Make

- Use Directory Make

  - Demonstrates a very simple makefile for a C program

  Info:

  We have 3 files: main.c, helper.c and helper.h, and a simple makefile

  Try:

  Make the code (just type make)

  Try changing the return value of function1 in helper.c and rerun make. Does the output change correctly?

  Add a new .c file and a recipe to build it. Remember the Tab character. Give it a dependency and check this works

- Use Directory Make_2

  - Demonstrates phony targets

  Info:

  Adds a simple clean step to the previous example

  Try:

  Make the code (just type make) and try the clean (make clean)

  Add the main program (“main”) to the clean step

  Try adding a file called clean and try make & make clean

  make: `clean' is up to date

  This is because clean is not PHONY. Uncomment the line and check it works

- Use Directory Make_3

  - Demonstrates variables

  Info:

Adds a few variables and demonstrates = and :=

Try:

Type make and observe what is printed. How does = and := differ?

Create a variable and print it using info

Create a variable CFLAGS and set it to -g Add this to the compilation recipes so the code builds with debug symbols. Use something like GDB to check.

- Use Directory Make_4

- Demonstrates automatic variables

Info:

Uses $@, $< and $^

Try:

Type 'make test' and observe what is printed. Read through the file and work out the chain of execution

Write some more interesting rules and prereqs to test the use of $@, $< and $^

- Use Directory Make_5

- Demonstrates a real Makefile

Info:

Try:

Adapt the example makefile to a suitable piece of your own code

Alternately, use one of the debugging examples, put each function in a separate file, and write a makefile for it

Bonus: write a very simple script (bash, Python etc) and have Make invoke this during the building