COL1000: Introduction to Programming

Lab 1 — Setting Up Python and IDLE

This guide is the reference document for the first programming lab. Its purpose is to help you confirm that Python is working, become familiar with IDLE, and successfully create, save, and run a simple Python program.

You are not expected to know Python before this lab. Week 1 is only about getting the programming environment working and learning the basic edit–save–run process. Topics such as variables, user input, conditions, loops, and larger programs will be introduced later in lectures and labs.

Instructions for course platforms such as Piazza and Gradescope will be provided separately.

If something does not work as expected, ask a TA rather than spending a long time trying to repair or install software yourself.

---

1 · Learning objectives

By the end of Lab 1, you should be able to:

- log in to a laboratory computer;
- confirm that Python 3 is available;
- open IDLE;
- distinguish between the IDLE Shell and the IDLE Editor;
- create and save a Python file;
- run a Python program using IDLE;
- find the output of your program;
- recognise a few common errors; and
- locate and reopen your saved file.

The most important skill for this lab is the following cycle:

> **Write → Save → Run → Check the output → Edit → Save → Run again**

---

2 · Logging in to the laboratory computer

1. Sit at an available computer.
2. Log in using the credentials provided for the laboratory, or use the instructed guest session.
3. The laboratory computers use **Linux (Ubuntu)**. You do not need prior Linux experience for this lab.
4. Do not change system settings or attempt to install software.

5. Save your work in the folder specified in this guide.
6. Log out before leaving the laboratory.

Laboratory computers may be reset or reassigned. Do not treat a laboratory computer as permanent storage for personal files.

---

3 · Check that Python is working

Python 3 should already be installed on the laboratory computers.

Step 1: Open a terminal

Press:

```
Ctrl + Alt + T
```

A terminal window should appear.

Step 2: Check the Python version

Type the following command and press **Enter**:

```
python3 --version
```

You should see output similar to:

```
Python 3.x.x
```

The exact version number may be different. The important part is that it begins with **Python 3**.

Step 3: Open IDLE

In the terminal, type:

```
idle3
```

Then press **Enter**.

You may also open IDLE from the applications menu by searching for **IDLE** or **IDLE 3**.

If Python or IDLE does not open, note the error message and ask a TA. Do not try to install Python or IDLE yourself.

---

4 · Understanding IDLE

IDLE is the program that you will use to write and run Python code. It has two main kinds of windows:

1. the **Shell**; and
2. the **Editor**.

It is important to understand the difference between them.

4.1 The IDLE Shell

The Shell is usually the first window that appears. It contains a prompt that looks like this:

```
>>>
```

The Shell lets you enter one instruction at a time. Python runs the instruction immediately after you press **Enter**.

Try the following examples one by one:

```
2 + 3
```

```
10 - 4
```

```
6 * 7
```

```
20 / 4
```

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

The Shell is useful for quick experiments.

Important points about the Shell

- The >>> symbols are the Shell prompt. You do not type them yourself.
- Instructions entered in the Shell run immediately.
- Work typed only in the Shell is **not saved as a program**.
- Do not write your full lab solution in the Shell.
- Closing the Shell removes its current contents from view.

Use the Shell to test small expressions, but use the Editor for programs that must be saved.

4.2 The IDLE Editor

The Editor is where you write a complete Python program.

To open it, select:

```
File → New File
```

or press:

```
Ctrl + N
```

A blank window will appear. Unlike the Shell, the Editor does **not** show a >>> prompt.

The Editor allows you to:

- write several lines of code;
- save the code in a `.py` file;
- edit the program later; and
- run the whole program.

---

5 · Your task for Lab 1

Your task in this lab is to create one Python file in your `col1000` folder, run it in IDLE, and save it with the correct name.

The program should do one simple thing:

- print your entry number on a separate line.

That is the entire required program for this lab. The goal is to practice the full edit-save-run cycle and to check that your file is being saved in the correct place with the correct filename.

Your output should look like this:

```
2026CS10123
```

Save the file using the required naming format, for example:

```
LAB1_Q1_2026CS10123_MON1.py
```

Replace the entry number and slot code with your own details.

Before you finish, make sure you can answer yes to these two questions:

- Did the program print the entry number correctly?
- Is the file named and saved in the required format?

If you want a quick reminder of the basic terminal commands, see the Linux Commands Reference Guide.

6 · Create and run your first Python program

Follow these steps carefully.

Step 1: Open a new Editor window

From the IDLE Shell, select:

```
File → New File
```

Step 2: Type the program

Enter the following code in the Editor:

```
print("Hello, World!")
print("Python is working.")
```

Do not type >>> before either line.

Step 3: Save the file

Press:

```
Ctrl + S
```

The first time you save, IDLE will ask for a filename and location.

Create or select a folder named:

```
col1000
```

Save the program as a Python file ending in `.py`.

For example:

```
LAB1_Q1_2026CS10123_MON1.py
```

Replace the entry number and slot code with your own details.

Step 4: Run the program

Press:

```
F5
```

or select:

```
Run → Run Module
```

IDLE may ask you to save the file again if you changed it. Save it before continuing.

The output will appear in the Shell:

```
Hello, World!
Python is working.
```

You may also see the word:

```
RESTART
```

This is normal. It means IDLE has started a fresh run of your program.

Step 5: Edit and run again

Return to the Editor and add another line:

```
print("My first Python program ran successfully.")
```

Save the file with **Ctrl + S**, then press **F5** again.

This edit–save–run process will be used throughout the course.

---

7 · Saving and organising your files

Course folder

Keep your programming files inside one folder in your home directory:

```
col1000/
```

You can create this folder from the IDLE Save dialog by using **Create Folder**.

Do not scatter lab files across the Desktop, Downloads folder, or unrelated directories.

Filename format

Unless the lab sheet gives a different instruction, use the following format:

```
LAB<n>_Q<n>_<ENTRY NUMBER>_<SLOT CODE>.py
```

Example:

```
LAB1_Q2_2026CS10123_MON1.py
```

Part	Rule	Example
`LAB<n>`	Lab number using one or more digits	`LAB1`
`Q<n>`	Question number	`Q2`
Entry number	Full entry number in capital letters	`2026CS10123`
Slot code	Code for your registered laboratory slot	`MON1`
Extension	Must be `.py`	`.py`

Slot codes

Registered laboratory slot	Code
Monday 9:00–11:00	`MON1`
Monday 11:00–1:00	`MON2`
Tuesday 9:00–11:00	`TUE1`
Wednesday 9:00–11:00	`WED1`
Thursday 9:00–11:00	`THU1`
Thursday 11:00–1:00	`THU2`
Friday 9:00–11:00	`FRI1`

Use the code for the slot in which you are officially registered.

Incorrect filename examples

The following filenames do not follow the required pattern:

```
lab1_q2.py
LAB1_Q2_2026CS10123_MON1.py
LAB1 Q2 2026CS10123 MON1.py
LAB1_Q2_2026cs10123_MON1.py
LAB1_Q2_2026CS10123_MON1.py.txt
LAB1_Q2_2026CS10123_MON1(1).py
```

Before leaving the lab, check that:

- the file is inside the `col1000` folder;
- the filename is correct;
- the entry number is correct;
- the slot code is correct; and
- the file ends in `.py`.

---

8 · The Python needed for Lab 1

Only a very small amount of Python is required this week.

7.1 Displaying text with `print()`

```
1  print("Hello, World!")
```

`print()` displays information in the Shell.

Text must be written inside quotation marks:

```
1  print("Welcome to COL1000")
```

Each `print()` instruction normally produces a new line:

```
1  print("One")
2  print("Two")
3  print("Three")
```

Output:

```
1  One
2  Two
3  Three
```

7.2 Text and calculations are different

Compare the following two instructions:

```
1  print(2 + 3)
2  print("2 + 3")
```

Output:

```
5
2 + 3
```

Without quotation marks, Python evaluates `2 + 3` as a calculation.

With quotation marks, Python treats `2 + 3` as text and displays the characters exactly as written.

7.3 Basic arithmetic

Python can perform arithmetic using:

Symbol	Operation	Example
+	Addition	`2 + 3`
–	Subtraction	`10 – 4`
*	Multiplication	`6 * 7`
/	Division	`20 / 4`

Try these expressions in the Shell.

You may notice that:

```
20 / 4
```

produces:

```
5.0
```

This is expected. Division produces a decimal-number result in Python.

7.4 Comments

A line beginning with # is a comment:

```
# This is a comment for the reader.
print("Hello")
```

Python ignores comments. They are used to explain code to people.

7.5 Printing more than one item

A comma can be used to display multiple items with spaces between them:

```
print("Hello", "World")
```

Output:

```
Hello World
```

For Lab 1, you do not need to use variables, user input, conditions, loops, or functions of your own.

---

9 · Common problems and error messages

Errors are a normal part of programming. A program failing to run does not mean that you are bad at programming.

When an error appears:

1. read the last line of the error message;
2. note the line number;
3. check that line and the line immediately above it;
4. correct one issue at a time; and
5. save and run the program again.

Problem or message	Likely cause	What to check
`SyntaxError: invalid syntax`	A missing bracket, quotation mark, or another typing error	Check the marked line and the line above it
`SyntaxError: unterminated string literal`	A quotation mark was opened but not closed	Count the quotation marks on that line
`NameError: name 'Hello' is not defined`	Text was written without quotation marks	Write `"Hello"` instead of `Hello`
`IndentationError: unexpected indent`	A line starts with an unnecessary space or tab	Remove spaces before the line
Nothing appears after running	The file was not run, or it contains no `print()` instruction	Click the Editor and press **F5**
`RESTART` appears with no other output	The program ran but did not display anything	Check for a correct `print()` line
Changes do not appear in the output	The latest changes were not saved	Press **Ctrl + S**, then **F5**

Problem or message	Likely cause	What to check
>>> causes an error in the Editor	The Shell prompt was copied into the file	Remove the >>> symbols
The file opens as text instead of Python	The filename may end in `.txt`	Confirm that it ends in `.py`
IDLE does not open	IDLE may not have launched correctly	Note the message and ask a TA

If a program seems stuck, click inside the Shell and press:

```
Ctrl + C
```

This interrupts the running program.

---

10 · Reopen a saved program

To reopen a program in IDLE:

1. open IDLE;
2. select **File → Open**;
3. open the `col1000` folder;
4. select the required `.py` file; and
5. click **Open**.

You can then edit the program, save it, and press **F5** to run it again.

You may also use **File → Save As** to create a copy with a different filename.

---

11 · Optional practice

Complete these only after the required setup is working.

Practice 1

```
print("My name is ...")
print("My entry number is ...")
print("My laboratory slot is ...")
```

Replace the dots with your information.

Practice 2

Try the following:

```
print("Hello" + "World")
print("Hello", "World")
```

Observe the difference in spacing.

Practice 3

Try:

```
print("5" + "5")
print(5 + 5)
```

Observe the difference between text and numbers.

Practice 4

Print a blank line:

```
print()
```

Practice 5

Create an error deliberately by removing one quotation mark:

```
print("Hello)
```

Run the program, read the error, repair the line, save the file, and run it again.

---

12 · Getting help

When asking a TA for help, be ready to show:

- the machine number;
- the Editor window containing your code;
- the complete error message in the Shell;
- the location where the file was saved; and
- the steps you followed before the problem occurred.

Do not repeatedly change unrelated settings or attempt to reinstall Python. Most first-lab problems are caused by a small filename, saving, or typing mistake and can be fixed quickly.

Separate instructions will be provided later for course communication, online platforms, submissions, grading, and academic-conduct policies.