Subroutines & Functions

- Suppose a friend contacted you and wanted to know how to do the trapezoidal rule. You email them the code segment below

```
total = 0
DO  i = 1, n+1
    x = a + (i-1)*h
    f = EXP(-x**2)
    IF( i == 1 .OR. i = n+1) THEN
        total = total + f
    ELSE
        total = total + 2f
    ENDIF
ENDDO
total = total * h/2
```

- They paste this code exactly as written into their program will it work?

- probably not. Why?

  - different variable names — major difference
  - different function — minor difference

- It would be useful to have a mechanism for encapsulating common processes in a way that
  - They can be used over & over without having to rewrite them all the time
  - The details of the process are (mostly) independent from other parts of the program.

- Example subroutine using area of triangle

Rules for Subroutines

- name - a subroutine must have a name
  - naming rules are the same as for any other variable
  - can't be the same as some other variable in the program
- lives "outside" the main program. Begins and ends outside the PROGRAM, END PROGRAM keywords.
- can be in the same .f90 file as the main program or in a separate .f90 file.
- you can pick the subroutine arguements to be in whatever order you want, but you must be consistent in how you call the routine once this is done.
- most common programming errors.
  - Type mismatches (ie, send in integer when subroutine is expecting a double)
  - Array Sizing Errors
- subroutines can call other subroutines (which can call other subroutines, etc.). The routine that calls a particular routine is the <u>calling routine</u>. It may be the main program or another subroutine.