Download the file `LAB-DAY2.tgz` into a directory of your choice and unpack it using the tar command.

```
> tar -zxvf LAB-DAY2.tgz
```

It will create a directory `LAB-DAY2` containing three fortran90 files

```
> cd LAB-DAY2
> ls
 harmonic.f90
 numerical_derivative.f90
 numerical_integration.f90
```

NUMERICAL INTEGRATION

The code `numerical_integration.f90` is performing numerical integration by the rectangular rule for `f(x) = cos(x)` between 0 and π/2. Compile the code

```
> ifort numerical_integration.f90 -o num_int.x
```

Run the code and reproduce the following table of integrated values and errors as a function of the number N of sub-intervals used; plot the error vs N on a log-log graph.

2	1.34076	0.34076
4	1.18347	0.18347
8	1.09496	0.09496
16	1.04828	0.04828
32	1.02434	0.02434
64	1.01222	0.01222
128	1.00612	0.00612

256	1.00306	0.00306
512	1.00153	0.00153
1024	1.00077	0.00077

Modify now the code so as to implement the trapezoidal and the Simpson's rules and calculate again the integral with the new code and plot and compare the error of the different schemes.

What happens when N becomes very large ($N \approx 10^6 - 10^8$ or larger) ?

NUMERICAL DIFFERENTIATION

Code `numerical_derivative.f90` implements the numerical derivative of `f(x) = cos(x)` by the asymmetric discretization.
Choose a value of `x`.
Run the code for several discretization step between 0.1 and 10^{-8}.
Plot the error in log-log scale and compare the behavior with the "theoretical" estimate.
Modify the code in order to implement the symmetric difference and the second derivative.
Repeat the error analysis and compare with the "theoretical" estimates.

INTEGRATION OF DIFFERENTIAL EQUATIONS

Code `harmonic.f90` implements the solution of the differential equation for the harmonic oscillator by

1. discretization of the problem on a symmetric interval $[-x_{max}, x_{max}]$ around the origin and exploiting the symmetry to solve it in $[0, x_{max}]$ only.
2. outward integration of the discretized equations in the $[0, x_{max}]$

interval. Unnormalized solution.

3. bisection method for the eigenvalue search.

Run the code with a trial energy which is NOT an eigenvalue and verify that the solution diverges at large value of x

Run the code specifying a trial energy (the GS, the first excited state, ... etc) and verify the relationship between degree of excitation and number of nodes.

Does the solution decay exponentially at large value of x ?

Does the situation improve if you optimize the eigenvalue energy by bisection, or you improve the integration by reducing the discretization ?

Forget about normalization for now and verify how the eigenvalue estimates improve reducing the discretization....

OK but what about normalization ? Why does the solution explodes ?

Modify the code so as to use a stable integration scheme in the classical forbidden region.

Compare with the classical probability for the GS, the first few excited states and an high energy state.

Can you modify the code to implement the Numerov's integration scheme and verify the higher accuracy that can be obtained for a given discretization.