- Ordinary Differential Equations: Differentiating only one variable at a time with respect to some variable (usually time).
- Radioactive Decay:
- $N_U = N_U(0)e^{-t/\tau}$ where tau is the time constant (turns out to be the mean lifetime of a nucleus).
- Taylor Expansion of $N_U(\Delta t) = N_U(0) + \frac{dN_U}{dt}\Delta t + \frac{1}{2}\frac{d^2N_U}{dt^2}(\Delta t)^2 + \ldots$
- $\Rightarrow N_U(t + \Delta t) \approx N_U(t) - \frac{N_U(t)}{\tau}\Delta t$
- The approximation above gives us a basis for the numerical solution for the radioactive decay problem. This approximation is the Euler Method.
- Coding, even in very popular languages like C and Fortran, is much like handwriting, everyone's styles are individualized.
- Naming functions accurately and calling them in order especially if they are part of a subroutine is important. For example, calling initialize, calculate, and store functions only really works in that order if everything is named properly (which it should be).
- Initialize sets the initial number of nuclei and time
- Calculate computes the number of remaining nuclei after the equation is set and used.
- Finally the store function writes the results to a file.
- The process of going through these subroutines can only be completed if everything is written out accurately and variables are properly defined.
- Main question for checking the output of a program: Does the output look reasonable?
- Always check that your program outputs the same result for different step sizes as there should be no true variation of a sizable margin.
- Errors can arise due to the fact that the Euler Method is an approximation technique and does not produce a perfectly accurate numerical result 100% of the time.
- Errors in result are also produced by the finite numerical precision in any program.
- Showing any variations caused by adjustment in the time steps used to analyze along with the correct numerical result for comparison is very beneficial.
- Guidelines to make your program as easily understandable as possible:
    - Program structure: The use of subroutines is very beneficial to the organization of the main program
    - Use Descriptive Names: Choose the names of variables and subroutines according to the specific problem at hand.
    - Use Comment Statements: Explain program logic and variable usage.
    - Sacrifice (almost) everything for clarity: Don't get caught trying to compact your program under the guise that it will make it run more smoothly. It is almost always better to take a few more lines of code to write it out, or a few more variables, than to make the code difficult to understand. If you want to tuneup a program this should only be done after the code has proven to work.