# Julai Programming exercise

In [9]:

```
a = 10;
print("value of a is = ", a)
#print("value of a is = {}".format(a)) // not working
```

```
value of a is = 10
```

In [35]:

```
c = 10;
#like in matlab ";" used to supress the output
```

In [36]:

```
c = 10
```

Out[36]:

```
10
```

# 1. Variables

In [11]:

```
typeof(a)
```

Out[11]:

```
Int64
```

In [14]:

```
a = "Julia"
```

Out[14]:

```
"Julia"
```

In [15]:

```
typeof(a)
```

Out[15]:

```
String
```

In [18]:

```
a = "\alpha"; # then press tab
a  = "α"
```

Out[18]:

```
"α"
```

In [19]:

```
typeof(a)
```

Out[19]:

```
String
```

In [20]:

```
δ = 10;
δ + 100
```

Out[20]:

```
110
```

In [21]:

```
pi
```

Out[21]:

```
π = 3.1415926535897...
```

In [37]:

```
pi = 10
# can't used predefined variable  as variable in julia
```

```
cannot assign variable MathConstants.pi from module Main

Stacktrace:
 [1] top-level scope at In[37]:1
```

In [38]:

```
sqrt(144)
```

Out[38]:

```
12.0
```

In [39]:

```
sqrt = 10
# can't used predefined function name  as variable in julia
```

```
cannot assign variable Base.sqrt from module Main

Stacktrace:
 [1] top-level scope at In[39]:1
```

# 2. Numbers

In [54]:

```
a = 2;
print(typeof(a))
print("\n")
a = 3.2;
print(typeof(a))
print("\nword size of the operating sys is", Sys.WORD_SIZE )
```

```
Int64
Float64
word size of the operating sys is64
```

Int64 :- Here 64 means julia is running on 64 bit operating system

In [55]:

```
print(4 / 5 )
print("\n")
print(div(4,5))
print("\n")
print(div(5,2))
```

```
0.8
0
2
```

In [56]:

```
3 * 4
```

Out[56]:

```
12
```

In [57]:

```
4 ^ 3
```

Out[57]:

```
64
```

In [58]:

```
12%10
```

Out[58]:

```
2
```

In [68]:

```
print(2 | 4)
# bitwise OR operator
```

```
6
```

In [71]:

```
print(3 & 5)
# bitwise AND operator
```

1

In [74]:

```
print(6 >> 1)
# bitwise right sift operator
```

3

In [79]:

```
# maximum number supported by Int64
typemax(Int64)
```

Out[79]:

9223372036854775807

In [81]:

```
# minimum number supported by Int64
typemin(Int64)
```

Out[81]:

-9223372036854775808

In [84]:

```
# maximum number supported by Float64
typemax(Float64)
```

Out[84]:

Inf

Inf means infinity

In [85]:

```
1/0
```

Out[85]:

Inf

In [86]:

```
0/0
```

Out[86]:

NaN

NaN means Not a munber / undefined

In [94]:

```
x = 5;
a = 2 * x^2 - 3 * x + 1;
print("Normal ", a)
a = 2x^2 - 3x + 1;
print("\nMore readable feature fo Julia ", a)
```

```
Normal 36
More readable feature fo Julia 36
```

# 3. Complex and Rational Numbers

In [96]:

```
a = 3 + 4im
```

Out[96]:

```
3 + 4im
```

In [98]:

```
typeof(a)
```

Out[98]:

```
Complex{Int64}
```

In [111]:

```
# real part of complex number
real(3 + 4im)
```

Out[111]:

```
3
```

In [112]:

```
# complex part of complex number
imag(3 + 4im)
```

Out[112]:

```
4
```

In [113]:

```
# like in matlab i here isnot for imaginary part
# since julia support mathematical coefficient so result will be 13
i = 5;
3 + 2i
```

Out[113]:

```
13
```

In [117]:

```
# angle of complex number in degree
angle(3 + 4im) * 180 / pi
```

Out[117]:

```
53.13010235415598
```

In [119]:

```
# magnitude of complex number
abs(3 + 4im)
```

Out[119]:

```
5.0
```

In [122]:

```
# complex multiplication
(3 + 4im)*(6 + 7im)
```

Out[122]:

```
-10 + 45im
```

In [125]:

```
# execution fo complex expresion
(3 + 4im)^(6 + 7im) / (1 + 1im)
```

Out[125]:

```
-15.820048216222766 - 5.53347341244951im
```

In [126]:

```
# Making complex number
complex(50,30)
```

Out[126]:

```
50 + 30im
```

expressing rational number without expressing in decimal form

In [127]:

```
y = 2//3
```

Out[127]:

```
2//3
```

In [128]:

```
typeof(y)
```

Out[128]:

```
Rational{Int64}
```

In [129]:

```
y = 2/3
```

Out[129]:

0.6666666666666666

In [134]:

```
4//6
```

Out[134]:

2//3

In [136]:

```
8//12
```

Out[136]:

2//3

In [137]:

```
float(8//12)
```

Out[137]:

0.6666666666666666