04 Software

SrWL Learning · Complete Classroom Teaching Version · 2026–2028 target

From operating systems to translators

55 editable slides / Explanation · Visual · Practice · Teacher Check

By the end of this chapter

- System and application software.
- Operating systems.
- Interrupts and utilities.
- Language translators.

Learning goals are organised into bite-sized lessons.

Our learning route

10 mini-lessons · each includes one guided check

01 System vs application software: A

02 System vs application software: B

03 Operating system functions: A

04 Operating system functions: B

05 Interrupts and utilities: A

06 Interrupts and utilities: B

07 High-level and low-level languages: A

08 High-level and low-level languages: B

09 Compilers and interpreters: A

10 Compilers and interpreters: B

01 System vs application software · Foundations

One clear idea to explain before moving to practice

System software supports computer operation.

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

System vs application software — key points

- System software supports computer operation.
- Application software performs tasks for users.

Worked context: Ask: Who primarily needs the service—the system or the user?

System vs application software — connect the ideas

Follow the concept from left to right

01

System software supports computer operation.

→

02

Application software performs tasks for users.

Apply the principles to this example: Ask: Who primarily needs the service—the system or the user?

Try it · System vs application software

Think (30 s) · Pair discussion (1 min) · Share

Explain the meaning of this idea and give a relevant example:

Application software performs tasks for users.

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

Teacher check · System vs application software

Model response · Correct misconceptions

Application software performs tasks for users.

EXAM TIP / EXTENSION

Use this worked context to add detail: Ask: Who primarily needs the service—the system or the user?

02 System vs application software · Applications

One clear idea to explain before moving to practice

Operating systems, utilities and translators are system software.

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

System vs application software — key points

- Operating systems, utilities and translators are system software.
- A word processor or graphics editor is application software.

Worked context: Ask: Who primarily needs the service—the system or the user?

System vs application software — connect the ideas

Follow the concept from left to right

01

Operating systems, utilities and translators are system software.

→

02

A word processor or graphics editor is application software.

Apply the principles to this example: Ask: Who primarily needs the service—the system or the user?

Try it · System vs application software

Think (30 s) · Pair discussion (1 min) · Share

Describe when this principle applies:

A word processor or graphics editor is application software.

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

Teacher check · System vs application software

Model response · Correct misconceptions

A word processor or graphics editor is application software.

EXAM TIP / EXTENSION

Use this worked context to add detail: Ask: Who primarily needs the service—the system or the user?

03 Operating system functions · Foundations

One clear idea to explain before moving to practice

Provide a user interface.

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

Operating system functions — key points

- Provide a user interface.
- Manage memory, processes and peripherals.

Worked context: An OS allocates resources between running programs.

Operating system functions — connect the ideas

Follow the concept from left to right

01

Provide a user interface.

→

02

Manage memory, processes and peripherals.

Apply the principles to this example: An OS allocates resources between running programs.

Try it · Operating system functions

Think (30 s) · Pair discussion (1 min) · Share

Identify the key technical terms and describe their roles: Manage memory, processes and peripherals.

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

Teacher check · Operating system functions

Model response · Correct misconceptions

Manage memory, processes and peripherals.

EXAM TIP / EXTENSION

Use this worked context to add detail: An OS allocates resources between running programs.

04 Operating system functions · Applications

One clear idea to explain before moving to practice

Manage files and access permissions.

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

Operating system functions — key points

- Manage files and access permissions.
- Schedule tasks and support multitasking.

Worked context: An OS allocates resources between running programs.

Operating system functions — connect the ideas

Follow the concept from left to right

01

Manage files and access permissions.

→

02

Schedule tasks and support multitasking.

Apply the principles to this example: An OS allocates resources between running programs.

Try it · Operating system functions

Think (30 s) · Pair discussion (1 min) · Share

Compare this concept with a related alternative:
Schedule tasks and support multitasking.

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

Teacher check · Operating system functions

Model response · Correct misconceptions

Schedule tasks and support multitasking.

EXAM TIP / EXTENSION

Use this worked context to add detail: An OS allocates resources between running programs.

05 Interrupts and utilities · Foundations

One clear idea to explain before moving to practice

An interrupt alerts the processor to an event.

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

Interrupts and utilities — key points

- An interrupt alerts the processor to an event.
- An interrupt handler responds to the event.

Worked context: An interrupt can be triggered by an input device.

Interrupts and utilities — connect the ideas

Follow the concept from left to right

01

An interrupt alerts the processor to an event.

→

02

An interrupt handler responds to the event.

Apply the principles to this example: An interrupt can be triggered by an input device.

Try it · Interrupts and utilities

Think (30 s) · Pair discussion (1 min) · Share

Apply this idea to a new computer science scenario:
An interrupt handler responds to the event.

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

Teacher check · Interrupts and utilities

Model response · Correct misconceptions

An interrupt handler responds to the event.

EXAM TIP / EXTENSION

Use this worked context to add detail: An interrupt can be triggered by an input device.

06 Interrupts and utilities · Applications

One clear idea to explain before moving to practice

Utilities support maintenance and security.

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

Interrupts and utilities — key points

- Utilities support maintenance and security.
- Examples include backups, encryption and disk tools.

Worked context: An interrupt can be triggered by an input device.

Interrupts and utilities — connect the ideas

Follow the concept from left to right

01

Utilities support maintenance and security.

→

02

Examples include backups, encryption and disk tools.

Apply the principles to this example: An interrupt can be triggered by an input device.

Try it · Interrupts and utilities

Think (30 s) · Pair discussion (1 min) · Share

Explain the meaning of this idea and give a relevant example:

Examples include backups, encryption and disk tools.

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

Teacher check · Interrupts and utilities

Model response · Correct misconceptions

Examples include backups, encryption and disk tools.

EXAM TIP / EXTENSION

Use this worked context to add detail: An interrupt can be triggered by an input device.

07 High-level and low-level languages · Foundations

One clear idea to explain before moving to practice

High-level language is more readable to programmers.

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

High-level and low-level languages — key points

- High-level language is more readable to programmers.
- Assembly language uses mnemonics.

Worked context: One high-level statement can translate into many machine instructions.

High-level and low-level languages — connect the ideas

Follow the concept from left to right

01

High-level language is more readable to programmers.

→

02

Assembly language uses mnemonics.

Apply the principles to this example: One high-level statement can translate into many machine instructions.

Try it · High-level and low-level languages

Think (30 s) · Pair discussion (1 min) · Share

Describe when this principle applies:

Assembly language uses mnemonics.

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

Teacher check · High-level and low-level languages

Model response · Correct misconceptions

Assembly language uses mnemonics.

EXAM TIP / EXTENSION

Use this worked context to add detail: One high-level statement can translate into many machine instructions.

08 High-level and low-level languages · Applications

One clear idea to explain before moving to practice

Machine code is executable instructions in binary.

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

High-level and low-level languages — key points

- Machine code is executable instructions in binary.
- Language level affects portability and control.

Worked context: One high-level statement can translate into many machine instructions.

High-level and low-level languages — connect the ideas

Follow the concept from left to right

01

Machine code is executable instructions in binary.

→

02

Language level affects portability and control.

Apply the principles to this example: One high-level statement can translate into many machine instructions.

Try it · High-level and low-level languages

Think (30 s) · Pair discussion (1 min) · Share

Identify the key technical terms and describe their roles: Language level affects portability and control.

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

Teacher check · High-level and low-level languages

Model response · Correct misconceptions

Language level affects portability and control.

EXAM TIP / EXTENSION

Use this worked context to add detail: One high-level statement can translate into many machine instructions.

09 Compilers and interpreters · Foundations

One clear idea to explain before moving to practice

A compiler translates a program before execution.

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

Compilers and interpreters — key points

- A compiler translates a program before execution.
- An interpreter translates and executes as it proceeds.

Worked context: Class discussion: compiler vs interpreter benefits.

Compilers and interpreters — connect the ideas

Follow the concept from left to right

01

A compiler translates a program before execution.

→

02

An interpreter translates and executes as it proceeds.

Apply the principles to this example: Class discussion: compiler vs interpreter benefits.

Try it · Compilers and interpreters

Think (30 s) · Pair discussion (1 min) · Share

Compare this concept with a related alternative:
An interpreter translates and executes as it proceeds.

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

Teacher check · Compilers and interpreters

Model response · Correct misconceptions

An interpreter translates and executes as it proceeds.

EXAM TIP / EXTENSION

Use this worked context to add detail: Class discussion: compiler vs interpreter benefits.

10 Compilers and interpreters · Applications

One clear idea to explain before moving to practice

Both detect some language errors.

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

Compilers and interpreters — key points

- Both detect some language errors.
- An IDE supports editing, running and debugging.

Worked context: Class discussion: compiler vs interpreter benefits.

Compilers and interpreters — connect the ideas

Follow the concept from left to right

01

Both detect some language errors.

→

02

An IDE supports editing, running and debugging.

Apply the principles to this example: Class discussion: compiler vs interpreter benefits.

Try it · Compilers and interpreters

Think (30 s) · Pair discussion (1 min) · Share

Apply this idea to a new computer science scenario:
An IDE supports editing, running and debugging.

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

Teacher check · Compilers and interpreters

Model response · Correct misconceptions

An IDE supports editing, running and debugging.

EXAM TIP / EXTENSION

Use this worked context to add detail: Class discussion: compiler vs interpreter benefits.

Five essential ideas

Recall these without looking at the teaching slides

01 System vs application software

02 Operating system functions

03 Interrupts and utilities

04 High-level and low-level languages

05 Compilers and interpreters

End-of-chapter challenge

Answer in your notebook, then compare with a partner

Give two operating system functions and one application software example.

TEACHER MODEL ANSWER

Examples: memory management, file management; spreadsheet application.