**Make tool**

What is make?

automation tool used for

* Running and compiling programs
* Aliasing commands that span over multiline

Why make is important?

Do I need to install make on my machine?

How does make work?

Make vs makefile

Makefile --> rules 🡪 target: dependencies

Actions

Whitespace sensitive

**Organize your work**

1. Choose a text editor (notepad++, nano)
2. Choose a command-line interpreter (git bash, terminal)
3. Create a folder
   1. cd ~
   2. cd Desktop
   3. mkdir cs330e\_classwork
   4. cd cs330e\_classwork
   5. mkdir make
   6. cd make
4. Git a Unix basic command cheat sheet
5. Install necessary software (Python, Git Bash, MinGW)
6. Create a file, Avg.pt, that contains the following.

def avg(marks):

assert len(marks) != 0

return sum(marks)/len(marks)

print("The average of 1, 2, 3 is: ", avg([1, 2, 3]))

1. On Git Bash, run the following sequence of command.

$ python Avg.py

$ python Avg.py > Output.txt

$ cat Output.txt

1. Use makefile to alias these commands under a target name to make re-run easier.
   1. Create a “makefile” with no extension that contains the following (a make rule with actions but no dependencies).

Output.txt:

python Avg.py

python Avg.py > Output.txt

cat Output.txt

Note: “>” redirect the output to a file (not to monitor). It overrides the contents of an existing file. “cat” lists the contents of a file.

* 1. On Git Bash, run the following command.

$ make Output.txt

Use case for make without dependencies: make searches for a makefile in the current folder, if there is a file then it will search for the target “Output.txt” within the file. If there is no such target, make returns an error; otherwise, make checks in the current directory looking for “Output.txt”, if it’s already exists, then it’ll say “up to date”, otherwise, it will execute the commands under the target “Output.txt”.

Note: running “make Output” will print the commands and the output of the commands. To suppress printing the commands, use “@” directly before each command.

* 1. modify the makefile by adding a dependency.

Output.txt: Avg.py

python Avg.py

python Avg.py > Output.txt

cat Output.txt

Use case for make that contains dependencies: same as the case without dependencies however, the command “make Output.txt” will execute the actions if

* “Output.txt” does not exist in the current folder or
* When one of the dependencies, “Avg.py”, is updated.

Note: “touch <file>” modifies the time stamp of an existing file. If the file does not exist, then an empty file will be created. “echo” is used to print out a message to the user, e.g., echo “Hello World!”.

* 1. If the target that you use is fictional. That is, you need the associated commands to be executed each time you run “make target” (even though the target exists in the current folder). You can declare the target to be phony by making it a prerequisite of the special target .PHONY (see [Special Built-in Target Names](https://www.gnu.org/software/make/manual/html_node/Special-Targets.html#Special-Targets))

.PHONY: target

1. Running “make” without a parameter in a multitarget makefile. The first target from top will only be executed.

Output:

@echo "Output without .txt"

Output.txt: Avg.py

@python Avg.py

@python Avg.py > Output.txt

@cat Output.txt

1. Specifying a default goal in a makefile.

In case you need to type “make” and want a specific target to executed, you can manually set a default target, your\_default\_target, by placing the following at the beginning of the makefile (can be placed after .PHONY).

.DEAFAULT\_GOAL := your\_default\_target

1. When having more than one than one file as the default makefile.

You can create a target (with any name) and place the names of all default targets under that name as dependencies.

all: Output Output.txt

Then, from Git Bash, you can run “make all”