understand that once a condition is found to be true, no other if statements are evaluated and once the code block for the true statement is completed, the program continues from the end of the if/else if statement.

\_name=”mars”

if(\_name == "earth"){

\_name = “EARTH"

} :(\_name == "pluto") {

\_name = "PLUTO"

}:(\_name == “mars") {

\_name = "MARS"

} :{

\_name = "UNKNOWN"

}

-------------------------------------------------------------------------------------------------------------------------------------------

MARS

**Note:** in this case “else if” equivalent to :(condition){} and else is equivalent to :{} this how we know the difference between them.

1. Operators (&&,||,<,>,<=,>=,…..)

operators are used to assign values, compare values, perform arithmetic operations, and more.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Addition | X=2+3 |
| - | Subtraction | Y=2-7 |
| \* | Multiplication | A=7\*9 |
| / | Division | D=11/2 |
| % | Modulus (division remainder) | M=9%3 |
| ++ | Increment | X++ |
| -- | Decrement | Y-- |
| = | equal | Y=5 |
| == | equal to | X==4 it will return true |
| != | Not equal to | X!=3 it will return true |
| > | Greater than | x>2 |
| < | Less than | X<3 |
| >= | Greater than or equal | x>=2 |
| <= | Less than or equal | X<=11 |
| && | and | (x>3 && y<=4) |
| || | or | (x>3 || y<=4) |
| ! | not | !(x==y) |

1. Loop Statements

Loops are handy, if you want to run the same code over and over again, each time with a different value.

* 1. @loop(start=0,end) as n{...}

We will use this loop to repeat this santace “i’m from earth” 10 times .

See the example in the next page.

Loop(0,10) as index{

Show(« i ‘m from earth »+index)

}

------------------------------------------------------------------------------------------------------------------------------------------

I’m from earth 1

I’m from earth 2

I’m from earth 3

I’m from earth 4

I’m from earth 5

I’m from earth 6

I’m from earth 7

I’m from earth 8

I’m from earth 9

I’m from earth 10

As you see above the result of our loop we repaet the santance 10 time ,all you have to do is to call the loop and give it a start number and end number.

* 1. @while(condition){..}

This type of Loops can execute a block of code as long as a specified condition is true.

i=0

while(i<3){

show(« i’m alien « )

i++ ;

}

------------------------------------------------------------------------------------------------------------------------------------------

I’m alien

I’m alien

I’m alien

this loop will only stop when the condition is true, In this case we got “I’m alien “ 3 times.

* 1. Enumerable.loop(item){...}

Loop(item) method executes a provided function once for each array element.

Countries=[‘morocco’,‘usa’,’Canada’, ‘France’]

Countries.loop(item){

If(item==’morocco’){

Show(“I’m alien from ”+item)

}

}

-----------------------------------------------------------------------------------------------------------------------------------------

I’m alien from morocco

after we declared the list of countries, we wanted to check first if there is an item has a value of 'morocco’. The result of the condition is true, therefore, it displays the sentence of "I'm alien from morocco'.

* 1. enumerable.loop(e,item){..}

this function takes two parameters the first one is where should the loop start ,and the second one is the name of the enumerable object.

See the example in the next page.

Countries=[‘morocco’,‘usa’,’Canada’, ‘France’]

Countries.loop(1,item){

If(item==’morocco’){

Show(“I’m alien from ”+item)

}

}

-----------------------------------------------------------------------------------------------------------------------------------------

In this case it will display nothing because, the loop starts from the second item of the list which is “usa”, therefore the condition will not recognize the first item which is morocco.

1. Defining Functions

In Aliens programming language functions are defined with $ keyword

There are two types of functions the first one is without parameters and the second one with the parameters.

let’s create our first functions:

#function without parameters

$Sayhey{

Show(‘hey ‘)

}

#function with parameters

$SayHello firstname lastname {

Show(“hello mr :”+firstname+” “+lastname)

}

1. How to import a Model?

In this language, we changed a little bit the flavor to make it easier for all of us to learn from and contribute to each other's code. this way we define a new ecosystem which we call models or galaxies.

models are a set of planets and functions that a program can use in order to make it easy for developers.

* 1. Import a model to your project

By default, you will have all the models in your project in case you don’t just check that nothing is missing.

from https://www.site.com/wem.mals load Saad

from @base load @convert, @math

load SaadModule

load @base from ../../file.mals load ThisPlanetA

using @convert

Space(){

}

1. Standard library
   1. Type Checking Functions

|  |  |  |
| --- | --- | --- |
| Function | Mining | example |
| $isnumber | return true if value is number | num= 10  test = $isnumber(num)  show(test)  >>>1 |
| $isalpha | return true If all characters in the string are alphabet. | Text= “Aliens”  test = $isalpha(Text)  show(test)  >>>1 |
| $isnumber | return true if value is number | num= 10  test = $isnumber(num)  show(test)  >>>1 |
| $isequal | return true if two values are equals | Text1= “mars”  Text2= “earth”  test = $isequal (Text1, Text2)  show(test)  >>>0 |
| $ismatch | return true if value respect regex expression | Text1= “Aliens”  test = $ismatch(text, “[A-z]\*”)  show(test)  >>>1 |

* 1. Strings Functions

|  |  |  |
| --- | --- | --- |
| Function | Mining | example |
| [str].lower() | return text to lowercase | Text= “ALIENS”  newText = Text.lower()  show(newText)  >>>aliens |
| [str].upper() | return text to uppercase | Text= “Aliens”  newText = Text.upper()  show(newText)  >>>ALIENS |
| [str].split(character/text) | return splitting text by character or text | Text= “Aliens Language”  newText = Text.split(“ ”)  show(newText[0])  >>> Aliens |
| [str].len() | return length of text | Text= “Aliens”  length = Text.len()  show(length)  >>>6 |
| [str].count(value) | return number of times the text is present | Text= “Aliens Language”  times = Text.count (“e”)  show(times)  >>> 2 |

* 1. List basic Functions

|  |  |  |
| --- | --- | --- |
| Function | Mining | example |
| [list].add(object) | add object to list | num= 10  test = $isnumber(num)  show(test)  >>>1 |
| [list].insert(object,index) | insert object into list | Text= “Aliens”  test = $isalpha(Text)  show(test)  >>>1 |
| [list].remove(object) | remove object from list | num= 10  test = $isnumber(num)  show(test)  >>>1 |
| [list].reverse() | return list reverse | Text1= “mars”  Text2= “earth”  test = $isequal (Text1, Text2)  show(test)  >>>0 |
| [list].size() | return size of list | Text1= “Aliens”  test = $ismatch(text, “[A-z]\*”)  show(test)  >>>1 |
| [list].clear() | Delete all element of list |  |

Chapiter 5: Introduction

Chapiter 6: Introduction

Chapiter 7: Introduction