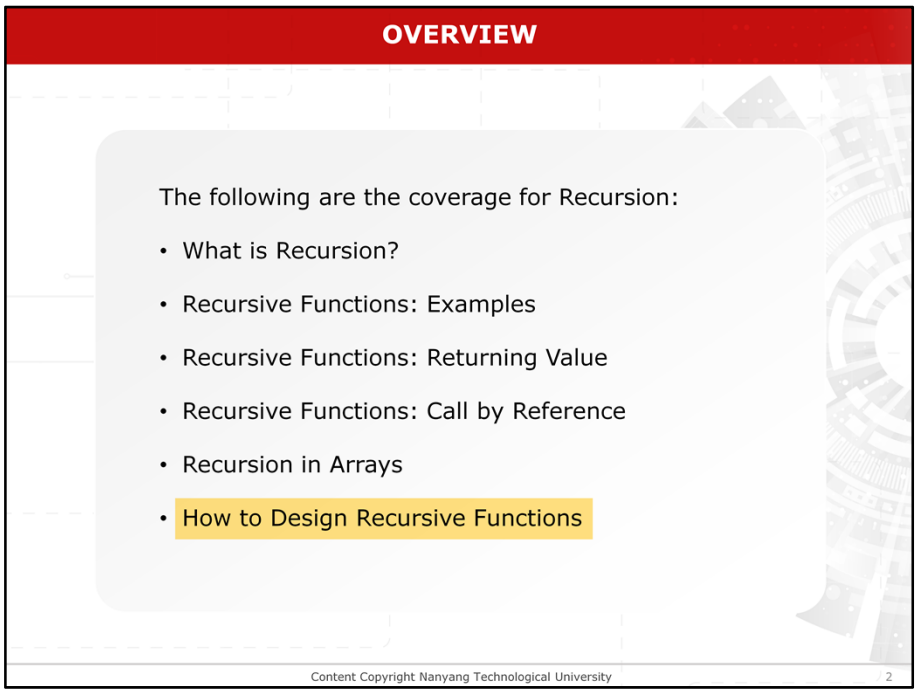
NANYANG TECHNOLOGICAL UNIVERSITY

CE1007/ CZ1007 DATA STRUCTURES

Lesson 10.6 How to Design Recursive Functions?

Assoc Prof Hui Siu Cheung

College of Engineering
School of Computer Science and Engineering



There are 6 main sections to cover for Recursion as shown. This video lesson focuses on the last part on

LESSON OBJECTIVES

After this lesson, you should be able to:

- Explain how to design recursive functions

 3

After this lesson, you should be able to explain

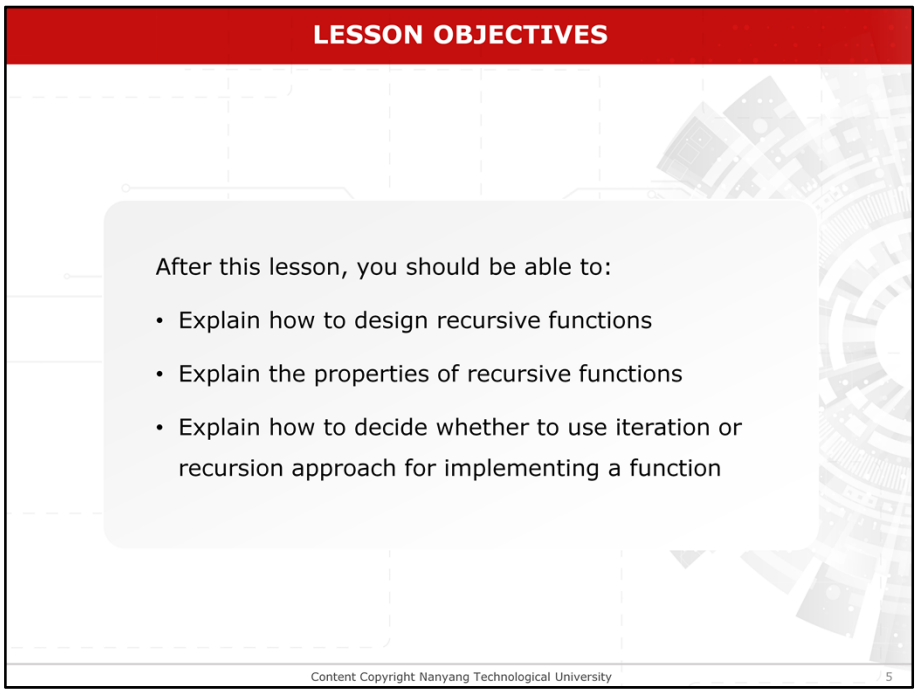
LESSON OBJECTIVES

After this lesson, you should be able to:

- Explain how to design recursive functions
- Explain the properties of recursive functions

 4

Explain the properties of recursive functions



Explain how to decide whether to use iteration or recursion approach for implementing a function

HOW TO DESIGN RECURSIVE FUNCTIONS?

There are 4 steps involved when you design any recursive functions:

HOW TO DESIGN RECURSIVE FUNCTIONS?

1. Find the **key step** (recursive condition)
   - How can the problem be divided into parts?
   - How will the key step in the middle be done?

 7

First, find the key step, the recursive condition.

Decide how can the problem be divided into parts, and how will the key step in the middle be done.

HOW TO DESIGN RECURSIVE FUNCTIONS?

1. Find the **key step** (recursive condition)
   - How can the problem be divided into parts?
   - How will the key step in the middle be done?
2. Find a **stopping rule** (terminating condition)
   - Small, special case that is trivial or easy to handle without recursion

 8

Second step is to find a stopping rule that is the terminating condition.

Decide the terminating condition which should be small and special case that is trivial or easy to handle without recursion.

HOW TO DESIGN RECURSIVE FUNCTIONS?

1. Find the **key step** (recursive condition)
   - How can the problem be divided into parts?
   - How will the key step in the middle be done?
2. Find a **stopping rule** (terminating condition)
   - Small, special case that is trivial or easy to handle without recursion
3. Outline your algorithm
   - Combine the stopping rule and the key step, using an **if-else** statement to select between them

 9

The third step is to outline your algorithm.

Combine the stopping rule and the key step, and use an **if-else** statement to select between them.

HOW TO DESIGN RECURSIVE FUNCTIONS?

1. Find the **key step** (recursive condition)
   - How can the problem be divided into parts?
   - How will the key step in the middle be done?
2. Find a **stopping rule** (terminating condition)
   - Small, special case that is trivial or easy to handle without recursion
3. Outline your algorithm
   - Combine the stopping rule and the key step, using an **if-else** statement to select between them
4. Check termination
   - Verify recursion always terminates (it is necessary to make sure that the function will also terminate)

 10

The last step is to check termination.

Verify the recursion always terminates. It is necessary to make sure that the function will also terminate.

PROPERTIES OF RECURSIVE FUNCTIONS

Properties of recursive functions.

From the recursive function examples covered so far, we have observed the following properties of recursive functions:

PROPERTIES OF RECURSIVE FUNCTIONS

- Each function has a terminating condition where no more call to it will be made

Each function has a terminating condition where no more call to it will be made

PROPERTIES OF RECURSIVE FUNCTIONS

- Each function has a terminating condition where no more call to it will be made
- Each function makes a call to itself with an argument, which is closer to the terminating condition

 13

Each function makes a call to itself with an argument, which is closer to the terminating condition.

PROPERTIES OF RECURSIVE FUNCTIONS

- Each function has a terminating condition where no more call to it will be made
- Each function makes a call to itself with an argument, which is closer to the terminating condition
- Each level of the function call has its own arguments

Each level of the function call has its own arguments

PROPERTIES OF RECURSIVE FUNCTIONS

- Each function has a terminating condition where no more call to it will be made
- Each function makes a call to itself with an argument, which is closer to the terminating condition
- Each level of the function call has its own arguments
- When a recursive call is made, control is transferred from the calling point to the first statement of the recursive function. When a call at a certain level is finished, control returns to the calling point one level up

 15

When a recursive call is made, control is transferred from the calling point to the first statement of the recursive function. When a call at a certain level is finished, control returns to the calling point one level up

RECURSION OR ITERATION?

Recursion or Iteration?

How can one decide whether to use iteration or recursion approach for implementing a function?

RECURSION OR ITERATION?

Advantage of using recursive functions:

when the problem is recursive in nature, a recursive function results in **short**, **clear** code

 17

The main advantage of using recursive functions is that when the problem is recursive in nature, a recursive function results in shorter and clearer code.

RECURSION OR ITERATION?

Advantage of using recursive functions:

when the problem is recursive in nature, a recursive function results in **short**, **clear** code

Disadvantage of using recursive functions:

recursion is more **expensive** (computationally) than iteration

 18

However, the main disadvantage of using recursive functions is that recursion is more expensive than iteration in terms of memory usage.

RECURSION OR ITERATION?

Any problem that can be solved recursively can also be solved iteratively (by using **loops**).

 19

Any problems that can be solved recursively can also be solved iteratively (by using loops).

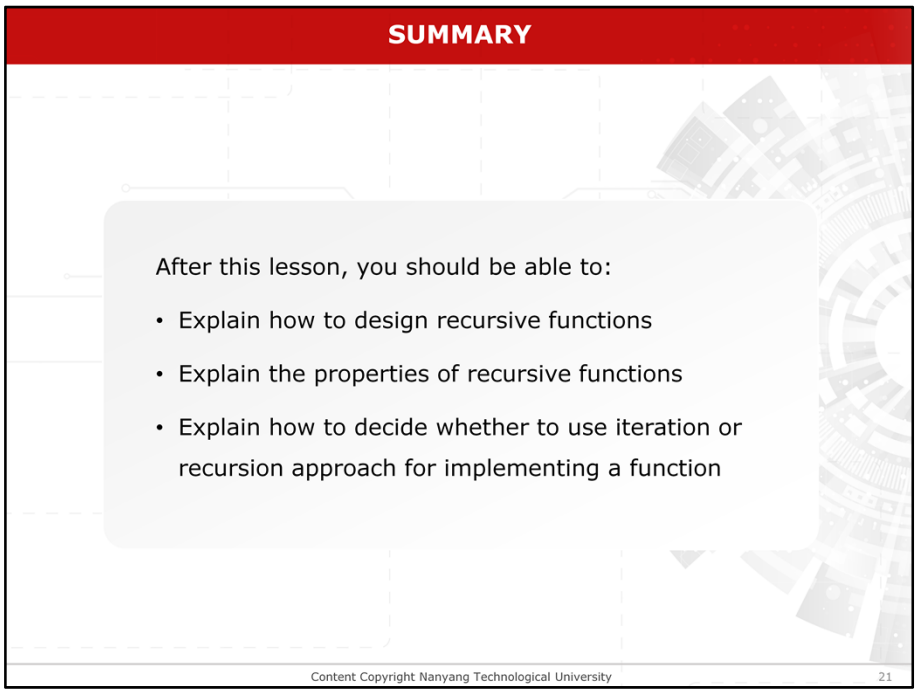
RECURSION OR ITERATION?

Any problem that can be solved recursively can also be solved iteratively (by using **loops**).

A recursive approach is normally chosen in preference to an iterative approach when the recursive approach more **naturally** mirrors the problem and the results in a program that is easier to understand and debug.

 20

A recursive approach is generally chosen over an iterative approach when the recursive approach can mirror the problem more naturally which results in a program that is easier to understand and debug.



In summary, after viewing this video lesson, you should be able to do the points listed.