Task 1:

A prime number is a positive integer that is only divisible by itself and 1. For example the numbers 7, 13, and 17 are prime. Previously, we wrote a function that will determine if a number is prime, and all prime numbers up to a certain value. We'll do a bit more with this, and organise our code into a module.

- Create a Python file, `primes.py`.
- In it, define a function is_prime(num) that returns True if a number is prime, and False otherwise. Add an appropriate docstring describing what function does.
    - **You should have this function already!**
    - Verify that you can call your `is_prime` function from another file, by importing this module
- Write a function `primes_up_to(n)` that prints all primes up to the argument provided. Make sure it gets a suitable docstring too.
    - **You should have this function, too**
- Write a function, `first_n_primes(n)` that finds and prints the first `n` primes. Note that although the name is similar, this function is doing something very different from the one above – `primes_up_to(1000)` will find all prime numbers less than 1000; `first_n_primes(1000)` finds the first 1000 prime numbers, which will definitely result in numbers greater than 1000.
- Outside the functions add a user interface under `if __name__ == "__main__":`
    - Your user interface should prompt the user whether they want to test the primality of a single number, find all primes less than a target number, or find the first `number` primes.
    - Hint: You can use `input` to prompt the user for which operation to perform, and then use conditional logic to call one of your functions from above.

Task 2:

Create a Python module (`tempConversion.py`) that converts between Fahrenheit, Celsius, and Kelvin temperatures

- Write six conversion functions between temperatures in Celsius, Kelvin, and Fahrenheit: `C2F`, `F2C`, `C2K`, `K2C`, `F2K`, and `K2F`, all in your module `tempConversion.py`.
- Import the module **from another file** make some sample calls on temperature conversions to verify that things are working as expected.
- Add a user interface to the `tempConversion.py` module that requests two separate inputs from the user
    - A temperature as first input
    - The corresponding temperature scale as the second input

- And then display the temperature in the two other scales. For example, entering 21.3 C results in output of 70.3 F 294.4 K.
    - Note that, as above, the user interface should be called under **if __name__ == "__main__":**
    - **Hint:** While it’s *possible* to do this by reading in a single string and pulling it apart, we’ve not learned much about strings yet, so the best bet is to call `input()` *twice* to read in the temperature and scale separately.