

# Tutorial

A Brief Introduction to Computational Tools

Serhat Beyenir

22 October 2025

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0  
International License (CC BY-SA 4.0).

Full text available at <https://creativecommons.org/licenses/by-sa/4.0/>.

# Contents

<b>Preface</b>	<b>1</b>
<b>1 Python Tutorial</b>	<b>3</b>
1.1 Requirements . . . . .	3
1.2 Basic Syntax . . . . .	3
1.3 The <code>print()</code> Function . . . . .	4
1.4 Formatting in <code>print()</code> . . . . .	4
1.5 Variables and Data Types . . . . .	4
1.5.1 Arithmetic Operations . . . . .	5
1.5.2 String Operations . . . . .	5
1.6 Python as a Calculator in Interactive Mode . . . . .	5
1.6.1 Parentheses for Grouping . . . . .	6
1.6.2 Variables . . . . .	6
1.6.3 Exiting Interactive Mode . . . . .	6
1.7 Control Flow . . . . .	6
1.7.1 Conditional Statements . . . . .	6
1.7.2 For Loop . . . . .	7
1.7.3 While Loop . . . . .	7
1.8 Functions . . . . .	7
1.8.1 The <code>def</code> Keyword . . . . .	7
1.8.2 The <code>lambda</code> Keyword . . . . .	8
1.9 The <code>math</code> Module . . . . .	8
1.9.1 Converting Between Radians and Degrees . . . . .	8
1.10 Writing Python Scripts . . . . .	9
1.11 Summary . . . . .	9
<b>2 SI Units</b>	<b>11</b>
2.1 Condenser Vacuum . . . . .	11
2.1.1 Given Data . . . . .	11
2.1.2 Absolute Pressure in mmHg . . . . .	11
2.1.3 Convert mmHg $\rightarrow$ kN/m <sup>2</sup> . . . . .	11
2.1.4 Convert kN/m <sup>2</sup> $\rightarrow$ bar . . . . .	11
2.1.5 Final Answers . . . . .	11

2.1.6	Code . . . . .	12
2.2	Oil Flow in Tubes . . . . .	12
2.2.1	Cross-sectional area of one tube . . . . .	12
2.2.2	Total area and volume flow rate . . . . .	12
2.2.3	Mass flow rate . . . . .	12
2.2.4	Final Answers . . . . .	12
2.2.5	Code . . . . .	12
2.3	Gauge Pressure . . . . .	13
2.3.1	Gauge Pressure . . . . .	13
2.3.2	Compute the density of oil using specific gravity . . . . .	13
2.3.3	Compute the gauge pressure . . . . .	13
2.3.4	Answer . . . . .	14
2.3.5	Code . . . . .	14
2.4	Absolute Pressure from Manometer Reading . . . . .	14
2.4.1	Relationship between absolute and gauge pressure . . . . .	14
2.4.2	Convert atmospheric pressure to Pa using . . . . .	15
2.4.3	Compute gauge pressure . . . . .	15
2.4.4	Compute absolute pressure . . . . .	15
2.4.5	Answer . . . . .	15
2.4.6	Code . . . . .	15
<b>3</b>	<b>Heat and Work</b>	<b>17</b>
3.1	Heat Required to Heat Steel . . . . .	17
3.1.1	The formula for heat . . . . .	17
3.1.2	Compute the temperature change . . . . .	17
3.1.3	Compute the heat required . . . . .	17
3.1.4	Answer . . . . .	17
3.1.5	Code . . . . .	18
3.2	Finding Specific Heat Capacity . . . . .	19
3.2.1	Recall the formula for heat . . . . .	19
3.2.2	Convert heat to joules . . . . .	19
3.2.3	Compute temperature change . . . . .	19
3.2.4	Solve for specific heat capacity . . . . .	19
3.2.5	Answer . . . . .	19
3.2.6	Code . . . . .	19
3.3	Work Done by Fluid in Cylinder . . . . .	20
3.3.1	The formula for work done at constant pressure . . . . .	20
3.3.2	Compute the change in volume . . . . .	21
3.3.3	Convert pressure to Pa . . . . .	21
3.3.4	Compute the work done . . . . .	21
3.3.5	Answer . . . . .	21
3.3.6	Code . . . . .	21
3.4	Enthalpy Calculation . . . . .	22
3.4.1	Answers . . . . .	23
3.4.2	Code . . . . .	23

<b>4</b>	<b>Perfect Gases</b>	<b>25</b>
4.1	Boyle's Law — Pressure–Volume Relationship . . . . .	25
4.1.1	Given: . . . . .	25
4.1.2	Final Volume of Gas . . . . .	25
4.1.3	Initial Volume of Gas . . . . .	26
4.1.4	Code . . . . .	26
4.2	Compression — Stroke Volume and Final Temperature . . . . .	27
4.2.1	Solution . . . . .	27
4.2.2	Final temperature . . . . .	28
4.2.3	Answers . . . . .	28
4.2.4	Code . . . . .	28
<b>5</b>	<b>Ideal Cycles</b>	<b>31</b>
5.1	Ideal Constant-Volume (Otto) Cycle . . . . .	31
5.1.1	Theory and formulas . . . . .	31
5.1.2	Code . . . . .	32
5.1.3	Using Actual Volume Ratios Instead of Direct $r$ . . . . .	33
5.1.4	Given . . . . .	33
5.1.5	Volume-explicit relations . . . . .	33
5.1.6	Code . . . . .	34
5.2	Ideal Constant-Pressure (Diesel) Cycle . . . . .	35
5.2.1	Code . . . . .	35
	<b>Appendices</b>	<b>39</b>
<b>A</b>	<b>SI System Common Mistakes</b>	<b>39</b>
<b>B</b>	<b>Greek Letters</b>	<b>41</b>



# List of Figures





# List of Tables

A.1	SI system rules and common mistakes . . . . .	39
B.1	Greek letters. . . . .	41
B.2	Commonly used symbols in engineering courses. . . . .	42



# 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](https://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	<code>f"{x:.2f}"</code>	<code>print(f"{3.14159:.2f}")</code>	3.14
Round to whole number	<code>f"{x:.0f}"</code>	<code>print(f"{3.9:.0f}")</code>	4
Thousands separator	<code>f"{x:, .2f}"</code>	<code>print(f"{1234567.89:, .2f}")</code>	1,234,567.89
Percentage	<code>f"{x:.1%}"</code>	<code>print(f"{0.756:.1%}")</code>	75.6%
Currency style	<code>f"\${x:, .2f}"</code>	<code>print(f"\${1234.5:, .2f}")</code>	\$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  $\sqrt{2}/2$ )
print(math.cos(angle))  # Output: 0.7071067811865476 (approximately  $\sqrt{2}/2$ )
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/m<sup>2</sup> 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 $\rightarrow$ kN/m<sup>2</sup>

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

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

#### 2.1.4 Convert kN/m<sup>2</sup> $\rightarrow$ bar

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

#### 2.1.5 Final Answers

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

### 2.1.6 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

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

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

### 2.2.3 Mass flow rate

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

### 2.2.4 Final Answers

Volume flow rate:  $\dot{v} \approx 22.6$  L/s

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

### 2.2.5 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
v_dot_m3_s = A_total * v
v_dot_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: {v_dot_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

$$P_g = \rho gh$$

where

$\rho$  = density of fluid (kg/m<sup>3</sup>)

$g$  = acceleration due to gravity (9.81 m/s<sup>2</sup>)

$h$  = depth (m)

### 2.3.2 Compute the density of oil using specific gravity

Specific gravity is defined as

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

where  $\rho_{\text{water}} = 1000$  kg/m<sup>3</sup>. Thus,

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

### 2.3.3 Compute the gauge pressure

$$P_g = \rho gh = 800 \times 9.81 \times 2$$

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

### 2.3.4 Answer

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

15.7 kPa

### 2.3.5 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:

$$P_{\text{gauge}} = -\rho_{\text{water}}gh$$



### 2.4.2 Convert atmospheric pressure to Pa using

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

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

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

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

### 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:

97.9 kPa
----------

### 2.4.6 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 · K is heated from  $T_1 = 15^\circ\text{C}$  to  $T_2 = 100^\circ\text{C}$ . 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:

204 kJ
--------

### 3.1.5 Code

```
# 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)")

# 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  $T_1 = 15^\circ\text{C}$  to  $T_2 = 100^\circ\text{C}$ . 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:

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

### 3.2.6 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")

```

### 3.3 Work Done by Fluid in Cylinder

A fluid in a cylinder is at pressure  $P = 700$  kPa. It is **expanded at constant pressure** from a volume of  $V_1 = 0.28$  m<sup>3</sup> to  $V_2 = 1.68$  m<sup>3</sup>. Determine the **work done**.

#### 3.3.1 The formula for work done at constant pressure

$$W = P\Delta V$$

where

$$\Delta V = V_2 - V_1$$

**3.3.2 Compute the change in volume**

$$\Delta V = 1.68 - 0.28 = 1.40 \text{ m}^3$$

**3.3.3 Convert pressure to Pa**

$$P = 700 \text{ kPa} = 700 \times 10^3 \text{ Pa} = 700,000 \text{ Pa}$$

**3.3.4 Compute the work done**

$$W = P\Delta V = 700,000 \times 1.40$$

$$W = 980,000 \text{ J} \approx 980 \text{ kJ}$$

**3.3.5 Answer**

The **work done** by the fluid during expansion is:

980 kJ
--------

**3.3.6 Code**

```
# Work Done by Fluid at Constant Pressure

# Given data
P_kPa = 700          # pressure in kPa
V1 = 0.28            # initial volume in m³
V2 = 1.68            # final volume in m³

# Convert pressure to Pa
P_Pa = P_kPa * 1000

# Compute change in volume
delta_V = V2 - V1

# Compute work done (J)
W_J = P_Pa * delta_V

# Convert to kJ
W_kJ = W_J / 1000

# Print results
print(f"Pressure: {P_Pa} Pa")
print(f"Change in volume: {delta_V:.2f} m³")
print(f"Work done: {W_J:.0f} J ({W_kJ:.0f} kJ)")
```

```

# Interactive Work Done Calculator

# Get user input
P_kPa = float(input("Enter the constant pressure (kPa): "))
V1 = float(input("Enter the initial volume (m³): "))
V2 = float(input("Enter the final volume (m³): "))

# Convert pressure to Pa
P_Pa = P_kPa * 1000

# Compute change in volume
delta_V = V2 - V1

# Compute work done (J)
W_J = P_Pa * delta_V

# Convert to kJ
W_kJ = W_J / 1000

# Display results
print("\nCalculation Results:")
print(f"Pressure: {P_Pa:.0f} Pa ({P_kPa:.0f} kPa)")
print(f"Change in volume: {delta_V:.2f} m³")
print(f"Work done: {W_J:.0f} J ({W_kJ:.2f} kJ)")

```

### 3.4 Enthalpy Calculation

120 kg of steam at  $p = 1000$  kPa and  $T = 200^\circ\text{C}$  has internal energy  $u = 2623$  kJ/kg and specific volume ( $v = 0.2061$  m³/kg). Calculate the specific enthalpy ( $h$ ) and the total enthalpy for the 120-kg mass.

Given for steam:

- mass  $m = 120$  kg
- pressure  $P = 1000$  kPa
- temperature  $T = 200^\circ\text{C}$  (not needed for calculation)
- internal energy  $u = 2623$  kJ/kg
- specific volume  $v = 0.2061$  m³/kg

The specific enthalpy is

$$h = u + Pv$$

Note: with (P) in kPa and (v) in m³/kg, the product (Pv) has units kJ/kg (since  $1 \text{ kPa m}^3 = 1 \text{ kJ}$ ).



Compute (Pv):

$$Pv = 1000 \times 0.2061 = 206.1 \text{ kJ/kg}$$

So the specific enthalpy is:

$$h = 2623 + 206.1 = 2829.1 \text{ kJ/kg}$$

Total enthalpy for the 120 kg of steam:

$$H = m h = 120 \times 2829.1 = 339,492 \text{ kJ}$$

### 3.4.1 Answers

- Specific enthalpy:  $h = 2829.1 \text{ kJ/kg}$
- Total enthalpy for 120 kg:  $H = 339,492 \text{ kJ}$

### 3.4.2 Code

```
# Given data
m = 120          # kg
P = 1000         # kPa
u = 2623         # kJ/kg
v = 0.2061       # m³/kg

# Enthalpy calculation
h = u + P * v    # kJ/kg
H_total = m * h  # kJ

print(f"Specific enthalpy h = {h:.1f} kJ/kg")
print(f"Total enthalpy H = {H_total:,.0f} kJ")
```



## Chapter 4

# Perfect Gases

### 4.1 Boyle's Law — Pressure–Volume Relationship

Assuming compression according to the law  $pV = \text{constant}$ :

1. Calculate the final volume when  $1 \text{ m}^3$  of gas at a pressure of  $120 \text{ kN/m}^2$  is compressed to a pressure of  $960 \text{ kN/m}^2$ .
2. Calculate the initial volume of a gas at a pressure of  $1.05 \text{ bar}$  that will occupy a volume of  $5.6 \text{ m}^3$  when it is compressed to a pressure of  $42 \text{ bar}$ .

#### 4.1.1 Given:

The compression (or expansion) follows the law:

$$p_1 V_1 = p_2 V_2$$

#### 4.1.2 Final Volume of Gas

Given:  $p_1 = 120 \text{ kN/m}^2$ ,  $V_1 = 1 \text{ m}^3$ ,  $p_2 = 960 \text{ kN/m}^2$

From  $p_1 V_1 = p_2 V_2$ :

$$V_2 = \frac{p_1 V_1}{p_2}$$

Substitute:

$$V_2 = \frac{120 \times 1}{960} = 0.125 \text{ m}^3$$

**Final Volume**

$$V_2 = 0.125 \text{ m}^3$$

**4.1.3 Initial Volume of Gas**

Given:  $p_1 = 1.05 \text{ bar}$ ,  $V_2 = 5.6 \text{ m}^3$ ,  $p_2 = 42 \text{ bar}$

From  $p_1 V_1 = p_2 V_2$ :

$$V_1 = \frac{p_2 V_2}{p_1}$$

Substitute:

$$V_1 = \frac{42 \times 5.6}{1.05}$$

$$V_1 = 224 \text{ m}^3$$

**Initial Volume**

$$V_1 = 224 \text{ m}^3$$

**4.1.4 Code**

```
# Boyle's Law Calculator (p1 * V1 = p2 * V2)

print("Boyle's Law Calculator: p V = p V\n")

# Get known values (enter 0 for the unknown)
p1 = float(input("Enter initial pressure p (in kN/m² or bar): "))
V1 = float(input("Enter initial volume V (m³): "))
p2 = float(input("Enter final pressure p (in kN/m² or bar): "))
V2 = float(input("Enter final volume V (m³): "))

print()

# Determine which variable is missing (the one entered as 0)
if V2 == 0:
    V2 = (p1 * V1) / p2
    print(f"Final volume V = {V2:.4f} m³")

elif V1 == 0:
    V1 = (p2 * V2) / p1
    print(f"Initial volume V = {V1:.4f} m³")
```

```

elif p2 == 0:
    p2 = (p1 * V1) / V2
    print(f"Final pressure p = {p2:.4f}")

elif p1 == 0:
    p1 = (p2 * V2) / V1
    print(f"Initial pressure p = {p1:.4f}")

else:
    print("All variables entered - nothing to calculate!")

print("\nUnits:")
print("• If you used bar, results are in bar.")
print("• If you used kN/m², results are in kN/m².")

```

## 4.2 Compression — Stroke Volume and Final Temperature

A gas is compressed in a cylinder from  $p_1 = 1$  bar and  $T_1 = 35^\circ\text{C}$  at the beginning of the stroke to  $p_2 = 37$  bar at the end of the stroke. The clearance volume is  $V_c = 850\text{ cm}^3$ . The compression index is  $n = 1.32$ . Find the stroke (swept) volume  $V_s$  and the temperature at the end of compression  $T_2$ .

### 4.2.1 Solution

For a polytropic compression with  $pV^n = \text{constant}$  (here  $n = 1.32$  and with clearance  $V_c$ ):

Start of compression (BDC) volume:

$$V_1 = V_c + V_s$$

End of compression (TDC) volume:

$$V_2 = V_c$$

From  $(p_1 V_1^n = p_2 V_2^n)$  we get

$$\left(\frac{V_2}{V_1}\right)^n = \frac{p_1}{p_2}$$

so

$$\frac{V_c}{V_c + V_s} = \left(\frac{p_1}{p_2}\right)^{1/n}.$$

Rearrange to solve for  $V_s$ :

$$V_s = V_c \left[ \left( \frac{p_2}{p_1} \right)^{1/n} - 1 \right].$$

Substitute numbers (convert  $V_c$  to  $\text{m}^3$ :  $850 \text{ cm}^3 = 850 \times 10^{-6} \text{ m}^3 = 0.00085 \text{ m}^3$ ):

$$V_s = 0.00085 \left[ \left( \frac{37}{1} \right)^{1/1.32} - 1 \right]$$

Numerical evaluation:

$$V_s \approx 0.012255 \text{ m}^3 = 12.26 \text{ L}.$$

### 4.2.2 Final temperature

For a polytropic process the temperature ratio is

$$\frac{T_2}{T_1} = \left( \frac{p_2}{p_1} \right)^{\frac{n-1}{n}}.$$

Convert  $T_1$  to kelvin:  $T_1 = 35^\circ\text{C} = 308.15 \text{ K}$ .

$$T_2 = 308.15 \left( \frac{37}{1} \right)^{\frac{1.32-1}{1.32}}$$

Numerical evaluation:

$$T_2 \approx 739.49 \text{ K} = 466.34^\circ\text{C}.$$

### 4.2.3 Answers

- Stroke (swept) volume:  $V_s \approx 0.01226 \text{ m}^3$  (12.26 L)
- Temperature at end of compression:  $T_2 \approx 739.5 \text{ K} = 466.3^\circ\text{C}$ .

### 4.2.4 Code

```
# Given data
p1 = 1          # bar
p2 = 37         # bar
Vc_cm3 = 850    # clearance volume in cm3
n = 1.32        # compression index
T1_C = 35       # °C
```

#### 4.2. COMPRESSION — STROKE VOLUME AND FINAL TEMPERATURE 29

```
# Unit conversions
Vc = Vc_cm3 * 1e-6      # m³
T1 = T1_C + 273.15     # K

# Stroke volume calculation
Vs = Vc * ((p2 / p1)**(1/n) - 1)

# Final temperature calculation
T2 = T1 * (p2 / p1)**((n - 1) / n)
T2_C = T2 - 273.15

print(f"Stroke volume Vs = {Vs:.5f} m³ ({Vs*1000:.2f} L)")
print(f"Final temperature T2 = {T2:.2f} K ({T2_C:.2f} °C)")
```





# Chapter 5

## Ideal Cycles

### 5.1 Ideal Constant-Volume (Otto) Cycle

In an ideal constant-volume (Otto) cycle the temperature at the beginning of compression is  $50^\circ\text{C}$ . The volumetric compression ratio is  $r = 5 : 1$ . The heat supplied during the cycle is  $q_{in} = 930 \text{ kJ/kg}$  of working fluid. Take  $\gamma = 1.4$  and  $c_v = 0.717 \text{ kJ/kg} \cdot \text{K}$ .

Calculate:

- A. The maximum temperature attained in the cycle.
- B. The work done during the cycle per kg of working fluid.
- C. The ideal thermal efficiency of the cycle.

#### 5.1.1 Theory and formulas

For an ideal Otto cycle (constant-volume heat addition) the relevant steps and formulas are:

1. Isentropic compression  $1 \rightarrow 2$ :

$$T_2 = T_1 r^{\gamma-1}$$

2. Constant-volume heat addition  $2 \rightarrow 3$ :

$$q_{in} = c_v(T_3 - T_2) \quad \Rightarrow \quad T_3 = T_2 + \frac{q_{in}}{c_v}$$

3. Isentropic expansion  $3 \rightarrow 4$ :

$$T_4 = T_3 r^{1-\gamma}$$

4. Heat rejected:

$$q_{out} = c_v(T_4 - T_1)$$

5. Net work per unit mass:

$$w = q_{in} - q_{out}$$

6. Thermal efficiency:

$$\eta = \frac{w}{q_{in}} = 1 - \frac{q_{out}}{q_{in}}$$

(For the ideal Otto cycle the formula reduces to the well-known expression  $\eta = 1 - r^{1-\gamma}$ , which provides a check.)

### 5.1.2 Code

```
# Otto-cycle calculation (per kg)

# Given data
r = 5.0          # compression ratio
gamma = 1.4
cv = 0.717       # kJ/kg·K
T1_C = 50.0      # °C
q_in = 930.0     # kJ/kg

# Conversions
T1 = T1_C + 273  # K

# Step 1: T2 (after isentropic compression)
T2 = T1 * r**(gamma - 1)

# Step 2: T3 (after constant-volume heat addition) -> Tmax
T3 = T2 + q_in / cv

# Step 3: T4 (after isentropic expansion)
T4 = T3 * r**(1 - gamma)

# Heat rejected
q_out = cv * (T4 - T1)

# Net work per kg
w_net = q_in - q_out

# Efficiency
eta = w_net / q_in

# Print results
```

```

print(f"T1 = {T1:.2f} K ({T1 - 273:.2f} °C)")
print(f"T2 = {T2:.2f} K ({T2 - 273:.2f} °C)")
print(f"T3 = {T3:.2f} K ({T3 - 273:.2f} °C) <-- Tmax")
print(f"T4 = {T4:.2f} K ({T4 - 273:.2f} °C)")
print()
print(f"q_in = {q_in:.2f} kJ/kg")
print(f"q_out = {q_out:.2f} kJ/kg")
print(f"w_net = {w_net:.2f} kJ/kg")
print(f"efficiency = {eta*100:.2f} %")

```

### 5.1.3 Using Actual Volume Ratios Instead of Direct $r$

In the ideal Otto cycle we have written:

$$T_2 = T_1 r^{\gamma-1},$$

with  $r = V_1/V_2$ . Here we make the volume-dependence explicit using  $V_1$  and  $V_2$ .

#### 5.1.4 Given

- $T_1 = 50^\circ\text{C} = 323\text{ K}$
- Volumetric compression ratio  $r = \frac{V_1}{V_2} = 5$  (we'll set  $V_1 = 5$ ,  $V_2 = 1$  so  $V_1/V_2 = 5$ )
- Heat supplied  $q_{in} = 930\text{ kJ/kg}$
- $\gamma = 1.4$ ,  $c_v = 0.717\text{ kJ/kg} \cdot \text{K}$

#### 5.1.5 Volume-explicit relations

Isentropic compression (1→2) (using volumes):

$$\frac{T_2}{T_1} = \left(\frac{V_1}{V_2}\right)^{\gamma-1} \Rightarrow T_2 = T_1 \left(\frac{V_1}{V_2}\right)^{\gamma-1}.$$

Constant-volume heat addition (2→3):

$$T_3 = T_2 + \frac{q_{in}}{c_v}.$$

Isentropic expansion (3→4) (volume ratio inverted):

$$\frac{T_4}{T_3} = \left(\frac{V_2}{V_1}\right)^{\gamma-1} \Rightarrow T_4 = T_3 \left(\frac{V_2}{V_1}\right)^{\gamma-1}.$$

Heat rejected:

$$q_{out} = c_v(T_4 - T_1),$$

Net work and efficiency:

$$w = q_{in} - q_{out}, \quad \eta = \frac{w}{q_{in}}.$$

### 5.1.6 Code

```
# Otto-cycle calculation using explicit volumes (per kg)

# Given data
V1 = 5.0          # arbitrary volume units (e.g. litres) at start of compression (BDC)
V2 = 1.0          # arbitrary volume units at end of compression (TDC)
gamma = 1.4
cv = 0.717        # kJ/kg·K
T1_C = 50.0       # °C
q_in = 930.0      # kJ/kg

# Conversions
T1 = T1_C + 273   # K

# Isentropic compression using volume ratio V1/V2
T2 = T1 * (V1 / V2)**(gamma - 1)

# Constant-volume heat addition
T3 = T2 + q_in / cv

# Isentropic expansion using inverted volume ratio V2/V1
T4 = T3 * (V2 / V1)**(gamma - 1)

# Heat rejected, net work and efficiency
q_out = cv * (T4 - T1)
w_net = q_in - q_out
eta = w_net / q_in

# Print results
print("--- Inputs ---")
print(f"V1 = {V1:.3f} (units), V2 = {V2:.3f} (units) --> V1/V2 = {V1/V2:.3f}")
print(f"T1 = {T1:.2f} K ({T1-273:.2f} °C)")
print()
print("--- Results ---")
print(f"T2 = {T2:.2f} K ({T2-273:.2f} °C)")
print(f"T3 = {T3:.2f} K ({T3-273:.2f} °C) <-- Tmax")
print(f"T4 = {T4:.2f} K ({T4-273:.2f} °C)")
```

```
print()
print(f"q_in = {q_in:.2f} kJ/kg")
print(f"q_out = {q_out:.2f} kJ/kg")
print(f"w_net = {w_net:.2f} kJ/kg")
print(f"efficiency = {eta*100:.2f} %")
```

## 5.2 Ideal Constant-Pressure (Diesel) Cycle

Given

- $T_1 = 50^\circ\text{C} = 323\text{ K}$
- Compression ratio  $r = \frac{V_1}{V_2} = 5$
- Heat supplied  $q_{in} = 930\text{ kJ/kg}$  (added at constant pressure)
- $\gamma = 1.4$ ,  $c_v = 0.717\text{ kJ/kg} \cdot \text{K}$

**Notes:** For constant-pressure heat addition (2→3),

$$q_{in} = c_p (T_3 - T_2), \quad c_p = c_v \gamma.$$

The cut-off (volume) ratio is  $\rho = \frac{V_3}{V_2} = \frac{T_3}{T_2}$  because  $p_2 = p_3$

### 5.2.1 Code

```
# Diesel-cycle calculation (per kg) - runnable in Quarto

import numpy as np
import matplotlib.pyplot as plt

# Given
r = 5.0          # V1 / V2
gamma = 1.4
cv = 0.717       # kJ/kg·K
cp = cv * gamma  # kJ/kg·K
T1 = 323.0       # K (simplified 273 + 50)
q_in = 930.0     # kJ/kg

# State 2: isentropic compression 1->2
T2 = T1 * r**(gamma - 1)

# State 3: constant-pressure heat addition 2->3
# q_in = cp * (T3 - T2)  ->  T3 = T2 + q_in / cp
```

```

T3 = T2 + q_in / cp

# Cut-off ratio (V3/V2) since p2 = p3
rho = T3 / T2

# State 4: isentropic expansion 3->4 (V4 = V1)
# T4 = T3 * (V3/V4)^(gamma-1) = T3 * (rho / r)^(gamma-1)
T4 = T3 * (rho / r)**(gamma - 1)

# Heat rejected, net work and efficiency
q_out = cv * (T4 - T1)
w_net = q_in - q_out
eta = w_net / q_in

# Pressures (normalized) using ideal-gas relation P ~ T/V
P1 = 1.0
P2 = P1 * (T2/T1) * (V1 := r) / (V2 := 1.0) # V1=r, V2=1 chosen for explicitness
P3 = P2
V3 = rho * V2
P4 = (T4 / T1) * (V1 / (V1)) * P1 # alternatively compute via ideal gas: P4 = P3 * (T4/T3)

# More consistent P4 via ideal gas:
P4 = P3 * (T4 / T3) * (V3 / V1)

# Curves for PV diagram
V_compression = np.linspace(V1, V2, 200) # 1 -> 2 (compression)
P_compression = P1 * (V1 / V_compression)**gamma

V_expansion = np.linspace(V3, V1, 200) # 3 -> 4 (expansion)
P_expansion = P3 * (V3 / V_expansion)**gamma

plt.figure(figsize=(8,5))
plt.plot(V_compression, P_compression, label="Isentropic compression (1-2)")
plt.plot([V2, V3], [P2, P3], 'r-', linewidth=2, label="Constant-pressure heat addition")
plt.plot(V_expansion, P_expansion, label="Isentropic expansion (3-4)")
plt.plot([V1, V1], [P4, P1], 'r-', linewidth=2, label="Constant-volume heat rejection")
# mark state points
plt.plot([V1, V2, V3, V1], [P1, P2, P3, P4], 'ko')
for (V,P,label) in [(V1,P1,"1"), (V2,P2,"2"), (V3,P3,"3"), (V1,P4,"4")]:
    plt.text(V, P*1.02, label, ha='center')

plt.xlabel("Volume (arbitrary units)")
plt.ylabel("Pressure (arbitrary units)")
plt.title("Ideal Diesel Cycle - P-V Diagram (normalized)")
plt.legend()

```

```
plt.grid(True)
plt.tight_layout()
plt.show()

# Print numeric results
print("---- Diesel Cycle Results (per kg) ----")
print(f"T1 = {T1:.2f} K")
print(f"T2 = {T2:.2f} K")
print(f"T3 = {T3:.2f} K (Tmax after constant-pressure heat addition)")
print(f"T4 = {T4:.2f} K")
print()
print(f"Cut-off ratio ( $\rho = V3/V2$ ) = {rho:.4f}")
print(f"q_in = {q_in:.2f} kJ/kg")
print(f"q_out = {q_out:.2f} kJ/kg")
print(f"Net work w = {w_net:.2f} kJ/kg")
print(f"Ideal thermal efficiency = {eta*100:.2f} %")
```





## 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/meter <sup>2</sup>	5 m/s <sup>2</sup> , 5 kg/m <sup>2</sup>	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 <sup>-1</sup> , 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 <sup>-1</sup> ) to (10 <sup>-9</sup> ), uppercase for (10 <sup>6</sup> ) 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
$\alpha$	A	alpha
$\beta$	B	beta
$\gamma$	$\Gamma$	gamma
$\delta$	$\Delta$	delta
$\epsilon$	E	epsilon
$\zeta$	Z	zeta
$\eta$	E	eta
$\theta$	$\Theta$	theta
$\iota$	I	iota
$\kappa$	K	kappa
$\lambda$	$\Lambda$	lambda
$\mu$	M	mu
$\nu$	N	nu
$\xi$	$\Xi$	xi
$o$	O	omicron
$\pi$	$\Pi$	pi
$\rho$	P	rho
$\sigma$	$\Sigma$	sigma
$\tau$	T	tau
$v$	$\Upsilon$	upsilon
$\phi$	$\Phi$	phi
$\chi$	X	chi

Lower Case	Upper Case	Name
$\psi$	$\Psi$	psi
$\omega$	$\Omega$	omega

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

Symbol	Name	Use	Course
$\Delta$	Delta	Change	Thermodynamics
$\Delta$	Delta	Displacement	Naval Architecture
$\nabla$	Nabla	Volume	Naval Architecture
$\Sigma$	Sigma	Sum	Thermodynamics, Naval Architecture, Applied Mechanics
$\sigma$	Sigma	Stress	Thermodynamics, Applied Mechanics
$\epsilon$	Epsilon	Modulus of elasticity	Thermodynamics, Applied Mechanics
$\eta$	Eta	Efficiency	Thermodynamics
$\omega$	Omega	Angular velocity	Thermodynamics, Applied Mechanics
$\rho$	Rho	Density	Thermodynamics, Naval Architecture
$\tau$	Tau	Torque	Thermodynamics, Applied Mechanics

# Index

colon, 3  
conditional statements, 6  
control flow, 6  
  
def keyword, 7  
  
for loop, 7  
  
indentation, 3  
  
lambda keyword, 8  
  
math module, 8  
  
print, 4  
  
variables, 4  
  
while loop, 7