1.1 Set up your *Python* development environment. Before running any program, you need access to a *Python* interpreter. Choose one of the options below; **Options A–C** are recommended for regular use, while **Option D** is a browser-based fallback for the first class session if you cannot install software on your machine.

Option A — Thonny (recommended for beginners)

Thonny is an IDE designed specifically for learning *Python*. It installs *Python* automatically, requires no configuration, and includes a step-by-step debugger that shows variable values as your program runs.

(a) Download the Windows installer from `https://thonny.org/` and run it with the default options. *Python* is bundled — no separate installation needed.

(b) Open Thonny. Write your code in the top *Editor* pane and press `F5` (or *Run* → *Run current script*) to execute it.

(c) The bottom *Shell* pane is the interactive interpreter: type expressions at the `>>` prompt and press `Enter` to evaluate them.

(d) To step through a program line by line, use *Run* → *Debug current script* (`F5` in debug mode) and watch variable values update in the *Variables* panel.

Option B — Visual Studio Code (VSCode)

VSCode is a lightweight, general-purpose editor with excellent *Python* support via an official extension.

(a) Download the Windows installer from `https://code.visualstudio.com/` and run it, accepting the default options (tick *Add to PATH* if offered).

(b) Open VSCode, click the *Extensions* icon in the left sidebar (or press `Ctrl+Shift+X`), search for `Python`, and install the extension published by *Microsoft*.

(c) Open a terminal inside VSCode via *Terminal* → *New Terminal* and confirm *Python* is available:

```
python --version
```

(d) To run a script, open any `.py` file; a *Run* button (triangle) appears in the top-right corner. You can also use the integrated terminal:

```
python hello.py
```

(e) For an interactive interpreter session, type `python` (or `python3` on some installations) in the terminal and press `Enter`.

Option C — Pyzo

Pyzo is a beginner-friendly IDE designed specifically for scientific *Python* work. It bundles an editor and an interactive shell in a single window.

(a) Download the Windows installer from `https://pyzo.org/downloads.html` and run it with the default settings.

(b) On first launch, Pyzo will prompt you to select a *Python* environment; choose the version that matches your installed *Python* (if unsure, pick the highest version available).

(c) Write your code in the *Editor* pane (left/top) and execute it with `F5` or via *Run* → *Run file*. The *Shell* pane (right/bottom) serves as the interactive interpreter — you can type expressions there directly at the `>>>` prompt.

Option D — Online compiler (backup for the first class)

If you cannot install any software, use the online *Python* compiler provided by *Programiz*:

`https://www.programiz.com/python-programming/online-compiler/`

No installation is required: type your code in the editor panel and click *Run*. Note that this tool does not support reading files from disk or persistent sessions; it is a temporary fallback only and should be replaced by Option A, B, or C before the next class.

1.2 Execute a first *Python* program.

(a) Using a text editor of your choice, write a small *Python* program:

hello.py
```
print("Hello, Python!")
```

Save this program in a file named `hello.py` in your home directory.

(b) Now let's execute the program you wrote: open the UNIX command interpreter (*Terminal*) and enter the following command:

```
python3 hello.py
```

If everything goes well, you should see the message `"Hello, world!`, printed. Congratulations, you've executed your first *Python* program!

(c) Alternatively, you can write and run the program directly in Thonny (or another IDE from Exercise 1): open Thonny, paste the code into the Editor pane, and press `F5`.

1.3 Let's now obtain some documentation on *Python* programming. On the course website, there's a link to the book *Think Python* in PDF format; this book is freely available. Try downloading it and save it in your area with the name `thinkpython.pdf`. You can also copy it for studying at home, printing, or sharing with friends (read the license on the first pages).

Using the graphical file manager, open a window with your home directory and locate the PDF file. Double-clicking it should allow you to view the document.

1.4 Use the *Python* interpreter (you can also use an IDE such as Thonny: type expressions directly in its Shell pane) to calculate the following expressions:

(a) `45 + 27`

(b) `2**3`

(c) `9 % 4`

(d) `"2"+ "34"`

(e) `2.5 * 4`

(f) `10/3`

(g) `10//3`

(h) `10%3`

1.5 For each of the previous exercise's items, indicate the type of the result; you can confirm your answers using the `type()` function:

```
>>> type("abc")
<class 'str'>
```

1.6 Using the *Python* interpreter (or the Shell pane of your IDE), calculate each of the following expressions. Indicate whether the result is an integer or a floating-point number.

(a) `(10-7)*(4-3)`

(b) `10-7*4-3`

(c) `17 / 3`

(d) `17 // 3`

(e) `17 % 3`

(f) `(3+5)/2*3`

(g) `(3+5)/(2*3)`

(h) `1 + 1/3`

(i) `1 + 1/3.0`

(j) `1 + 1//3`

(k) `2**3`

(l) `2.0**3`

1.7 Simulate the step-by-step execution of the following programs and indicate the final values of the variables; use the *Python Tutor* (`http://pythontutor.com`) to verify the results.

(a)
```
a = 121
b = 45
t = a
a = b
b = t
```

(b)
```
p = 1
p = p * 2
p = p * 3
p = p * 4
```

(c)
```
N = 1
N = N*10 + 2
N = N*10 + 3
N = N*10 + 4
```

(d)
```
x = 2.0
r = x
r = 0.5*(r + x/r)
r = 0.5*(r + x/r)
r = 0.5*(r + x/r)
```

(e)
```
a = 54
b = 24
r = a%b
a = b
b = r
r = a%b
a = b
b = r
```

1.8 Simulate the execution of the following programs, indicating the values of the variables after each step.

(a)
```
s = 0
s = s**2 + 1
s = s**2 + 2
s = s**2 + 3
```

(b)
```
s = 0
s = (s + 1)**2
s = (s + 2)**2
s = (s + 3)**2
```

(c)
```
n = 1
s = 0
s = s + n
n = n + 1
s = s + n
n = n + 1
s = s + n
n = n + 1
```

(d)
```
x = 3
y = 1
y = x*y + 1
y = x*y + 1
y = x*y + 1
```

(e)
```
x = 3
y = x**3+x**2+x+1
```

1.9 Write the following programs in your preferred text editor:

(a)
```
x = input()
print(x)
```

(b)
```
x = input("Please write something: ")
print(x)
```

Execute each of them in the terminal with (for example):

```
$ python3 test.py
[type your name and press ENTER on this line]
```

Alternatively, run the script from your IDE: in Thonny, press `F5`; in VSCode, click the Run button. Input from `input()` is typed in the Shell / terminal pane of the IDE.
Describe what you observed.

1.10 Use the `input` function to write a program that asks the user, `"What's your name? "` and prints the message `"Hello, [NAME]"` where `[NAME]` is the sequence of characters entered when running the program.

1.11 Translate each of the following mathematical expressions into *Python* and execute them in the interpreter (or the Shell pane of your IDE). You can use auxiliary variables to store intermediate values.

(a) $(1 + x)(-1 + 2x)$ for $x = 2$

(b) $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4}$

(c) 2^{2^3}

(d) $\frac{1-x^2}{2x}$ when $x = 5$

(e) $1 + \frac{1}{2+\frac{1}{x}}$ for $x = 2$

(f) $\sqrt{x^2 + y^2}$ for $x = 2$ and $y = 0.5$

(g) $\sqrt{b^2 - 4ac}$ for $a = 1$, $b = 1$, and $c = -1/3$

(h) $\sin(\pi - \alpha)$ for $\alpha = \pi/4$

Mathematical functions and constants are defined in the `math` module: `math.sqrt()`, `math.exp()`, `math.sin()`, `math.cos()`, `math.pi`, etc.

1.12 Execute the following expressions in the *Python* interpreter (or the Shell pane of your IDE). If an error occurs, indicate whether it is a syntax or semantic error. If there is no error, check its type using `type(...)`.

(a) `97+555`

(b) `97$+555$`

(c) `math.sqrt(2)`

(d) `math.sqrt(-2)`

(e) `2(math.pi)`

(f) `2*math.pi`

(g) `str(2*math.pi)`

(h) `int(2*math.pi)`

(i) `"97"+555`

(j) `97 == "97"`

(k) `97 == int("97")`

(l) `102 <= 97`

(m) `102 <= "97"`

(n) `"102" <= "97"`

(o) `"the value of pi is " + math.pi`

(p) `"the value of pi is " + str(math.pi)`

1.13 Write a program that asks the user the kilometers traveled and the number of liters of fuel that a car spent, and print the consumption in liters spent per 100 kilometers.

1.14 Write a program that asks the user for the radius of a circle and then calculates and prints its area.

1.15 Write a program that asks the user for a temperature in degrees Fahrenheit and then prints its equivalent in degrees Celsius. Use the conversion formula $C = \frac{5}{9}(F - 32)$, where F is the temperature in degrees Fahrenheit and C is in degrees Celsius.

1.16 Write a program that asks the user for the lengths a and b of the two sides of a right-angled triangle and then prints the length of the hypotenuse.