

UNIT 3 • SESSION 3

Software — and your site goes live

Today you write a small Python program, and it answers to an address on the public internet.

2ND SEMESTER • SCHOOL YEAR 2026-2027

Introduction to Computing • Prof. Ryan V. Labrador

Saturday, 29 August 2026 • 2-hour lecture + 3-hour lab

Two hours, three parts — then the event

Part 1

Software

System versus application, and what an operating system really does

Part 2

Licensing

Proprietary, open source, freeware, SaaS — and piracy, honestly

Part 3

Going live

Apache versus a Python program, why Flask exists, and your first index.py

After

Laboratory

Ten groups. Ten public websites. Today.

By the end of today you will be able to...

1

Distinguish system software from application software and give examples of each.

2

Describe the four jobs an operating system does: processes, memory, files, devices.

3

Compare Windows, macOS, Linux, Android and iOS on what they are actually for.

4

Name the main software licence types and say what each one permits you to do.

5

Explain, in Philippine legal and practical terms, why pirated software is a bad deal.

6

Explain what a web server does and trace how your file reaches somebody else's browser.

7

Connect to a real Linux server over SFTP and find your group's app folder.

8

Say why a Python program is needed to serve a page, and what had to be installed for it to work.

PART 1

Software

The instructions that make all that metal do something.

Three layers, and each only talks to its neighbours

3

Applications

What you actually open. Browser, Word, games, GCash. They ask the operating system for everything and never touch the hardware directly.

2

Operating system

The layer in between. It turns 'save this file' into thousands of hardware instructions, and it stops programs from fighting over the same parts.

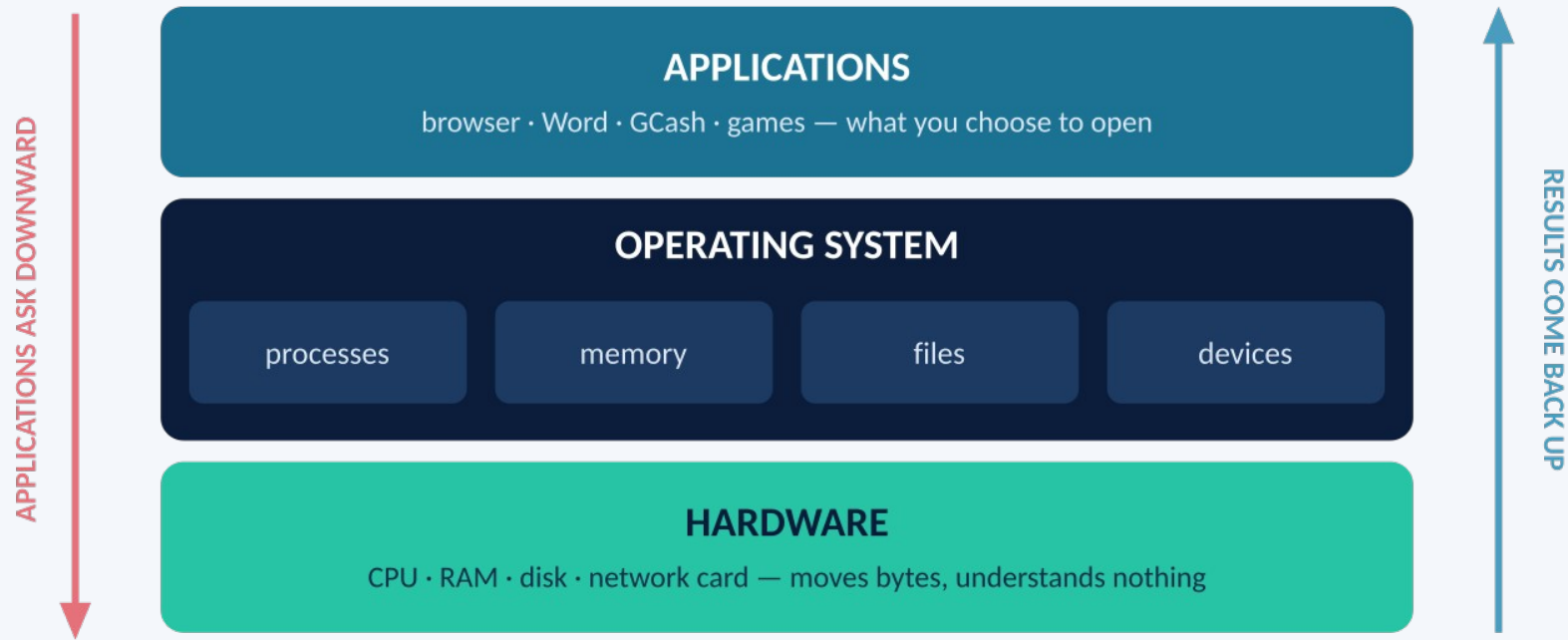
1

Hardware

The metal from last week. It can do arithmetic and move bytes. It has no idea what a file, a window or a photograph is.

An application cannot reach the disk on its own. It has to ask. This is the whole reason operating systems exist.

The operating system is the layer in between



An application cannot touch the disk. It has to ask. That is why operating systems exist.

The four grey boxes inside the middle layer are the operating system's four jobs. We open each one next.

System software and application software

System software

- Runs the machine, not your task
- Operating systems: Windows, Android, Linux
- Device drivers — how the OS learns a new device
- Firmware, and utilities like antivirus and backup
- You rarely open it on purpose; you notice it when it fails

Application software

- Runs your task, not the machine
- Word processors, spreadsheets, browsers
- Messaging, banking, games, editing tools
- Written to sit on top of an operating system
- This is what people mean when they say 'an app'

Rule of thumb: if you chose to open it, it is application software.

What an operating system actually does



Process management

Decides which program runs on which core, and for how long. This is why your phone can play music while you type a message.



Memory management

Gives every program its own space in RAM and stops one program reading another's. It is also why one app crashing does not take the machine with it.



File management

Folders, filenames, permissions. It tracks where every byte of your file physically sits on the disk, so you never have to.



Device management

Talks to the printer, the Wi-Fi card, the keyboard. A driver is how the operating system learns a device it has never met before.

What happens between the button and the desktop

Step 1

Firmware wakes

The chip on the motherboard checks the hardware is present. This is ROM doing its job.

Step 2

Bootloader runs

It finds the operating system on the drive and loads the first part of it into RAM.

Step 3

The kernel starts

The core of the OS takes control of the CPU, the memory and the devices.

Step 4

You see a desktop

The interface loads last. By the time you see it, most of the work is finished.

Every one of these steps is why a computer takes time to start. It is not being lazy.

Operating systems you will meet



Windows

Most desktops here, most offices, most computer labs. The widest software library.



macOS

Apple laptops and desktops. Unix underneath, which makes it popular with developers.



Linux

Free and open source. Runs most web servers on earth — including the one your site goes on.



Android

Linux underneath. The phone in the majority of pockets in this room right now.



iOS

iPhones and iPads. Tightly controlled, which is both its strength and its complaint.



ChromeOS

A browser that boots. Cheap school laptops. Almost everything happens online.

Card 3 matters today: your website will be served by a Linux machine this afternoon.

Utility programs



Antivirus

Watches for malicious files and behaviour. Windows ships with one already.



Backup

Copies your work somewhere else. The only utility that has ever saved a thesis.



Compression

ZIP and RAR. Makes files smaller for sending; we cover how in Unit 6.



Disk cleanup

Removes temporary files. A full drive slows a machine to a crawl.



Update manager

Keeps the OS and drivers current. Most updates close security holes.



Task manager

Shows what is running and what is eating the RAM. You used it last week.

Application software by category



Productivity

Documents, spreadsheets, slides.
Word, Excel, LibreOffice, Google Docs.



Communication

Messenger, Viber, email, Zoom, Google Meet. Most-used category in the country.



Creative

Photo, video, audio, design. Photoshop, GIMP, Canva, CapCut.



Business systems

Payroll, inventory, enrolment, hotel booking. What BSIS graduates get hired to build.



Entertainment

Games, streaming, music. The category that pushes hardware hardest.



Financial

GCash, Maya, online banking. Software your family already trusts with money.

Card 4 is your degree. Somebody wrote the enrolment system you used in June.

PART 2

Licensing

Software is almost never bought. It is licensed — under conditions.

How software is licensed

Proprietary

You buy permission

Windows, Microsoft 365, Adobe. Source stays closed; the licence limits machines and users.

Open source

You may read and change it

Linux, LibreOffice, VLC, Firefox. Free, and the licence often requires sharing your changes.

Freeware

Free, but still closed

Free to use, yet you cannot see or modify the code. Free is not the same as open.

Shareware

Free for a while

Full features for a trial period, then it stops and asks for payment.

SaaS

You rent access

Google Workspace, Canva Pro, Netflix. Nothing installed; stop paying and access ends.

Pirated software in the Philippines

What gets said

- "Everybody here uses a cracked copy."
- "The real one costs a month's allowance."
- "It works fine. Nothing happened to me."
- "The shop installed it, not me."
- "Nobody is checking anyway."

What is actually true

- It is copyright infringement under RA 8293, the IP Code
- The cost is real — and legal free alternatives exist for nearly everything
- Cracked installers are a well-known route for malware onto home PCs
- No security updates: the machine stays permanently unpatched
- Employers are audited. Graduates have lost jobs over this.

This is not a lecture about morality. It is about risk, updates, and your future employer.

Legal software that costs nothing



LibreOffice

Documents, spreadsheets, slides.
Opens and saves Word and Excel files.
Open source.



Google Docs

Runs in a browser, nothing to install.
Ideal on a shared laboratory PC.



GIMP

Photo editing. Harder to learn than
Photoshop, genuinely capable, free.



Canva free tier

Posters, decks, social graphics. The free
tier covers coursework comfortably.



VS Code

The code editor professionals actually
use. Free. You will meet it in Unit 12.



Linux

A complete operating system. Free —
and it will serve your website this
afternoon.

Every tool on this slide is legal, free, and enough for everything this course asks of you.

PART 3

Going live

For the next hour, this stops being a class about computing.

What a web server actually is



A computer that never sleeps

Same parts you named last Saturday: CPU, RAM, storage — running Linux. Ours sits at 58.71.69.30 and it does not get switched off.



A machine running programs that answer

Something on it has to be awake and listening. Apache is. From today, so is a small program of yours.



A machine that answers to a name

g01.cic.rmlsystems.net points at that same computer. How a name becomes a machine is Session 9 — today you just use it.

There is no cloud. There is somebody else's computer, in a building, with a power bill — and programs running on it.

How a request becomes a page



Five steps. You are responsible for two of them. The internet does the other three, every single time.

An HTML page, complete — and written by hand

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Group Turing</title>
</head>
<body>
  <h1>Group Turing</h1>
  <p>BS Information Systems, City of Ilagan College</p>
</body>
</html>
```

Eleven lines, and a complete web page. Hold on to the shape of it — in four slides the same page comes back, with a program around it.

What you just looked at

DOCTYPE

The declaration

Tells the browser this is a modern HTML page. Always the very first line.

html lang

The wrapper

Everything lives inside it. lang says which language, which helps search and screen readers.

head

About the page

Title and settings. Nothing inside head appears on the page itself.

meta charset

The alphabet

Makes ñ, é and emoji display properly. Leave it out and Filipino names break.

body

Everything visible

Headings, paragraphs, links, images. This is where you will live for fifteen weeks.

One page is easy. Ten are not.

Write it as a file

- The words Group 1 are typed into the file, three times
- Group 7 needs its own copy, edited in both places
- Ten groups, ten files, thirty edits
- Change the college name and you do it ten times
- The page can never say anything it did not already say

Let a program build it

- The group number is written ONCE, at the top
- The rest of the page reads it from there
- Ten groups, one file, ten one-word edits
- Change the college name once and all ten follow
- The page can work something out before it answers

Today the difference looks small. In three weeks your page has to convert a number somebody just typed — and no file can be written in advance for that.

Apache serving a file, and a Python program

	APACHE, SERVING A FILE	YOUR PYTHON PROGRAM
What it is	Application software	Application software
Who wrote it	Somebody else, in 1995	You, this afternoon
When asked for a page	Finds the file, sends a copy	Runs your function, builds a reply
Can the page change?	Only if you upload a new file	Every request, if you want
Is anything of yours running?	No	Yes — and it can stop
Extra software needed	None. It is already there.	Python 3, pip, Flask

Both are application software, running on the same Linux, asking the same kernel for the same disk. That is this morning's lesson, standing in front of you.

Written as a file, and written as a program

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Welcome Group 1</title>
</head>
<body>
  <h1>Welcome, Group 1</h1>
  <p>Introduction to Computing</p>
</body>
</html>
```

index.py

```
from flask import Flask, render_template_string
app = Flask(__name__)

INFO = {"group": "1"}

PAGE = """<!DOCTYPE html>
<html lang="en">
<head>
  <title>Welcome Group {{ i.group }}</title>
</head>
<body>
  <h1>Welcome, Group {{ i.group }}</h1>
  <p>Introduction to Computing</p>
</body>
</html>"""

@app.route("/")
def welcome():
    return render_template_string(PAGE, i=INFO)
```

Identical pages. On the left, Group 1 is typed three times and is there forever. On the right it is typed once, at the top, and the page asks for it.

Why you cannot just write Python and be done

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 214
Connection: close
```

A browser does not ask for a page. It opens a connection, sends a request in a format agreed in 1991, and expects those four lines back before your HTML.

Without Flask

You write all of that

Open a socket, read the request, work out what was asked for, write the status line and the headers yourself. Hundreds of lines before you print a single word of your own.

With Flask

You write the last part only

One line saying which address you answer, one function saying what to send. Flask does every other part of that conversation for you.

What has to be installed before any of this works



Already there **Linux**

The operating system. It owns the hardware and hands it out. Nothing else on this list runs without it.



Already there **Apache**

Takes every request that arrives on port 80 and decides where it goes. From today it passes yours inward, to your program.



Installed **Python 3**

The interpreter — the program that reads your index.py and does what it says. Without it your file is just text.



Installed **pip**

Python's installer. It fetches libraries other people wrote. It comes with Python 3.



Installed **Flask**

pip install flask. One command. This is the library that speaks HTTP so you do not have to.



Configured **The proxy line**

One line of Apache config sending g01.cic.rmlsystems.net to your program on port 8001. Already done for you.

Six things. Two were already running, three were installed with one command each, and one is a line of configuration. That is the whole stack under your website.

The top half of index.py

Line 1**from flask import ...**

Bring in two things Flask gives us. Flask itself, and a function that turns a page of HTML into a finished reply.

Line 2**app = Flask(__name__)**

Make the application and call it app. Every Flask program on earth starts with this line. `__name__` tells Flask where it lives so it can find your static folder.

INFO**A dictionary**

A labelled box of facts. `INFO["group"]` is "1". This is the ONLY part your group edits, and everything else on the page reads from it.

PORT**8000 + your group**

Computed from the group number above, so you never type it twice. Group 7 gets 8007. Two groups on one port is the one error that stops both.

PAGE**Triple quotes**

Ordinary HTML, kept in a Python variable. Three quote marks let a string run over many lines. Inside it, `{{ i.group }}` is a hole the group number drops into.

The bottom half — the part that answers

```
@app.route("/")
def welcome():
    return render_template_string(PAGE, i=INFO)

if __name__ == "__main__":
    app.run(host="127.0.0.1", port=PORT, debug=False, use_reloader=True)
```

@app.route**The address**

When somebody asks for / — your homepage — run the function underneath. Change this line and the page moves.

def welcome**The function**

A name for the work. Everything indented under it belongs to it. The indent is not decoration; it IS the grammar.

return ...**The answer**

Fill the holes in PAGE with the facts in INFO, and send the result back. This is the line that builds the page.

app.run**Switch it on**

127.0.0.1 means only Apache can reach it. debug=False stays False. use_reloader means saving the file restarts it for you.

The one line that makes it yours

```
"group": "1",
```



Change it to your number

Group 7 types 7. The title, the heading, the eyebrow line and your port all change with it.



Then the motto one sentence

The only writing on the page that is yours rather than the college's. Make it worth reading.



Then upload one file

index.py goes into the app folder. Refresh. Your site is different from the other nine.

One digit and one sentence between the page you are looking at and a page that is yours.

YOURS

From this afternoon, this is your address

g01.cic.rmlsystems.net

g01

Turing

g02

Lovelace

g03

Hopper

g04

Babbage

g05

Berners-Lee

g06

Hamilton

g07

Ritchie

g08

Noyce

g09

Torvalds

g10

Banatao

Ten addresses. They did not exist this morning. By tonight they will answer.

Your address already answers

Right now, before the lab

- Open `g01.cic.rmlsystems.net` on your phone
- A page loads. A plain default page.
- Nobody in your group wrote a word of it
- It has been on the public internet since this morning
- Screenshot it. This is the only 'before' you get.

By the time you go home

- Same address. Same server. Same folder.
- One file replaced: `app/index.py`
- Your group's number, your motto, your words
- Sent to a parent, a cousin, a friend in another province
- Nothing else about the internet changed.

Publishing is not creating a website. It is replacing one file inside a program that is already running.

How you get your program onto that machine

Protocol	SFTP	Secure File Transfer Protocol. Copies files to a server over an encrypted connection.
Host	58.71.69.30	The machine itself. In FileZilla you type it as sftp://58.71.69.30 — with the prefix.
Port	22	SFTP and SSH both live on port 22. Plain FTP uses 21, and plain FTP is not what we are doing.
Username	g01 ... g10	One account per group. Your group number is your username. Nothing to create.
Password	On your credential slip	Handed to your Group Leader on paper, in the lab. It is not on this screen and it will not be.
Folder	app	Not public_html. Drop index.py in there, refresh the page, and the change is live in about two seconds
Forget the sftp:// prefix and FileZilla tries plain FTP on port 21 and fails. That is the most common mistake in this lab.		

Where your file has to land

app / index.py

app

Where the program lives

Not on the internet by itself. Nothing in here is served because it is in here — it is served because your program is running and chose to send it.

index.py

The program

The file Python reads. Replace this one file and your whole site changes. It is already there and it is already running.

static

Pictures and styles

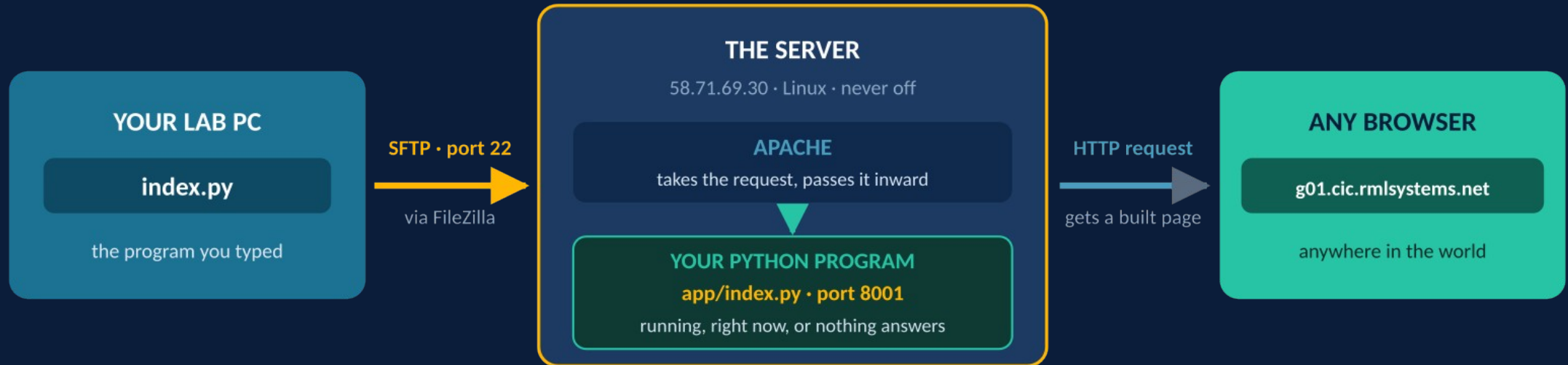
The seal, and anything else the page loads rather than says. Flask serves this folder for you, at /static/.

public_html

Not used any more

It is still on the server and you can leave it alone. From today nothing in it is served. Your address goes to your program instead.

From the machine in front of you to the world



You upload the PROGRAM. The program builds a fresh page every time somebody asks — nothing is sitting there ready.

One file. One folder. One address. Everything else this afternoon is detail.

What 'live' actually means

Public

Anyone can open it

Classmates, family, a stranger in Cebu, a recruiter reading your CV in three years.

Durable

Treat it as permanent

Assume anything you publish can be screenshotted before you think to delete it.

Named

Your group's name is on it

Ten sites, ten reputations. Nobody publishing here is anonymous.

Yours

You are responsible for it

No images you did not make or licence. No classmate's personal data without consent.

We come back to that last line properly in Unit 10, with the Data Privacy Act.

Four things from today



One

Software has layers

Hardware, operating system, applications. Each layer only talks to its neighbours.



Two

The OS is a manager

Processes, memory, files, devices. It decides who gets the hardware, and when.



Three

Licence, not ownership

You almost never own software. You own permission, under stated conditions.



Four

A program plus an address

A website is a program that answers, an address it answers to, and something keeping it running.

If you remember one thing: the four jobs of an operating system.

Ten groups. Ten websites. Today.



First

Credential slips

Group Leaders collect and sign for their group's slip. One slip per group, on paper.



Then

Connect

SFTP to 58.71.69.30 on port 22, into the app folder. Replace index.py. Reload the address.



AI task

Ask AI for a page

Then find two things in it you never asked for, and explain what they do.



Plan B

It is already written

If the network fails, we save to USB and publish from one machine. Nobody leaves unpublished.

Group leaders: bring last week's USB stick to the front now — and stay behind to sign for your slip.

Ten websites that did not exist this morning.

Before you go home today, ten addresses on the public internet will belong to this class. In six weeks you will trace exactly how they work. Keep the address — you will put it on a CV one day.

Let's go.