C++

Compile C++

```
g++ -o out /Users/guillaume/Downloads/Perso/Informatique/C++/Test/
main.cpp
```

Execute C++

```
./out
```

Git

Cloner repo

```
git clone git@github.com:johnny-zhong/ImperialQuantLibMentoring.git /
Users/guillaume/Downloads/Perso/Informatique/Mentoring
```

Check for changes and push

```
cd /Users/guillaume/Downloads/Perso/Informatique/Mentoring

git checkout Guillaume-branch

git diff main..your-branch-name

git add file1.txt   # Stage a specific file
git add .           # Stage all changes

git commit -m "some comment"

git push origin your-branch-name:main
```

Create branch

```
git checkout -b branch-name
```

Update Repo

```
cd /Users/guillaume/Downloads/Perso/Informatique/Mentoring

git pull
```

Z Shell

Localiser language

```
which ocaml
```

Afficher/Masquer fichiers cachés

```
defaults write com.apple.finder AppleShowAllFiles YES
defaults write com.apple.finder AppleShowAllFiles NO
```

Convert a file in the Terminal Zsh

```
convert /Users/guillaume/Downloads/Perso/Informatique/OCaml/oui /Users/
guillaume/Downloads/Perso/Informatique/OCaml/oui.png
```

OCaml

Localiser package

```
ocamlfind query package_name
```

Compiler

```
#Graphics
ocamlfind ocamlc -o /Users/guillaume/Downloads/Perso/Informatique/OCaml/
MonteCarlo -I /Users/guillaume/.opam/4.14.0/lib/graphics graphics.cma /
Users/guillaume/Downloads/Perso/Informatique/OCaml/MonteCarlo.ml

#CSV
ocamlfind ocamlc -o /Users/guillaume/Downloads/Perso/Informatique/OCaml/
test -linkpkg -package csv /Users/guillaume/Downloads/Perso/Informatique/
OCaml/test.ml
```

Exécuter

```
ocamlc -o /Users/guillaume/Downloads/Perso/Informatique/OCaml/test /
Users/guillaume/Downloads/Perso/Informatique/OCaml/MonteCarlo.ml
```

Compiler/Executer with dune

Here is a small example of how to use Dune. In the same directory as hello.ml, create a file named dune and put the following in it:

```
(executable
 (name hello))
```

That declares an executable (a program that can be executed) whose main file is hello.ml.
Also create a file named dune-project and put the following in it:

```
(lang dune 3.4)
```

That tells Dune that this project uses Dune version 3.4, which was current at the time this version of the textbook was released. This project file is needed in the root directory of every source tree that you want to compile with Dune. In general, you'll have a dune file in every subdirectory of the source tree but only one dune-project file at the root.
Then run this command from the terminal:

```
dune build hello.exe
```

Dune will create a directory _build and compile our program inside it. That's one benefit of the build system over directly running the compiler: instead of polluting your source directory with a bunch of generated files, they get cleanly created in a separate directory. Inside _build there are many files that get created by Dune. Our executable is buried a couple of levels down:

```
_build/default/hello.exe
```

But Dune provides a shortcut to having to remember and type all of that. To build and execute the program in one step, we can simply run:

```
dune exec ./hello.exe
```

```
Dune build file.exe
_build/default/file.exe
Dune clean
```

OWL for Ocaml

https://www.cl.cam.ac.uk/~lw525/owl/chapter/plot.html