Solution of Laboratory Work 2

Exercise 1 :

```
--> a = 7; b = 3;
--> sum = a + b;
--> difference = a - b;
--> product = a * b;
--> quotient = a / b;
--> a = 5.2; b = 1.8;
--> sum = a + b;
--> difference = a - b;
--> product = a * b;
--> quotient = a / b;
```

Exercise 2 :

```
--> integer = 5;
--> integer_type = typeof(integer);
--> real = 3.14;
--> real_type= typeof(reel);
```

Scilab treats integers and real numbers as **`constant`** data, meaning they are of the **double** data type, which is used for floating-point numbers.

Exercise 3 :

```
--> real_number = 3.14159265359
 real_number  =
               3.1415927
--> format(5)
--> real_number
 real_number  =
               3.14
--> format(26)
format: Wrong value for input argument #1: Must be in
the interval [2, 25].
--> floor(real_number)
 ans  =
        3.
--> ceil(real_number)
 ans  =
        4.
--> round(real_number)
 ans  =
        3.
```

Exercise 4 :

```
--> 1/sqrt(8^3+2)-2*sind(45)/exp(2)+log(4)
--> A=1/sqrt(8^3+2);
--> B=2*sind(45)/exp(2);
--> C=log(4);
--> disp(A-B+C)
```

Exercise 5 :

```
--> small_number = 1.230D-07
--> large_number= 1.234567890D+9
```

Exercise 6 :

The range of real numbers in double precision goes from approximately 2.2×10^{-308} to 1.8×10^{308}.

```
--> positive_number = 1.23456789012345D-100;
--> negative_number = -1.23456789012345D-100;
```

Exercise 7 :

```
--> epsilon = %eps
 epsilon =
           2.220D-16
--> result_eps = 1 + epsilon
 result_eps =
               1.
-->%eps+1==1
 ans  =
        F
```

In Scilab, double-precision numbers can range roughly from 2.2×10^{-308} to 1.8×10^{308}. The constant **%eps** ($\approx 2.22 \times 10^{-16}$) does not represent the smallest number in this range, but instead measures the precision of floating-point calculations, it is the smallest number that can, in theory, makes **1 + %eps** slightly greater than **1**. It is used to assess the precision of numerical calculations.

```
--> infinity = %inf
 infinity =
            Inf
--> result_inf_positive = infinity + 1000

 result_inf_positive =
                         Inf
--> result_inf_negative = infinity - 1000
  result_inf_negative =
                         Inf
```

Operations with **%inf** behave in such a way that they generate special results. Adding **%inf** to a positive number gives **%inf**, and adding **%inf** to a negative number also gives **%inf**. This reflects the behavior of mathematical infinities.

Exercise 8 :

```
--> z1=complex(2,1), z2=complex(1,2)
--> addition = z1 + z2,  subtraction  = z1 - z2
--> multiplication  = z1 * z2, division = z1 / z2
--> z1_bar=conj(z1),  real_z2 = real(z2)
--> imaginary_z2 = imag(z2)
--> modulus_z1 = abs(z1)
--> argument_z1 = atan(imag(z1), real(z1))
--> [radius, angle]=polar(z1)
--> figure(0)
--> clf()
--> hf=gcf()
--> hf.background=-2
--> ha=gca()
--> ha.data_bounds=[-5 -5; 5 5]
--> xgrid()
--> plot([0 2],[0 1],'b','LineWidth',3)
--> plot([0 1],[0 2],'r','LineWidth',3)
--> xlabel('Real axis (Re)','FontSize',2)
--> ylabel('Imaginary axis (Im)','FontSize',2)
--> legend('$\Large{z_{1}}$','$\Large{z_{2}}$')
--> plot(0,0,'sk')
--> plot(2,1,'sk')
--> plot(1,2,'sk')
--> xstring(2,1,'$\Large{z_{1}=2+i}$')
--> xstring(1,2,'$\Large{z_{2}=1+2i}$')
--> title('Complex numbers','FontSize',1)
```