Lab 1 · Part 1 Foundations of Computation AI Level 0

Setting Up and Running Your First Program

Install Python and VS Code, learn to run a .py file from the terminal, and submit once. Part B shows you chaos in ten lines — shift the starting value by one millionth and twenty-five rounds later nothing looks the same.

Due: Homework 1 · Sat 10 Oct, 23:59

What this session is for

- Get Python 3.13 and VS Code installed on your own machine
- Run a program two ways: the Run button in VS Code, and a command in the terminal
- Go through the submission process once
- See for yourself how a tiny difference in a starting value gets amplified
- Meet a handful of basic terminal commands
- Walk through a complete submission once

AI policy

This week is **Level 0**: no AI-generated or AI-completed code. If something goes wrong during installation you may ask an AI what an error message means, but the code you submit has to be yours.

Installation goes wrong, and that is normal

With a hundred-odd people on every conceivable laptop and OS version, some setups will fight back. It is not a reflection on you. TAs are in the room the whole session — put your hand up. **Do not spend an hour alone wrestling with it.**

How each lab session is arranged

Problems come in three parts, **all open at once** — you do not have to finish one before starting another:

- **Part A** (required) — covers the week's material, graded automatically
- **Part B** (deeper) — one larger problem in several escalating steps
- **Part C** (fundamentals) — command line, git, remote access; a short segment every week, and **examinable**

Installation will eat most of today, so all three parts are light this week. Every later session follows the same shape.

1. Install Python

The course site hosts a copy of the official **Python 3.13.15** installers. The campus network reaches python.org at unpredictable speeds, and the first ninety minutes are too short to spend waiting, so **download them from here**:

Your machine	Download	Size
Windows (almost certainly this one)	python-3.13.15-amd64.exe	28 MB
Windows on ARM (Snapdragon laptops)	python-3.13.15-arm64.exe	27 MB
macOS (Intel and Apple Silicon alike)	python-3.13.15-macos11.pkg	69 MB

These are byte-for-byte the files from python.org/downloads, kept locally.

Not the Microsoft Store version of Python

The Store build handles paths and permissions differently from the official installer, and it produces baffling failures later when you start installing libraries. Every year somebody loses an afternoon to it. **Use the installer above.**

On Windows, also **install to the default location on C:** — not D:, and not a path with non-ASCII characters in it.

Windows

Run the installer and **tick `Add python.exe to PATH` on the very first screen**, then click Install Now.

Missing this checkbox causes a lot of grief

The `Add to PATH` box sits at the bottom of the window and is unticked by default, so it is easy to miss. Without it, typing `python` in a terminal gives you "not recognized as an internal or external command." The fix: run the installer again, choose Modify, and tick it.

macOS

Download the `.pkg` installer and click through it.

The `python3` that ships with macOS is there for the system's own use and is often an older release. Once you have installed from python.org, use that one.

On Linux there is nothing to install

Ubuntu and friends ship with `python3`; run `python3 --version` and see whether it is recent enough. Installing a newer one is `sudo apt install python3.13`.

Do not casually replace the system's default python3, though. A great deal of Linux plumbing is itself written in Python, and swapping the interpreter out from under it produces a cascade of baffling failures.

Check it worked

Open a terminal (PowerShell or Terminal on Windows, Terminal on macOS) and type:

```
python --version
```

You should see `Python 3.13.x`.

If the command is not found, try `python3 --version` on macOS and Linux. On Windows it means PATH was not set — reinstall with the checkbox ticked.

Why there are two commands, `python` and `python3`

Python 2 and Python 3 are incompatible, and the two coexisted for many years. Throughout that period the convention was that `python` meant 2.x and `python3` meant 3.x. Python 2 reached end of life in 2020, but the convention outlived it: **macOS and Linux usually offer only `python3`**, while the official Windows installer provides both names. These notes always write `python`; the check above tells you which one your machine has.

2. Install VS Code

Download it from code.visualstudio.com.

Then add one extension:

1. Click the Extensions icon in the left bar (the four squares), or press `Ctrl+Shift+X` (`Cmd+Shift+X` on macOS)
2. Search for `Python`
3. Install the official one — publisher Microsoft, by far the most downloaded

It provides syntax highlighting, error checking, the Run button and the debugger.

3. Write your first program

1. Make a folder somewhere for this course's code. **Keep the path free of spaces and non-ASCII characters** — some tools handle them badly. Something like `D:\cs1602` or `~/cs1602`.
2. In VS Code, File → Open Folder, and open it.
3. Create a file called `hello.py`.
4. Put this in it:

```
print("Hello world")
```

5. Click the ▶ Run button in the top right. A terminal panel opens below and shows `Hello world`.

Now run it from the terminal

The Run button is convenient, but you should know what it does for you.

Press `Ctrl+`` (the backtick, below Esc) to open a terminal inside VS Code, and type:

```
python hello.py
```

Same output. **This is what running a Python program actually is** — the Run button just types that line on your behalf.

Why bother with the terminal

Sooner or later you will run code on a server with no graphical interface, and the terminal is all there is. Plenty of tools — installing libraries, managing versions, running tests — are command-line tools to begin with. This course does not go deep, but Lab 8 works through the common Linux commands properly.

4. Five small experiments

With Python installed, work through the five things below in the **interactive interpreter**. Type `python` (or `python3`) in a terminal to get in — the prompt is `>>>` — and `exit()` or `Ctrl+D` to get out.

Your answers go on the "Setup check and first observations" answer sheet on the platform. **Each is a single line to type; the point is not the typing, it is whether the result matches what you expected.**

1. **Two kinds of quote.** Run `print("SJTU")` and then `print('SJTU')`, and compare.
2. **Python's own creed.** Run `import this` and read it. These nineteen lines by Tim Peters are the community's statement of taste, and this course keeps coming back to them.
3. **A very large number.** Evaluate `2 ** 100`.
4. **Two kinds of division.** Evaluate `10 / 3` and `10 // 3`.
5. **One addition.** Evaluate `0.1 + 0.2`.

The fifth one is a thread running through this course

`0.1 + 0.2` is not `0.3`. That is not a bug in Python and there is nothing wrong with your machine — every language using IEEE 754 floating point (C, Java, JavaScript, Excel) agrees. **L2 explains why**, and Part B below is the same phenomenon seen from another angle.

For fun (not marked)

Put this in `star.py` and run it:

```
import turtle

tur = turtle.Turtle()
tur.speed(6)
tur.getscreen().bgcolor("black")
tur.color("cyan")
tur.penup()
tur.goto(-200, 50)
tur.pendown()

def star(t, size):
    if size <= 10:
        return
    for _ in range(5):
        t.forward(size)
        star(t, size / 3)
        t.left(216)

star(tur, 360)
turtle.done()
```

A five-pointed star, with smaller stars growing out of each edge, and smaller ones out of those — a **fractal**. Under twenty lines of code draw it, and the trick is that `star` calls itself. That is **recursion**, and L7 is devoted to it.

You are not expected to follow it yet. Change `360` and `size / 3` and watch what happens.

The drawing window seems to hang

`turtle.done()` waits for you to close the window, so the terminal sits there doing nothing. That is not a crash. Close the drawing window and the terminal comes back.

On macOS, `ModuleNotFoundError: No module named 'tkinter'` means your Python was built without the GUI library. Skip this one; nothing else depends on it.

5. Submitting

Open the grading platform and **register with your student number**. There is no access code — if your number is on the course roster, you can register. Then open Homework 1.

That page has two kinds of entry:

Entry	What you hand in	How it is marked
Code problem	one function	the platform runs the tests and scores it immediately
Answer sheet	multiple choice and written answers	the choices are marked immediately; the written answers go into the grading report for a TA

Part A's programming exercises are code problems. This week's five experiments and the questions in Parts B and C are answer sheets — **you answer them on the page itself; there is no separate document to hand in**. Everything you write is folded into the grading report, which you download and submit to Canvas at the end of term.

Start a habit this week: submit something before every lab session ends, even if it is incomplete. The platform keeps your best result, and answer sheets can be revised and resubmitted as often as you like.

Part A — required

1-1 Introduce yourself

Write a program that uses `print()` to output three lines: your name, your student number, and one sentence of your choosing.

```
Zhang San
525030910001
Hoping to write code I'm actually happy with this term
```

1-2 Change of base

Use `bin()`, `oct()` and `hex()` to show your student number in base 2, 8 and 16.

> **提示**
>
> A student number is a string of digits — but `bin("525030910001")` raises an error, because what is inside quotes is text, not a number. Work out what to write instead.

1-3 Predict, then run

Print the results of the following. **Guess each one before you run it**, and see how you did:

```
7 / 2
7 // 2
7 % 2
2 ** 10
```

Write them down first, then run and check. **Whichever one you got wrong goes on the "Part A · Arithmetic and reading errors" answer sheet**, along with what you had expected. We cover all four operators properly in L2.

1-4 Break it on purpose

Produce each of the three errors below and paste the complete message onto the answer sheet:

1. Misspell `print` as `pirnt`
2. Use full-width parentheses: `print ("hello")`

3. Leave a quote unclosed: `print("hello)`

Then answer: what is the **first line** of each error? Does the line number it points at match where the mistake actually is?

> **This problem is not filler**
>
> Over the next fifteen weeks you will see error messages hundreds of times. **How quickly you can read them directly determines how fast you write code.** From today, the reaction to red text should be to read it, not to panic.

Part B — going deeper: chaos in ten lines

You are not expected to understand the syntax yet. Type it in, run it, watch what happens. What this shows you is a property of **computation itself**, not of Python.

Create `chaos.py`:

```
x = float(input("Enter a number between 0 and 1: "))

for i in range(25):
    x = 3.9 * x * (1 - x)
    print(f"{i + 1:3}  {x:.17f}")
```

The last two lines need four extra spaces of indentation — that is how Python marks "these lines belong to the loop". L4 covers it properly.

B-1 Run it

Enter `0.25` and run it. Note the last line — the next question compares against it.

That formula is the logistic map, a model from population biology: `x` is the current population as a fraction of what the environment supports, and each round works out the

next generation. The 3.9 is the growth rate.

B-2 Nudge the starting value by one millionth

Run it again with `0.250001`.

The two starting values differ by **0.000001**. Comparing the two runs:

1. How far apart are the two values at round 10?
2. How far apart at round 25?
3. From roughly which round can you no longer see any relationship between the two sequences?

> **This is the butterfly effect**
>
> In 1961 the meteorologist Edward Lorenz restarted a weather simulation from a mid-run value, typing 0.506 instead of the 0.506127 the machine had stored. He assumed a difference that small could not matter. The two forecasts diverged completely, and chaos theory came out of the surprise.
>
> The practical consequence: **long-range weather forecasting is impossible in principle**, not because computers are too slow, but because you can never measure the starting conditions precisely enough.

B-3 Same starting value, two ways of writing the same formula

Replace the line inside the loop with this, and enter `0.25` again:

```
x = 3.9 * x - 3.9 * x * x
```

`3.9 * x * (1 - x)` and `3.9 * x - 3.9 * x * x` are **algebraically identical** — expand the brackets and check.

Answer:

1. Do the two versions give the **same** value at round 25?
2. From which round does a difference appear?
3. Why do you think that is? (A guess is fine — the answer is in **L2, part four**.)

B-3 is the one worth thinking about

In B-2 the inputs differed, so differing outputs are at least explicable. In B-3 the input is identical and the formula is mathematically identical, yet the results differ — **which tells you the machine is not computing the mathematical expression you wrote.** The second lecture explains why.

Part C — fundamentals: your first terminal session

Why this section exists

This course is an **introduction to computation**, not just to Python. Every lab from here on carries a short fundamentals segment: the command line, version control, remote access, permissions, shell scripts. Fifteen minutes a week adds up to a working set of developer skills by the end of term. **It is examinable.**

Sooner or later you will run programs on a server with no graphical interface, where the terminal is all there is. Plenty of tools — installing libraries, version control, running tests — are command-line programs to begin with.

Four commands

Command	What it does	Mnemonic
`pwd`	which directory am I in	**p**rint **w**orking **d**irectory
`ls`	list what is in this directory	**l**i**s**t
`cd name`	move into a directory	**c**hange **d**irectory
`cd ..`	go up one level	`..` means the parent

All four work in Windows PowerShell (`ls` and `pwd` are aliases there).

> **Press Tab**
>
> Type `cd cs` and press Tab; the shell completes it to `cd cs1602`. **This is the single most valuable habit at a command line** — it saves time and it cannot misspell.

C-1 Walk through it

Do these in order and copy **each command and its output** straight out of the terminal onto the answer sheet (no screenshot needed):

1. `pwd` — see where you are
2. `cd` to the folder holding your course code
3. `ls` — check that `hello.py` and `chaos.py` are both there
4. `python hello.py` — run it
5. `cd ..` to go up a level, then `pwd` again to confirm you moved

“Can’t open file” is almost always a path problem

If `python hello.py` reports `can't open file`, you are nearly always in the wrong directory. Run `pwd` to see where you are, then `ls` to check whether the file is there.

Checklist

Before you submit, go through this:

- [ ] `python --version` prints 3.13.x
- [ ] the Python extension is installed in VS Code
- [ ] the Run button works on `hello.py`
- [ ] `python hello.py` works on the same file
- [ ] the five experiments are done and the “Setup check” answer sheet is submitted
- [ ] all four Part A problems submitted
- [ ] the “Part B · Chaos in ten lines” answer sheet is submitted
- [ ] the “Part C · Your first terminal session” answer sheet is submitted
- [ ] you have joined the course group

When it will not install

Symptom	Usually means	Fix
`'python' is not recognized`	PATH not set on Windows	Rerun the installer → Modify → tick Add to PATH
`zsh: command not found: python`	different command name on macOS	Try `python3`, or check where it installed
Run button is greyed out	Python extension missing	Install the official Microsoft Python extension
VS Code cannot find an interpreter	none selected	`Ctrl+Shift+P` → `Python: Select Interpreter` → pick 3.13
`can't open file` at runtime	terminal is in the wrong directory	`pwd` to check, `cd` to the file's folder

Still stuck? Put your hand up, or post in the course group. **Include a screenshot of the full error and say which OS and version you are on.**