Displaying output in Julia

```
println("Yedukondalavaada, Venkataramana, Govinda, Gooooooovinda")
```

```
Yedukondalavaada, Venkataramana, Govinda, Gooooooovinda
```

The `print()` command is used to display output to the console

```
begin
    text = "The quick brown fox jumps over the lazy dog"
    print(text)
end
```

```
The quick brown fox jumps over the lazy dog
```

`print()` command, the output is shown in the console. The next `print()` command continues in the same line.

The `println()` command prints and moves the print cursor to the next line.

```
#Comments in Julia are done with a Hashtag, just like how it is done in Python
```

Declaring and Intializing variables in Julia

Variables in Julia can be declared by just writing their name. There's no need to define a datatype with it. Initializing variables can be done at the time of declaring variables. This can be done by simply assigning a value to the named variable.

```
begin
    a = 10
    b = "Random String"
    println(a)
    println(b)
end
```

```
10
Random String
```

Rules for naming a variable in Julia:

- Variable names in Julia must start with an underscore, a letter(A-Z or a-z) or a Unicode character greater than 00A0(nbsp).
- Variable names can also contain digits(0-9) or !, but must not begin with these.
- Operators like (+, ^, etc.) can also be used to name a variable.
- Variable names can also be written as words separated by underscore, but that is not a good practice and must be avoided unless necessary.

```
begin
    # Julia program to define variables

    # Assigning Integer
    x = 10

    # Assigning String
    y = "Hello World"

    # Assigning Float Value
    z = -15.5

    # Using Operator as variable name




    println(x)
    println(y)
    println(z)

end
```

```
10
Hello World
-15.5
```

In Julia, in order to find the datatype, you use the `typeof()` command

```
begin
    println(typeof(x))
    println(typeof(z))
end
```

```
Int64
Float64
```

Changing Datatypes in Julia

In order to cast the string to a integer, you use the parse function. `parse(Int64, num)`

```
begin
    num = "69.69"
    println(typeof(num))
    num = parse(Float64, num)
    println(typeof(num))
end
```

```
String
Float64
```

Functions in Julia

A function is defined with the following syntax

```
function function_name(var1, var2) Function Body return value end
```

A function is called using the traditional paranthesis syntax.

```
begin
    # Defining a function
    function print_ka_function(integer)
        println("this is a function")
        println(integer)
    end

    # Function call
    print_ka_function(3)
    print_ka_function(4)
end
```

```
this is a function
3
this is a function
4
```

```
func (generic function with 1 method)
```

```
#Defining a function with inputs

function func(x,y)
    output = x + y
    return output
end

```

```
begin
    random_variable = func(10,20)

    println(random_variable)
end
```

```
30
```

Writing a Program to solve the Quadratic Equation

The roots of a Quadratic Equation are given by the following formula

$$x = \frac{-b \pm \sqrt{(b^2 - 4 * a * c)}}{2 * a}$$

Where a, b and c are the coefficients of x^2, x and the constant respectively.

Intialize the variables, don't bother with user input.

Solution:

```
let
    #Intializing the Variables
    a = 5
    b = 6
    c = 10
    println("The coefficient of x^2 is ", a)
    println("The coefficient of x is ", b)
    println("The constant is ", c)

    #Calculating the Discriminant
    d = complex(b^2 - 4 * a * c)
    d = d ^ 0.5

    #Calculating the first root
    root1 = -b + d
    root1 = root1/(2 * a)

    #Calculating the second root
    root2 = -b - d
    root2 = root2/(2 * a)

    #Displaying the roots
    println("The Two roots are")
    println(root1)
    println(root2)

end
```

```
The coefficient of x^2 is 5
The coefficient of x is 6
The constant is 10
The Two roots are
-0.6 + 1.2806248474865698im
-0.6 - 1.2806248474865698im
```