Exercise 01

All `program`s *must* contain `implicit none`. While not strictly necessary, it will eliminate an entire class of bugs.

You should use the lecture materials for help/inspiration, but please don't copy and paste! There is some value to be had in typing up the programs yourself.

Hello World!

1. Open a text file called `hello.f90`. Write a simple "hello world" program that prints a message to the screen. Include at least one comment. Use `gfortran hello.f90` to compile it to `a.out`. Run your program with `./a.out`. Check it does what you think it should.
2. Time to break the program! Delete the `p` in `program` and try to recompile. What happens? What does the error message say?
3. Undo the deletion. How many other single-character deletions that break the program can you find? Which characters don't matter?

Hello <name>!

1. Write up the `hello_input` program into a new file, `hello_input.f90`. Compile, this time using the `-o` flag to give the executable a name. Run the program and give it some input.
2. Try the following ways to break the program. For each method, try to explain why the program behaves the way it does.
    1. Enter two words when it asks your name
    2. Enter a single word longer than 20 characters
    3. Enter a number with a decimal point
3. Create a second `character(len=20)` variable, try reading the two `character` variables with a single `read`, and add your new variable to the `print`

Summing integers

1. Write a program that sums all the integers from 1 to 100
2. Modify the program so that it takes an integer from user input, and then sums all the integers up to that number.

Further

1. What happens if the user supplies a negative number? Hint: try looping over the `read` until the number is acceptable

Solving quadratics

The solutions to the quadratic $ax^2 + bx + c = 0$ is

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

1. Take three real numbers from user input, and print the two solutions for x using the above formula. If there are no real solutions, print a message saying so.
2. Extend your program to also print complex solutions

Factorial

The factorial of n is

$$n! = 1 \cdot 2 \cdot 3 \cdots (n-2) \cdot (n-1) \cdot n.$$

1. Write a function that computes $n!$ by using a `do` loop
2. Write a `recursive` function that computes $n!$ by calling itself with $(n-1)$.

Euler Integration

The Euler update formula for some function $f(y,t)$ is

$$y^{n+1} = y^n + (\Delta t) f(y^n, t^n),$$

where $y^n = y(n\Delta t)$ is an approximation to $y(t)$, Δt is the timestep, and t^n is the current time. Use this formula to solve the ODE

$$f(y,t) = \frac{dy}{dt} = \sin^2(t), y(0) = 0.$$

The exact solution at $t = \pi/2$ is $y = \pi/4$.

1. Write a program that has two functions: one that returns $f(t) = \sin^2(t)$, and one that returns y^{n+1} given y^n, Δt and t^n. The second function should call the first. Use the default `real` for floating point variables.
2. Read an integer N from the user, and then take N timesteps from 0 to $\pi/2$. Compute the error. How does it vary as you increase N?
3. Change the `real` variables to kind `real64`. Now what is the error as you increase N?