

Practical Go Lessons

Maximilien Andile

July 19, 2022

Contents

1	Programming a computer	19
1.1	What will you learn in this chapter?	19
1.2	Technical concepts covered	19
1.3	Introduction	20
1.4	The four hardware components	20
1.5	Memory	21
1.6	CPU	22
1.7	What is a program?	22
1.8	How to speak to the machine?	23
1.9	Test yourself	25
1.10	Key takeaways	27
2	The Go language	29
2.1	What will you learn in this chapter?	29
2.2	Technical concepts covered	29
2.3	The myth of creation	30
2.4	Motivations	30
2.5	Go key features	31
2.6	The State of Go	31
2.7	Test yourself	32
2.8	Key takeaways	32
3	The terminal	33
3.1	What will you learn in this chapter?	33
3.2	Introduction	34
3.3	Graphical User Interface (GUI)	34
3.4	Command Line Interface (CLI)	34
3.5	How to interact with a shell: the terminal	34
3.6	How to use the terminal	37
3.7	Test yourself	39
3.8	Key takeaways	39
4	Setup your development environment	41
4.1	What will you learn in this chapter?	41
4.2	Technical concepts covered	41
4.3	Introduction	42
4.4	Computer architecture	42
4.5	Go toolchain installation	42

4.6	A tour of Go environment variables	50
4.7	Common errors	51
4.8	Test yourself	52
4.9	Key takeaways	53
5	First Go application	55
5.1	What will you learn in this chapter?	55
5.2	Introduction	55
5.3	Application specification	56
5.4	Project directory	56
5.5	IDE	56
5.6	Source file	56
5.7	Compilation	58
5.8	Test yourself	58
5.9	Exercise	59
5.10	Key takeaways	59
6	Binary and Decimal	61
6.1	What will you learn in this chapter?	61
6.2	Introduction: Numerals, numbers, and quantity	62
6.3	Etymology and symbols	62
6.4	The decimal system	63
6.5	The binary system	65
6.6	Storage capacity	66
6.7	How to store images, videos,... ?	67
6.8	32 vs. 64 bits systems	68
6.9	Test yourself	70
6.10	Key takeaways	71
7	Hexadecimal, octal, ASCII, UTF8, Unicode, Runes	73
7.1	What will you learn in this chapter?	73
7.2	Technical concepts covered	73
7.3	Introduction	74
7.4	Base 16: the hexadecimal representation	74
7.5	Base 8: the octal representation	76
7.6	Data representation bits, nibble, bytes, and words	77
7.7	What about other characters?	78
7.8	Vocabulary	79
7.9	Character sets and encoding	79
7.10	How ASCII works?	80
7.11	How UTF-8 works?	82
7.12	Strings	83
7.13	String literals	83
7.14	Runes	84
7.15	Test yourself	86
7.16	Key takeaways	87

8	Variables constants and builtin types	89
8.1	What will you learn in this chapter?	89
8.2	Technical concepts covered	90
8.3	A variable is a space in memory	90
8.4	Where are variables stored?	90
8.5	Variable identifier (name)	91
8.6	Basic Types	91
8.7	Variable declaration	93
8.8	What is a constant?	96
8.9	Typed constants	96
8.10	Untyped constants	97
8.11	Choosing your identifiers (variable names, constant names)	100
8.12	Practical Application	101
8.13	Test yourself	102
8.14	Key takeaways	103
9	Control statements	105
9.1	What will you learn in this chapter?	105
9.2	Technical concepts covered	105
9.3	A world without control statements	106
9.4	Boolean Expressions	110
9.5	Comparison Operators	111
9.6	Application : boolean expressions & comparison operators	112
9.7	If / else statement	113
9.8	Nested if / else statements	115
9.9	If without else	118
9.10	if / else if / else	120
9.11	Switch case	123
9.12	For statement with a single condition	126
9.13	For statement with a for clause	129
9.14	Practical applications	133
9.15	Test yourself	140
9.16	Key takeaways	141
10	Functions	143
10.1	What will you learn in this chapter?	143
10.2	Technical concepts covered	143
10.3	What is a function?	143
10.4	Why do we need them?	144
10.5	Parameters and results	145
10.6	Number of parameters, number of results	145
10.7	Return statement	145
10.8	A standard function: 2 parameters, one result	146
10.9	Scope of a function	147
10.10	2 parameters, 1 named result	149
10.11	parameter, 1 result	150
10.12	parameter, 0 result	151
10.13	The main function	152
10.14	Practical Application	153
10.15	Challenges solutions	157

10.16	Bits of Advice	159
10.17	Test yourself	159
10.18	Key takeaways	160
11	Packages and imports	161
11.1	What will you learn in this chapter?	161
11.2	Technical concepts covered	162
11.3	Program, package, source files	162
11.4	&Source files	163
11.5	File organization	164
11.6	The main package	165
11.7	The go.mod file	166
11.8	An example program	168
11.9	Practical application 1	169
11.10	Modular programming	170
11.11	Modular programming / Go modules / Go packages	171
11.12	Package naming convention	172
11.13	Linking package together: a trial and error approach.	172
11.14	Import Path & Import declaration	175
11.15	The internal directory	178
11.16	Practical application 2: refactor source code	178
11.17	Test yourself	182
11.18	Key takeaways	183
12	Package initialization	185
12.1	What will you learn in this chapter?	185
12.2	Technical concepts covered	185
12.3	Example: a MySQL driver	185
12.4	Initialization rules	188
12.5	Rules illustration	188
12.6	Can we define several init functions?	191
12.7	Order of variables initialization (advanced)	191
12.8	Test yourself	194
12.9	Key takeaways	195
13	Types	197
13.1	What will you learn in this chapter?	197
13.2	Technical concepts covered	197
13.3	What is a type?	198
13.4	Predeclared types	198
13.5	Composite type	199
13.6	How to create new types: type definition	200
13.7	Type struct variable creation	201
13.8	How to access a field: selector expression	203
13.9	Embedded fields	203
13.10	Usage of embedded fields	204
13.11	Test Yourself	205
13.12	Key Takeaways	206

14 Methods	207
14.1 What will you learn in this chapter?	207
14.2 What is a method?	208
14.3 Methods are capabilities	209
14.4 Method Names	209
14.5 How to call methods	210
14.6 Should I use a pointer receiver or a value receiver?	210
14.7 Receiver type and method call	211
14.8 Methods visibility	212
14.9 Receiver name conventions	214
14.10 Application exercise	214
14.11 Test yourself	218
14.12 Key Takeaways	219
15 Pointer type	221
15.1 What will you learn in this chapter?	221
15.2 Technical concepts covered	221
15.3 What is a pointer?	222
15.4 Pointer types	222
15.5 How to create a variable of pointer type?	223
15.6 Zero value of pointer types	223
15.7 Dereferencing	223
15.8 Maps and channels	225
15.9 Slices	226
15.10 Pointers to structs	229
15.11 Methods with pointer receivers	229
15.12 Test yourself	231
15.13 Key takeaways	232
16 Interfaces	235
16.1 What will you learn in this chapter?	235
16.2 Technical concepts covered	235
16.3 Introduction	236
16.4 A basic definition of an interface	236
16.5 Basic Example	236
16.6 The compiler is watching you!	238
16.7 Example : <code>database/sql/driver.Driver</code>	239
16.8 Interface embedding	239
16.9 Some useful (and famous) interfaces from the standard library	239
16.10 Implicit implementation	241
16.11 PHP and JAVA	241
16.12 The empty interface	242
16.13 Application: Cart storage	244
16.14 Why should you use interfaces?	245
16.15 Bits of Advice	247
16.16 Test yourself	247
16.17 Key takeaways	248

17 Go Modules	249
17.1 What will you learn in this chapter?	249
17.2 Introduction	250
17.3 Dependency	250
17.4 Definition of a Go module	250
17.5 The go.mod file	251
17.6 Vocabulary: API	254
17.7 Vocabulary: Version	254
17.8 Vocabulary: tag, revision, prerelease, release	254
17.9 Semantic Versioning	255
17.10 Dependency Graph	257
17.11 Dependency solving	258
17.12 The problem of version selection	258
17.13 The Go approach: Minimal Version Selection	259
17.14 Vocabulary: checksum	264
17.15 Vocabulary: hash vs encoding vs encryption	265
17.16 The go.sum file	265
17.17 Example	267
17.18 Cleanup task before a release	271
17.19 Where are downloaded versions of modules stored?	272
17.20 Is go.sum a lock file?	273
17.21 Go.sum & go.mod : commit them or not ?	274
17.22 Other commands to know	274
17.23 Test yourself	278
17.24 Key takeaways	279
18 Go Module Proxy	281
18.1 What will you learn in this chapter?	281
18.2 Introduction	281
18.3 What is a proxy server?	282
18.4 The reasons behind the birth of the Go Module Proxy	283
18.5 How does it Work	283
18.6 Configuration of the Go Module Proxy	283
18.7 The four endpoints of a Go Module Proxy (advanced)	284
18.8 Common error: the newest version is not downloaded	285
18.9 Useful links to debug issues	285
18.10 Test yourself	286
19 Unit tests	287
19.1 What will you learn in this chapter?	287
19.2 Introduction	288
19.3 What is unit testing?	288
19.4 What is a test case, a test set, an assertion?	289
19.5 Why Unit testing your code?	289
19.6 Where to put the tests?	290
19.7 How to write a basic unit test	290
19.8 Assertion libraries	293
19.9 How to run unit test	294
19.10 How to write a table test	295
19.11 The two go test modes	297

19.12	Running unit test in parallel	299
19.13	Advanced usage of the go test command	300
19.14	Code Coverage	302
19.15	Test-Driven Development (TDD)	306
19.16	Test yourself	308
19.17	Key Takeaways	309
20	Arrays	311
20.1	What will you learn in this chapter?	311
20.2	Technical concepts covered	311
20.3	Definition	312
20.4	Usage example	312
20.5	Array creation	312
20.6	Zero value	313
20.7	Built-in functions	314
20.8	Access elements of an array	314
20.9	Iterating over an array	315
20.10	How to find an element inside an array with a for loop?	316
20.11	Comparison	316
20.12	How to pass arrays to functions?	316
20.13	How to copy an array?	318
20.14	How to take the address of an array?	318
20.15	Two-dimensional arrays	318
20.16	Multi-dimensional arrays	321
20.17	Efficiency consideration	322
20.18	Limitations	322
20.19	Test yourself	322
20.20	Key Takeaways	323
21	Slices	325
21.1	What will you learn in this chapter?	325
21.2	Technical concepts covered	325
21.3	Definition	326
21.4	Creation of a new slice	326
21.5	Slicing an array, a pointer to an array, or a slice	326
21.6	Do slicing copy data?	327
21.7	Slicing a string	328
21.8	Length	328
21.9	Internal memory representation	328
21.10	Capacity	330
21.11	Length and capacity relation	330
21.12	Slices in function parameters	330
21.13	make built-in function	332
21.14	copy builtin function	332
21.15	append builtin function	333
21.16	How slices grow	334
21.17	Slice elements index	336
21.18	Access an element by its index	336
21.19	Iterate over elements of a slice	337
21.20	Merge two slices	338

21.21 Find an element in a slice	338
21.22 Remove an element at index ¹	338
21.23 Put an element at index ²	339
21.24 Remove all elements of a slice	341
21.25 Prepend an element to a slice	342
21.26 Multidimensional slices	342
21.27 Test Yourself	344
21.28 Key Takeaways	345
22 Maps	347
22.1 What will you learn in this chapter?	347
22.2 Technical concepts covered	347
22.3 Why do we need maps?	348
22.4 What is a map?	348
22.5 Keys: types allowed	350
22.6 Keys must be distinct	350
22.7 Elements	350
22.8 How to create a map	351
22.9 What is a hash table?	351
22.10 Hash table time complexity	353
22.11 Go internals: The hash table implementation	353
22.12 Example usage setup	355
22.13 Initialize and add a key/element pair	357
22.14 Retrieve a value	357
22.15 Delete an entry	359
22.16 Length	359
22.17 Iterate over a map	359
22.18 Two-dimensional maps (map of maps)	360
22.19 Test Yourself	361
22.20 Key Takeaways	363
23 Errors	365
23.1 What will you learn in this chapter?	365
23.2 Technical concepts covered	365
23.3 The term error covers four different concepts	366
23.4 Example: a program that can fail	366
23.5 Error interface as result	369
23.6 How to create standalone errors?	371
23.7 How to wrap an error into another error ?	373
23.8 How to handle errors?	374
23.9 Bits of Advice	378
23.10 Application	380
23.11 Test yourself	392
23.12 Key Takeaways	393

¹Algorithm from : <https://github.com/golang/go/wiki/SliceTricks>

²Algorithm from : <https://github.com/golang/go/wiki/SliceTricks>

24 Anonymous Functions and closures	395
24.1 What will you learn in this chapter?	395
24.2 Technical concepts covered	395
24.3 Anonymous functions	396
24.4 Function types	397
24.5 Functions are “first-class citizens”	397
24.6 Closures	398
24.7 Some common use cases of closures	400
24.8 Test yourself	404
24.9 Key takeaways	405
25 JSON, XML encoding, decoding	407
25.1 What will you learn in this chapter?	407
25.2 Technical concepts covered	407
25.3 Introduction	408
25.4 Encoding / Decoding	408
25.5 Marshal / Unmarshal	408
25.6 What is JSON	409
25.7 Unmarshal JSON	409
25.8 Marshal JSON	411
25.9 Struct tags	413
25.10 Marshal / Unmarshal XML	415
25.11 xml.Name type	417
25.12 Specificity of XML tags	418
25.13 Struct type generators	421
25.14 Custom JSON Marshaler / Unmarshaler (advanced)	421
25.15 Test yourself	423
25.16 Key takeaways	425
26 Basic HTTP Server	427
26.1 What will you learn in this chapter?	427
26.2 Technical concepts covered	427
26.3 What is behind a web page	428
26.4 HTML basics	428
26.5 What is HTTP ?	430
26.6 The journey of an HTTP request	430
26.7 Anatomy of an HTTP request	431
26.8 Anatomy of an HTTP response	432
26.9 Status code of an HTTP response	434
26.10 How to build a simple HTTP Server	435
26.11 How to test a local web server with Chrome	436
26.12 How to test a local web server with cURL?	439
26.13 How to serve an HTML page with a Go Server?	440
26.14 Common error: address already in use	444
26.15 Common error: too many open files	445
26.16 Application: A/B testing server	447
26.17 Test yourself	451
26.18 Key Takeaways	452

27 Enum, Iota and Bitmask	455
27.1 What will you learn in this chapter?	455
27.2 Technical concepts covered	455
27.3 Enum definition	456
27.4 Go and enums	456
27.5 Building a type that can be used “as” enum.	456
27.6 A perfectible solution	457
27.7 Enum libraries	458
27.8 Iota	459
27.9 Byte / Bit	461
27.10 Operations on bits	461
27.11 Bitmasks (advanced)	469
27.12 Test yourself	472
27.13 Key takeaways	473
28 Dates and time	475
28.1 What will you learn in this chapter?	475
28.2 Technical concepts covered	475
28.3 How to get the current time?	476
28.4 Unix Epoch	477
28.5 One time, different clocks	477
28.6 Format the time	478
28.7 How to parse a date/time contained in a string	480
28.8 How to get the time elapsed between two points	480
28.9 time.Duration type	481
28.10 How to check if a time is between two points	481
28.11 How to add days, hours, minutes, seconds to a time	481
28.12 Iterate over time	482
28.13 Test yourself	483
28.14 Key takeaways	484
29 Data storage : files and databases	485
29.1 What will you learn in this chapter?	485
29.2 Technical concepts covered	485
29.3 Introduction	486
29.4 Create a file	486
29.5 File flags	486
29.6 File mode	487
29.7 Write to a file: CSV example	490
29.8 The Example Use Case	492
29.9 MySQL Database	492
29.10 PostgreSQL Database	497
29.11 MongoDB	502
29.12 ElasticSearch	510
29.13 Test yourself	518
29.14 Key takeaways	519

30 Concurrency	521
30.1 What you will learn in this chapter ?	521
30.2 Technical concepts covered	521
30.3 Preliminary definitions	522
30.4 Concurrency pitfalls	523
30.5 Goroutines	526
30.6 Channels	528
30.7 Select statement	536
30.8 Some common use cases of channels and selects	539
30.9 Wait groups	541
30.10 Application : wait group & channels	544
30.11 Mutexes	550
30.12 Concurrent program design	553
30.13 Test yourself	555
30.14 Key Takeaways	557
31 Logging	559
31.1 What will you learn in this chapter?	559
31.2 Technical concepts covered	559
31.3 Introduction	560
31.4 Failure is common	560
31.5 Example logs	560
31.6 Standard log package	560
31.7 Logrus	562
31.8 Logging Format: the Syslog approach	564
31.9 Test yourself	567
31.10 Key Takeaways	567
32 Templates	569
32.1 What will you learn in this chapter?	569
32.2 Technical concepts covered	569
32.3 What is a web page template	570
32.4 Why is it useful	570
32.5 Two template packages	570
32.6 Getting started with templates.	571
32.7 Template actions	572
32.8 Dot notation	573
32.9 Print text	573
32.10 Print dates	575
32.11 Nested templates	575
32.12 Variables	577
32.13 Call a method	578
32.14 Predefined global functions	578
32.15 Custom template functions	580
32.16 Conditionals	581
32.17 Comparison operators	582
32.18 Iteration	583
32.19 Get the value defined at a specific index	584
32.20 Embed template files into the Go Binary	585
32.21 Debugging a template	585

32.22 Test yourself	585
32.23 Key Takeaways	586
33 Application Configuration	589
33.1 What will you learn in this chapter?	589
33.2 Technical concepts covered	589
33.3 Introduction	589
33.4 What is it?	590
33.5 How to do it?	590
33.6 Command-line options	590
33.7 Environment variables	592
33.8 File-based configuration	593
33.9 The <code>github.com/spf13/viper</code> module	594
33.10 Problems and how to avoid them	596
33.11 Security	597
33.12 Test yourself	597
33.13 Key Takeaways	598
34 Benchmark	599
34.1 What will you learn in this chapter?	599
34.2 Technical concepts covered	599
34.3 Introduction	599
34.4 What is a benchmark	600
34.5 How to write a benchmark	600
34.6 How to run benchmarks	601
34.7 Benchmark flags	603
34.8 How to read benchmark results	603
34.9 Detect memory allocations	605
34.10 Benchmark with variable input	606
34.11 Common error: <code>b.N</code> as argument	611
34.12 Test yourself	611
34.13 Key Takeaways	612
35 Build an HTTP Client	615
35.1 What will you learn in this chapter?	615
35.2 Technical concepts covered	615
35.3 Client / Server	616
35.4 What is a REST API	616
35.5 A basic HTTP Client	618
35.6 Request Headers	620
35.7 Real-world example: the Github API	622
35.8 Test yourself	630
35.9 Key Takeaways	631
36 Program Profiling	633
36.1 What will you learn in this chapter?	633
36.2 Technical concepts covered	633
36.3 What is profiling?	634
36.4 Why profiling?	634
36.5 Getting started with profiling	634

36.6 Reading profile files	636
36.7 What is a call stack?	640
36.8 CPU time	641
36.9 What is a sample	644
36.10Pprof command line	646
36.11Display profile in a web browser	647
36.12Interactive web interface	649
36.13What is code optimization?	651
36.14Common optimization techniques	652
36.15Real-world optimization problem	655
36.16Heap profiling	666
36.17Final note	670
36.18Test yourself	670
36.19Key Takeaways	671
37 Context	673
37.1 What will you learn in this chapter?	673
37.2 Technical concepts covered	673
37.3 Introduction	674
37.4 What is Context Oriented Programming?	674
37.5 The “context” package: history and use cases	675
37.6 Linked list	678
37.7 The root context: Background	679
37.8 Add context to your function/methods	680
37.9 Deriving context	680
37.10WithCancel	681
37.11WithTimeout / WithDeadline	681
37.12Example usage	681
37.13WithDeadline	688
37.14Cancellation propagation	690
37.15An important idiom : defer cancel()	694
37.16WithValue	696
37.17Key Takeaways	699
38 Generics	701
38.1 What will you learn in this chapter?	701
38.2 Introduction	702
38.3 What does generic mean?	702
38.4 Why do we need generics?	703
38.5 But we already have the empty interface; why do we need those gener- ics?	703
38.6 Type Parameters: a way to parametrize functions, methods, and types.	704
38.7 What is a type constraint?	705
38.8 Type constraints that you can use right away	707
38.9 How to call a generic function or method?	708
38.10Type inference: enjoy being lazy	708
38.11Types can also have a type parameter list!	709
38.12When is it a good idea to use generics?	710
38.13When it’s not a good idea to use generics?	712
38.14Some generics libraries that you can use in your code	714

38.15 Test yourself	714
38.16 Key Takeaways	716
39 An Object-Oriented language ?	717
39.1 What will you learn in this chapter?	717
39.2 Technical concepts covered	717
39.3 Introduction	718
39.4 What is an object-oriented language	718
39.5 Class, object & instance	718
39.6 Go classes ?	719
39.7 Abstraction	721
39.8 Encapsulation	721
39.9 Polymorphism	722
39.10 Inheritance	723
39.11 Test yourself	727
39.12 Key Takeaways	728
40 Upgrading / Downgrading Go	731
40.1 What will you learn in this chapter?	731
40.2 Technical concepts covered	731
40.3 Checking your version	731
40.4 Where is your Go binary?	732
40.5 Download a new version	732
40.6 Install the (new/old) version	733
41 Design Recommendations	735
41.1 What will you learn in this chapter?	735
41.2 Technical concepts covered	735
41.3 Package names	736
41.4 Use interfaces	737
41.5 Source files	739
41.6 Error Handling	740
41.7 Methods and functions	743
41.8 Key takeaways	753
42 Cheatsheet	755
42.1 Build a program	755
42.2 Slices	755
42.3 Go modules	757
42.4 Goimports	758
42.5 Unit Tests	758
42.6 Naming Conventions	760
42.7 The error interface	760
42.8 Type assertion	761
42.9 Environment variables	763
42.10 Useful links	763
Bibliography	770

Foreword

This book is about Go.

My main objective is to teach you the language in a progressive way. I also tried to clarify and explain some common computer science notions that can be difficult to grasp, especially for newcomers.

I started to write it in 2018 during weekends and nights. At the end of 2020, I decided to quit my job to work on it full-time. Those 2.5 years of writing were intense but rewarding.

I share this work for free on the website (<https://www.practical-go-lessons.com/>). For years I learned so many things from others on the internet. Today, I want to give back to my community!

I want to thank my family for its support, especially my spouse, who brings me joy and happiness every day.

Thank you!

Thank you so much for buying this book! I hope it will help you build great programs.

Feel free to share your progress with me. You can contact me directly via Twitter: <https://twitter.com/MaximilienAld>

Gopher images

The original gopher images are Creative Commons Attribution 3.0 licensed.

The creator of the gopher image is Renee French³

The drawings presented on this book have been created by Binliang Peng⁴.

Feedback

If you spot an error or suggest an improvement, do not hesitate to share it with me on this page: <https://www.practical-go-lessons.com/feedback>

Legal Notice

Copyright (c) 2021 Maximilien Andile

³https://en.wikipedia.org/wiki/Ren%C3%A9_French

⁴<https://www.instagram.com/binliang.alexander.peng/>

The contents of this book and the programs are provided "as is" without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the contents of this book and the programs or the use or other dealings in the contents of this website and the programs.

Trademarks designations that might appear on the pages of this book belong to their owners.

Chapter 1

Programming a computer



1.1 What will you learn in this chapter?

- We will list the different hardware parts of a computer.
- We will see what a program is and how it looks like.
- We will understand how a program is loaded and executed.

1.2 Technical concepts covered

- Memory Unit, Arithmetic and Logic Unit, Input/Output, Control Unit
- Central memory, Auxiliary memory
- Volatile and non-volatile memory
- RAM/ROM
- CPU
- High and low-level languages

- Assembly language, assembler
- Compiled and interpreted language

1.3 Introduction

This book is about Go. Before jumping into our main subject, you need to know some basic knowledge about computers.

Experienced programmers can skip this chapter. Beginners should take some time to study it.

Your programs will run on hardware. Knowing how your hardware is working may improve the design of your programs.

Firstly we will describe the main components of a computer. Then we will see what a program is and how the machine handles it.

1.4 The four hardware components

A computer is composed of **four** main parts :

- **The memory unit (MU)** where data and programs are stored.
 - For instance, we can store into the memory unit the grades of a college class. We can also hold a program that will compute the class's average grade.
- **The arithmetic and logic unit (ALU)**. Its role is to perform arithmetic and logical operations on data stored into the memory unit. This unit can perform, for instance, additions, incrementations, decrementations, which are called **operations**. In general, each operation requires two operands and output a result.
 - Let's say we want to add 5 and 4. Those two numbers are the operands. The result of this operation is 9. Operands are loaded from the memory unit. The ALU is an electrical circuitry that is designed to execute operations.
- **The input and output unit (I/OU)** will be in charge of loading data into the memory unit from an **input** device. This unit also sends data from the memory unit to an **output** device.
 - An input device is, for example, the touchpad of your computer, your keyboard, your mouse.
 - An output device is, for instance, your monitor.
- **The control unit (CU)** will receive instructions from programs and will control the activity of the other units.

The four hardware components represent a **schematic view** of the computer's components.