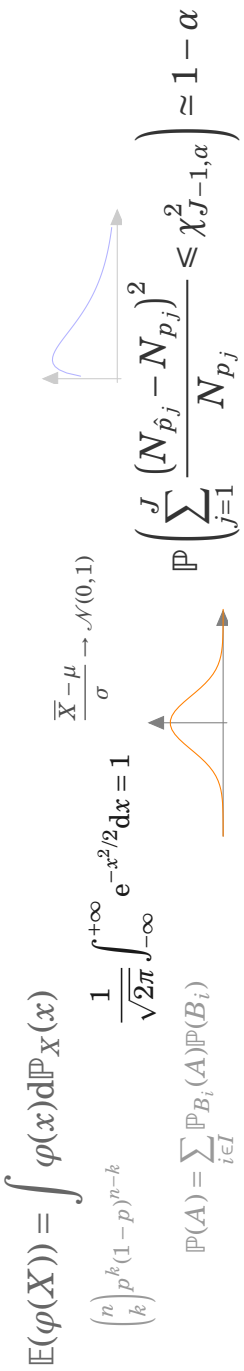

1 Read the help file and try out every command you see in it.

2 Let `A=[1 2]` and `B=[3 4]`. Try the following commands in scilab's console : `A', A * B, A*B', A'*B, A.*B`.

3 Write a program that takes as an argument a number d and gives the area of a circle of diameter d. Your program will give an error message if the diameter is negative.

4 Fibonacci's sequence $(F_n)_n$ is characterized by the fact that every number after the first two is the sum of the two preceding ones :

$$\begin{cases} F_0 = 0,\ F_1 = 1, \\ \forall n \geqslant 2,\ F_n = F_{n-1} + F_{n-2}. \end{cases}$$

Write a program that takes n as an argument and that outputs F_n.
If you're up to it you can try a recursive version !

5 Plot the sine and cosine functions on the same graphe with different colors and add a legend.

6 Let $f : \mathbb{N}^\star \to \mathbb{N}$ defined for all $n \in \mathbb{N}^\star$ by :

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ is even,} \\ 3n+1 & \text{otherwise.} \end{cases}$$

The Collatz conjecture says that given any positive integer n, the sequence $(a_n)_n$ defined by $a_0 = n$ and $a_{n+1} = f(a_n)$ will eventually reach the number 1. Check that this conjecture is true for all integers between 1 and 1000.

7 Plot a "large number" of points with random coordinates using the `rand` function. Do they look uniformly distributed on the unit square ? Compute the proportion of points that are below the line $\mathscr{D} : y = x$.

8 Using the `grand` function with the 'uin' distribution, create a game that :

- randomly selects a number between 0 and 100
- asks the user to guess the number (using the `input` function)
- tells the user whether his guess is greater or lesser than the answer and lets him play again until he finds it
- outputs the number of trials when the game is over

9 The middle-square method, suggested by John von Neumann in 1946, is a well-known pseudo-random number generator. Take a number x_0, compute x_0^2 and let x_1 be the digits in the middle of x_0^2. Repeating this procedure gives a list of (pseudo)-random numbers. If you work with 10-digits numbers, then divide each of them by 10^{10} and you will have a list of (pseudo)-random numbers between 0 and 1. In order to complete this exercise, you can use the function below :

```
function [x] = middle(num,k)
    //Extracts k digits from the middle of number num
    format(24)
    s = string(num)
    n = length(s)
    deb = round((n-k)/2)
    v = [deb :deb+k-1]
    x = part(s,v)
endfunction
```

1. Write a `neumann` program following this procedure.
2. Answer exercise 7 with your `neumann` function instead of `rand`.

10 Using a Riemann sum, give an approximation of $\int_0^1 x\,dx$.