Tutorial

A Brief Introduction to Computational Tools

Serhat Beyenir

16 October 2025

Contents

List of Figures

List of Tables

Preface

A Brief Introduction to Computational Tools presents a collection of tutorials based on lecture notes from classes, designed to give learners clear and essential insights into key topics. No previous programming experience is required. Each tutorial guides you step by step through the concepts with hands-on examples.

Chapter 1

Python Tutorial

This tutorial introduces Python programming, covering basic concepts with examples to illustrate key points. We will start by using Python as a calculator, then explore variables, functions, and control flow.

1.1 Requirements

To follow this tutorial, you must have Python (version 3.10 or later) installed on your computer. Python is available for Windows, macOS, and Linux. Additionally, ensure you have a text editor or an Integrated Development Environment (IDE) to write Python code. We recommend Positron, a user-friendly IDE with a built-in terminal for running Python scripts, though other editors like VS Code or PyCharm are also suitable.

Alternatively, you can use a web-based environment like python-fiddle.com

1.2 Basic Syntax

Python uses indentation (typically four spaces) to define code blocks. A colon (`:`) introduces a block, and statements within the block must be indented consistently. Python is case-sensitive, so `Variable` and `variable` are distinct identifiers. Statements typically end with a newline, but you can use a backslash (\) to continue a statement across multiple lines.

```
total = 1 + 2 + 3 + \
        4 + 5
print(total)  # Output: 15
```

Basic syntax rules:

- Comments start with `#` and extend to the end of the line.

- Strings can be enclosed in single quotes (`'`), double quotes (`"`), or triple quotes (`'''` or `"""`) for multi-line strings.
- Python is case-sensitive, so `Variable` and `variable` are different identifiers.

1.3 The `print()` Function

The `print()` function displays output in Python.

```
name = "Rudolf Diesel"
year = 1858
print(f"{name} was born in {year}.")
```

Output: `Rudolf Diesel was born in 1858.`

1.4 Formatting in `print()`

The following table illustrates common f-string formatting options for the `print()` function:

Format	Code	Example	Output
Round to 2 decimals	`f"{x:.2f}"`	`print(f"{3.14159:.2f}")`	`3.14`
Round to whole number	`f"{x:.0f}"`	`print(f"{3.9:.0f}")`	`4`
Thousands separator	`f"{x:,.2f}"`	`print(f"{1234567.89:,.2f}")`	`1,234,567.89`
Percentage	`f"{x:.1%}"`	`print(f"{0.756:.1%}")`	`75.6%`
Currency style	`f"${x:,.2f}"`	`print(f"${1234.5:,.2f}")`	`$1,234.50`

Note: The currency symbol (e.g., `$`) can be modified for other currencies (e.g., `€`, `£`) based on the desired locale.

1.5 Variables and Data Types

Variables store data and are assigned values using the `=` operator.

```
x = 10
y = 3.14
name = "Rudolph"
```

Python has several built-in data types, including:

- Integers (`int`): Whole numbers, e.g., `10`, `-5`
- Floating-point numbers (`float`): Decimal numbers, e.g., `3.14`, `-0.001`

- Strings (`str`): Text, e.g., `"Hello"`, `'World'`
- Booleans (`bool`): `True` or `False`

1.5.1 Arithmetic Operations

```
a = 10
b = 3
print(a + b)  # Addition: 13
print(a - b)  # Subtraction: 7
print(a * b)  # Multiplication: 30
print(a / b)  # Division: 3.3333...
print(a // b) # Integer Division: 3
print(a ** b) # Exponentiation: 1000
```

1.5.2 String Operations

```
first_name = "Rudolph"
last_name = "Diesel"
full_name = first_name + " " + last_name  # Concatenation using +
print(full_name)  # Output: Rudolph Diesel
print(f"{first_name} {last_name}")  # Concatenation using f-string
print(full_name * 2)  # Repetition: Rudolph DieselRudolph Diesel
print(full_name.upper())  # Uppercase: RUDOLPH DIESEL
```

Note: String repetition (`*`) concatenates the string multiple times without spaces. For example, `full_name * 2` produces `Rudolph DieselRudolph Diesel`.

1.6 Python as a Calculator in Interactive Mode

Python's interactive mode allows you to enter commands and see results immediately, ideal for quick calculations. To start, open a terminal (on macOS, Linux, or Windows) and type:

```
python3  # Use 'python' on Windows if 'python3' is not recognized
```

You should see the Python prompt:

```
>>>
```

Enter expressions and press **Enter** to see results:

```
2 + 3  # Output: 5
7 - 4  # Output: 3
6 * 9  # Output: 54
8 / 2  # Output: 4.0
8 // 2 # Output: 4
2 ** 3 # Output: 8
```

1.6.1 Parentheses for Grouping

```
(2 + 3) * 4  # Output: 20
2 + (3 * 4)  # Output: 14
```

1.6.2 Variables

```
x = 10
y = 3
x / y  # Output: 3.3333333333333335
```

1.6.3 Exiting Interactive Mode

To exit, type:

```
exit()
```

Alternatively, use: - **Ctrl+D** (macOS/Linux) - **Ctrl+Z** then Enter (Windows)

1.7 Control Flow

Control flow statements direct the execution of code based on conditions.

1.7.1 Conditional Statements

Conditional statements allow you to execute different code blocks based on specific conditions. Python provides three keywords for this purpose:

- `if`: Evaluates a condition and executes its code block if the condition is `True`.
- `elif`: Short for "else if," it checks an additional condition if the preceding `if` or `elif` conditions are `False`. You can use multiple `elif` statements to test multiple conditions sequentially, and Python will execute the first `True` condition's block, skipping the rest.
- `else`: Executes a code block if none of the preceding `if` or `elif` conditions are `True`. It serves as a fallback and does not require a condition.

The following example uses age to categorize a person as a Minor, Adult, or Senior, demonstrating how `if`, `elif`, and `else` work together.

```
# Categorize a person based on their age
age = 19
if age < 18:
    print("Minor")
elif age <= 64:
    print("Adult")
```

```
else:
    print("Senior")
```

Output: `Adult`

1.7.2 For Loop

A `for` loop iterates over a sequence (e.g., list or string).

```
components = ["piston", "liner", "connecting rod"]
for component in components:
    print(component)
```

Output:

```
piston
liner
connecting rod
```

1.7.3 While Loop

A `while` loop executes as long as a condition is true. Ensure the condition eventually becomes false to avoid infinite loops.

```
count = 0
while count <= 5:
    print(count)
    count += 1
```

Output:

```
0
1
2
3
4
5
```

1.8 Functions

1.8.1 The `def` Keyword

Functions are reusable code blocks defined using the `def` keyword. They can include default parameters for optional arguments.

```
def add(a, b=0):
    return a + b
print(add(5))      # Output: 5
```

```
print(add(5, 3))   # Output: 8

def multiply(*args):
    result = 1
    for num in args:
        result *= num
    return result
print(multiply(2, 3, 4))  # Output: 24
```

1.8.2 The `lambda` Keyword

The `lambda` keyword creates anonymous functions for short, one-off operations, often used in functional programming.

```
celsius_to_fahrenheit = lambda c: (c * 9 / 5) + 32
print(celsius_to_fahrenheit(25))  # Output: 77.0
```

1.9 The `math` Module

The `math` module provides mathematical functions and constants.

```
import math
print(math.sqrt(16))  # Output: 4.0
print(math.pi)        # Output: 3.141592653589793
```

```
import math
angle = math.pi / 4  # 45 degrees in radians
print(math.sin(angle))  # Output: 0.7071067811865475 (approximately √2/2)
print(math.cos(angle))  # Output: 0.7071067811865476 (approximately √2/2)
print(math.tan(angle))  # Output: 1.0
```

Note: Floating-point arithmetic may result in small precision differences, as seen in the `sin` and `cos` outputs.

```
import math
print(math.log(10))       # Natural logarithm of 10: 2.302585092994046
print(math.log(100, 10))  # Logarithm of 100 with base 10: 2.0
```

1.9.1 Converting Between Radians and Degrees

The `math` module provides `math.radians()` to convert degrees to radians and `math.degrees()` to convert radians to degrees, which is useful for trigonometric calculations.

```
import math
degrees = 180
radians = math.radians(degrees)
```

```
print(f"{degrees} degrees is {radians:.3f} radians")  # Output: 180 degrees is 3.142 radians

radians = math.pi / 2
degrees = math.degrees(radians)
print(f"{radians:.3f} radians is {degrees:.1f} degrees")  # Output: 1.571 radians is 90.0 degrees
```

1.10 Writing Python Scripts

Write Python code in a `.py` file and run it as a script. Create a file named `script.py`:

```
# script.py
import math
print("Square root of 16 is:", math.sqrt(16))
print("Value of pi is:", math.pi)
print("Sine of 90 degrees is:", math.sin(math.pi / 2))
print("Natural logarithm of 10 is:", math.log(10))
print("Logarithm of 100 with base 10 is:", math.log(100, 10))
```

To run the script, open a terminal, navigate to the directory containing `script.py` using the `cd` command (e.g., `cd /path/to/directory`), and type:

```
python3 script.py  # or python script.py on Windows
```

Output:

```
Square root of 16 is: 4.0
Value of pi is: 3.141592653589793
Sine of 90 degrees is: 1.0
Natural logarithm of 10 is: 2.302585092994046
Logarithm of 100 with base 10 is: 2.0
```

1.11 Summary

This tutorial covered Python basics, including syntax, variables, data types, operations, control flow, and functions. Python's rich ecosystem includes libraries like:

- **NumPy**: For numerical computations and array manipulations.
- **Matplotlib**: For data visualization and plotting.
- **Pandas**: For data manipulation and analysis with tabular data structures.
- **Pint**: For handling physical quantities and performing unit conversions.

You can explore these libraries to enhance your Python programming skills further. For example installing them can be done using `pip`:

```
pip install numpy matplotlib pandas pint
```

`pip` is Python's package manager for installing and managing additional libraries.

Chapter 2

SI Units

2.1 Condenser Vacuum

Condenser vacuum gauge reads 715 mmHg when barometer stands at 757 mmHg. State the absolute pressure in kN/m² and bar.

2.1.1 Given Data

$$P_{atm} = 757 \text{ mmHg}, \quad P_{vac} = 715 \text{ mmHg}$$

2.1.2 Absolute Pressure in mmHg

$$P_{abs} = P_{atm} - P_{vac} = 757 - 715 = 42 \text{ mmHg}$$

2.1.3 Convert mmHg → kN/m²

$$P = \rho g h = 13{,}600 \cdot 9.81 \cdot 0.001 = 133.416 \text{ Pa} = 0.133416 \text{ kN/m}^2$$

$$P_{abs} = 42 \cdot 0.133416 = 5.6034 \text{ kN/m}^2$$

2.1.4 Convert kN/m² → bar

$$P_{abs} = \frac{5.6034}{100} = 0.056 \text{ bar}$$

2.1.5 Final Answers

$$\boxed{P_{abs} = 42 \text{ mmHg} = 5.6034 \text{ kN/m}^2 = 0.056 \text{ bar}}$$

2.1.6 Python Code

```
P_atm_mmHg = 757
P_vac_mmHg = 715
MMHG_TO_KN_M2 = 0.133416
KNM2_TO_BAR = 1 / 100
P_abs_mmHg = P_atm_mmHg - P_vac_mmHg
P_abs_kNm2 = P_abs_mmHg * MMHG_TO_KN_M2
P_abs_bar = P_abs_kNm2 * KNM2_TO_BAR
print(f"Absolute Pressure = {P_abs_mmHg:.2f} mmHg")
print(f"Absolute Pressure = {P_abs_kNm2:.3f} kN/m²")
print(f"Absolute Pressure = {P_abs_bar:.4f} bar")
```

2.2 Oil Flow in Tubes

Oil flows full bore at a velocity of $v = 2$ m/s through 16 tubes of diameter $d = 30$ mm. Density of oil: $\rho = 0.85$ g/mL. Find **volume flow rate** (L/s) and **mass flow rate** (kg/min).

2.2.1 Cross-sectional area of one tube

$$A = \pi \frac{d^2}{4} = \pi \frac{0.03^2}{4} \approx 7.0686 \times 10^{-4} \text{ m}^2$$

2.2.2 Total area and volume flow rate

$$A_{\text{total}} = 16 \cdot 7.0686 \times 10^{-4} \approx 0.01131 \text{ m}^2$$

$$Q = A_{\text{total}} \cdot v \approx 0.02262 \text{ m}^3/\text{s} \approx 22.62 \text{ L/s}$$

2.2.3 Mass flow rate

$$\dot{m} = \rho \cdot Q = 850 \cdot 0.02262 \approx 19.227 \text{ kg/s} \approx 1153.6 \text{ kg/min}$$

2.2.4 Final Answers

Volume flow rate: $Q \approx 22.6$ L/s

Mass flow rate: $\dot{m} \approx 1154$ kg/min

2.2.5 Python Code

```
import math
v = 2.0
N = 16
```

```
d = 0.03
rho = 0.85 * 1000
A = math.pi * d**2 / 4
A_total = N * A
Q_m3_s = A_total * v
Q_L_s = Q_m3_s * 1000
m_dot_kg_s = rho * Q_m3_s
m_dot_kg_min = m_dot_kg_s * 60
print(f"Volume flow rate: {Q_L_s:.2f} L/s")
print(f"Mass flow rate: {m_dot_kg_min:.2f} kg/min")
```

2.3 Gauge Pressure

An oil of specific gravity (relative density) SG = 0.8 is contained in a vessel to a depth of $h = 2$ m. Find the **gauge pressure** at this depth in kPa.

2.3.1 Gauge Pressure

$$P_g = \rho g h$$

where

ρ = density of fluid (kg/m³)

g= acceleration due to gravity (9.81 m/s²)

h = depth (m)

2.3.2 Compute the density of oil using specific gravity

Specific gravity is defined as

$$\text{SG} = \frac{\rho_{\text{oil}}}{\rho_{\text{water}}}$$

where $\rho_{\text{water}} = 1000$ kg/m³. Thus,

$$\rho_{\text{oil}} = \text{SG} \times \rho_{\text{water}} = 0.8 \times 1000 = 800 \text{ kg/m}^3$$

2.3.3 Compute the gauge pressure

$$P_g = \rho g h = 800 \times 9.81 \times 2$$

$$P_g = 15696 \text{ Pa} \approx 15.7 \text{ kPa}$$

2.3.4 Answer

The **gauge pressure** at a depth of 2 m in the oil is:

$$\boxed{15.7 \text{ kPa}}$$

2.3.5 Python Code

```
# Gauge Pressure Calculation for Oil

# Given data
specific_gravity = 0.8  # SG of oil
depth_m = 2.0           # depth in meters
g = 9.81                # acceleration due to gravity in m/s²
rho_water = 1000        # density of water in kg/m³

# Compute density of oil
rho_oil = specific_gravity * rho_water

# Compute gauge pressure (Pa)
P_g_Pa = rho_oil * g * depth_m

# Convert to kPa
P_g_kPa = P_g_Pa / 1000

# Print results
print(f"Density of oil: {rho_oil:.1f} kg/m³")
print(f"Gauge pressure at {depth_m} m depth: {P_g_Pa:.1f} Pa ({P_g_kPa:.2f} kPa)")
```

2.4 Absolute Pressure from Manometer Reading

A water manometer shows a pressure in a vessel of 400 mm **below atmospheric pressure**. The atmospheric pressure is measured as 763 mmHg. Determine the **absolute pressure** in the vessel in kPa.

2.4.1 Relationship between absolute and gauge pressure

$$P_{\text{abs}} = P_{\text{atm}} + P_{\text{gauge}}$$

Since the manometer shows a pressure **below atmospheric**, the gauge pressure is negative:

$$P_{\text{gauge}} = -\rho_{\text{water}} g h$$

2.4.2 Convert atmospheric pressure to Pa using

$$P = \rho g h = 13{,}600 \cdot 9.81 \cdot 0.001 = 133.416 \text{ Pa}$$

So

$$P_{\text{atm}} = 763 \times 133.416 \approx 101,801 \text{ Pa} \approx 101.8 \text{ kPa}$$

2.4.3 Compute gauge pressure

Water column height:

$$h = 400 \text{ mm} = 0.4 \text{ m}$$

Density of water: $rho_{\text{water}} = 1000 \text{ kg/m}^3$, $g = 9.81 \text{ m/s}^2$

$$P_{\text{gauge}} = -\rho g h = -1000 \times 9.81 \times 0.4$$

$$P_{\text{gauge}} = -3924 \text{ Pa} \approx -3.92 \text{ kPa}$$

2.4.4 Compute absolute pressure

$$P_{\text{abs}} = P_{\text{atm}} + P_{\text{gauge}} \approx 101.8 - 3.92 \approx 97.9 \text{ kPa}$$

2.4.5 Answer

The **absolute pressure** in the vessel is:

$$\boxed{97.9 \text{ kPa}}$$

2.4.6 Python Code

```
# Absolute Pressure Calculation from Water Manometer

# Given data
h_mm = 400               # manometer reading in mm (below atmospheric)
atm_mmHg = 763           # atmospheric pressure in mmHg
rho_water = 1000         # density of water in kg/m³
g = 9.81                 # gravity in m/s²
mmHg_to_Pa = 133.416     # conversion factor

# Convert manometer height to meters
h_m = h_mm / 1000
```

```
# Convert atmospheric pressure to Pa
P_atm_Pa = atm_mmHg * mmHg_to_Pa

# Gauge pressure (negative because below atmospheric)
P_gauge_Pa = - rho_water * g * h_m

# Absolute pressure
P_abs_Pa = P_atm_Pa + P_gauge_Pa

# Convert to kPa
P_abs_kPa = P_abs_Pa / 1000

# Print results
print(f"Atmospheric pressure: {P_atm_Pa:.1f} Pa ({P_atm_Pa/1000:.1f} kPa)")
print(f"Gauge pressure: {P_gauge_Pa:.1f} Pa ({P_gauge_Pa/1000:.2f} kPa)")
print(f"Absolute pressure in the vessel: {P_abs_Pa:.1f} Pa ({P_abs_kPa:.2f} kPa)")
```

Chapter 3

Heat and Work

3.1 Heat Required to Heat Steel

A steel block of mass $m = 5$ kg and specific heat capacity $c = 480$ J/kg $\cdot$ K is heated from $T_1 = 15°$C to $T_2 = 100°$C. Determine the **heat required**.

3.1.1 The formula for heat

$$Q = mc\Delta T$$

where

$\Delta T = T_2 - T_1$ is the temperature change.

3.1.2 Compute the temperature change

$$\Delta T = T_2 - T_1 = 100 - 15 = 85 \text{ K}$$

3.1.3 Compute the heat required

$$Q = mc\Delta T = 5 \times 480 \times 85$$

$$Q = 204,000 \text{ J}$$

3.1.4 Answer

The **heat required** to raise the temperature of the steel is:

$$\boxed{204 \text{ kJ}}$$

3.1.5 Python Code

```python
# Heat Required to Heat Steel

# Given data
mass = 5            # kg
specific_heat = 480 # J/kg·K
T_initial = 15      # °C
T_final = 100       # °C

# Temperature change
delta_T = T_final - T_initial

# Heat required (in J)
Q_J = mass * specific_heat * delta_T

# Convert to kJ
Q_kJ = Q_J / 1000

# Print results
print(f"Temperature change: {delta_T} K")
print(f"Heat required: {Q_J:.0f} J ({Q_kJ:.0f} kJ)")
```

```python
# Interactive Heat Calculation

# Get user input
mass = float(input("Enter the mass of the object (kg): "))
specific_heat = float(input("Enter the specific heat capacity (J/kg·K): "))
T_initial = float(input("Enter the initial temperature (°C): "))
T_final = float(input("Enter the final temperature (°C): "))

# Calculate temperature change
delta_T = T_final - T_initial

# Calculate heat required
Q_J = mass * specific_heat * delta_T
Q_kJ = Q_J / 1000

# Display results
print("\nCalculation Results:")
print(f"Temperature change: {delta_T:.2f} K")
print(f"Heat required: {Q_J:.2f} J ({Q_kJ:.2f} kJ)")
```

3.2 Finding Specific Heat Capacity

A liquid of mass $m = 4$ kg is heated from $T_1 = 15^\circ$C to $T_2 = 100^\circ$C. The heat supplied is $Q = 714$ kJ. Determine the **specific heat capacity** c of the liquid.

3.2.1 Recall the formula for heat

$$Q = mc\Delta T$$

where $\Delta T = T_2 - T_1$.

3.2.2 Convert heat to joules

$$Q = 714 \text{ kJ} = 714 \times 1000 = 714,000 \text{ J}$$

3.2.3 Compute temperature change

$$\Delta T = T_2 - T_1 = 100 - 15 = 85 \text{ K}$$

3.2.4 Solve for specific heat capacity

$$c = \frac{Q}{m\Delta T} = \frac{714,000}{4 \times 85}$$

$$c = \frac{714,000}{340} \approx 2100 \text{ J/kg} \cdot \text{K}$$

3.2.5 Answer

The **specific heat capacity** of the liquid is:

$$\boxed{c \approx 2100 \text{ J/kg} \cdot \text{K}}$$

3.2.6 Python Code

```
# Specific Heat Capacity Calculation

# Given data
mass = 4          # kg
T_initial = 15    # °C
T_final = 100     # °C
Q_kJ = 714        # heat supplied in kJ

# Convert heat to joules
Q_J = Q_kJ * 1000
```

```
# Temperature change
delta_T = T_final - T_initial

# Calculate specific heat capacity
c = Q_J / (mass * delta_T)

# Print results
print(f"Temperature change: {delta_T} K")
print(f"Specific heat capacity: {c:.0f} J/kg·K")
```

```
# Interactive Specific Heat Capacity Calculator

# Get user input
mass = float(input("Enter the mass of the liquid (kg): "))
T_initial = float(input("Enter the initial temperature (°C): "))
T_final = float(input("Enter the final temperature (°C): "))
Q_kJ = float(input("Enter the heat supplied (kJ): "))

# Convert heat to joules
Q_J = Q_kJ * 1000

# Temperature change
delta_T = T_final - T_initial

# Calculate specific heat capacity
c = Q_J / (mass * delta_T)

# Display results
print("\nCalculation Results:")
print(f"Temperature change: {delta_T:.2f} K")
print(f"Specific heat capacity: {c:.2f} J/kg·K")
```

Appendix A

SI System Common Mistakes

Using the SI system correctly is crucial for clear communication in science and engineering. Below are common mistakes in using the SI system, examples of incorrect usage, and how to correct them.

Table A.1: SI system rules and common mistakes

Concept	Mistake	Correct Usage	Notes
Use of SI Unit Symbols	`m./s`	`m/s`	Use the correct format without additional punctuation.
Spacing Between Value & Unit	`10kg`	`10 kg`	Always leave a space between the number and the unit symbol.
Incorrect Unit Symbols	`sec`, `hrs`, `°K`	`s`, `h`, `K`	Use the proper SI symbols; symbols are case-sensitive.
Abbreviations for Units	`5 kilograms (kgs)`	`5 kilograms (kg)`	Avoid informal abbreviations like "kgs"; adhere to standard symbols.

Concept	Mistake	Correct Usage	Notes
Multiple Units in Expressions	`5 m/s/s`, `5 kg/meter²`	`5 m/s²`, `5 kg/m²`	Use compact, standardized formats for derived units.
Incorrect Use of Prefixes	`0.0001 km`	`100 mm`	Choose prefixes to keep numbers in the range (0.1 x < 1000).
Misplaced Unit Symbols	`5/s`, `kg10`	`5 s⁻¹`, `10 kg`	Symbols must follow numerical values, not precede them.
Degrees Celsius vs. Kelvin	`300°K`	`300 K`	Kelvin is written without "degree"
Singular vs. Plural Units	`5 kgs`, `1 meters`	`5 kg`, `1 meter`	Symbols do not pluralize; full unit names follow grammar rules.
Capitalization of Symbols	`Kg`, `S`, `Km`, `MA`	`kg`, `s`, `km`, `mA`	Symbols are case-sensitive; use uppercase only where specified (e.g., `N`, `Pa`).
Capitalization of Unit Names	`Newton`, `Pascal`, `Watt`	`newton`, `pascal`, `watt`	Unit names are lowercase, even if derived from a person's name, unless starting a sentence.
Prefix Capitalization	`MilliMeter`, `MegaWatt`	`millimeter`, `megawatt`	Prefixes are lowercase for (10^{-1}) to (10^{-9}), uppercase for (10^6) and larger (except `k` for kilo).
Formatting in Reports	`5`, `Temperature: 300`	`5 kg`, `Temperature: 300 K`	Always specify units explicitly.

Appendix B

Greek Letters

The following tables present the names of Greek letters and selected symbols commonly used in engineering courses, ensuring precise reference and avoiding reliance on informal descriptors such as "squiggle."

Table B.1: Greek letters.

Lower Case	Upper Case	Name
α	A	alpha
β	B	beta
γ	Γ	gamma
δ	Δ	delta
ϵ	E	epsilon
ζ	Z	zeta
η	E	eta
θ	Θ	theta
ι	I	iota
κ	K	kappa
λ	Λ	lambda
μ	M	mu
ν	N	nu
ξ	Ξ	xi
o	O	omicron
π	Π	pi
ρ	P	rho
σ	Σ	sigma
τ	T	tau
υ	Υ	upsilon
ϕ	Φ	phi
χ	X	chi

Lower Case	Upper Case	Name
ψ	Ψ	psi
ω	Ω	omega

Table B.2: Commonly used symbols in engineering courses.

Symbol	Name	Use	Course
Δ	Delta	Change	Thermodynamics
Δ	Delta	Displacement	Naval Architecture
∇	Nabla	Volume	Naval Architecture
Σ	Sigma	Sum	Thermodynamics, Naval Architecture, Applied Mechanics
σ	Sigma	Stress	Thermodynamics, Applied Mechanics
ϵ	Epsilon	Modulus of elasticity	Thermodynamics, Applied Mechanics
η	Eta	Efficiency	Thermodynamics
ω	Omega	Angular velocity	Thermodynamics, Applied Mechanics
ρ	Rho	Density	Thermodynamics, Naval Architecture
τ	Tau	Torque	Thermodynamics, Applied Mechanics

Index