

Review

- Abstraction
- Expression and evaluation
- Operator and operand
- Value and type
- Name and variable
- Assignment
- Memory model

Functions

Lecture 2-1

Hyung-Sin Kim



SNU Graduate School of Data Science

Defining Our Own Functions

- Python does provide useful built-in functions (e.g., max, min...), but they are not enough
- For example, assume that we want to convert celsius degree to fahrenheit degree
 - `convert_to_fahrenheit(10)`
 - We want to get 50 (Equation: $\text{fahrenheit} = \text{celsius} * 9/5 + 32$)
 - But we get an error when typing it, which means that there is **no function** named “convert_to_fahrenheit”


Defining Our Own Functions

- Let's make our own function “convert_to_fahrenheit”
 - `def convert_to_fahrenheit(celsius):`
 - `return celsius * 9 / 5 + 32`
- Function body is indented! Without the indentation, you will see an error
 - **Indentation** must be the same block of codes
- Let's do this again
 - `convert_to_fahrenheit(10)`
 - And change the argument freely!

Defining Our Own Functions – Exercise

- Define “convert_to_celsius(fahrenheit),” which converts fahrenheit degree to celsius degree
- Can someone screen-capture and share on the chat?

Defining Our Own Functions – Generalization

- `def <<function_name>> (<<parameters>>):`  function header
- `<<function_body>>`
- Parameters are variables
- Most functions have a return statement at the end of the function body
 - `return <<expression>>`
 - It evaluates the expression, produces a value, which is the result of the function call

Local Variables

- Implement “convert_to_fahrenheit” in a different way
 - `def convert_to_fahrenheit(celsius):`
 - `a = 9 / 5`
 - `b = 32`
 - `return celsius * a + b`
- **Local variables:** Variables created within a function
 - Parameters are also local variables
 - They are erased when the function returns (**cannot** be used outside of the function)

Namespace

- When Python executes a function call, it creates a **namespace** in which to store **local variables** for that call
- If a variable name in the namespace is same as a variable in another namespace, Python just considers **the current namespace!**

Memory Models for Function Calls

Lecture 2-2

Hyung-Sin Kim



SNU Graduate School of Data Science

So... what happens when you call a function?

Execute a Function Call

- `max(3+8, 5)`
 - Step 1: Evaluate the **arguments** left to right.
 - Step 2: Create a **namespace** to hold the function call's **local variables**, including the parameters.
 - Step 3: Pass the resulting **argument values** into the function by assigning them to the **parameters**
 - Step 4: Execute the **function body**. When a **return** statement is executed, the function terminates and the value of the expression in the return statement is used as the value of the function call

Memory Model for Function Call

- `def doubling(x):`
- `return 2*x`
- `x = 5`
- `x = doubling(x+5)`

frames

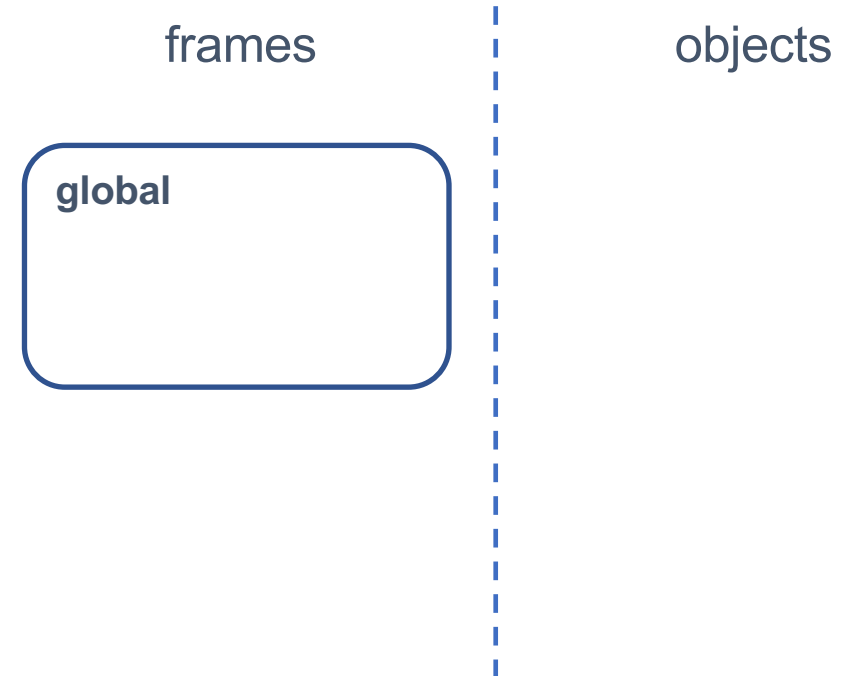
*Frames for
namespaces*

objects

Memory objects

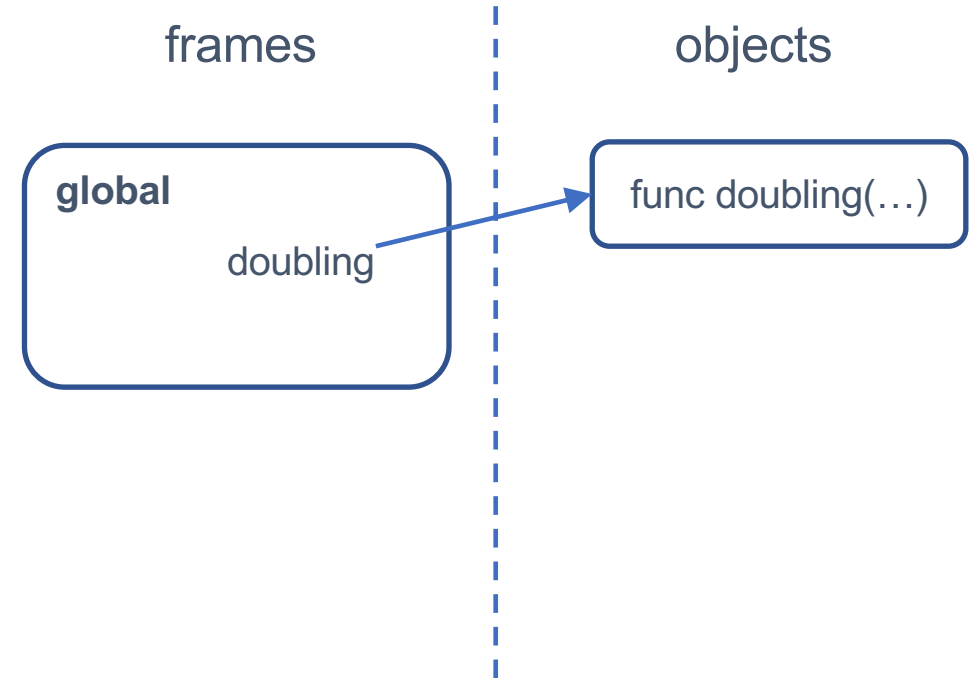
Memory Model for Function Call

- `def doubling(x):`
- `return 2*x`
-
- `x = 5`
- `x = doubling(x+5)`



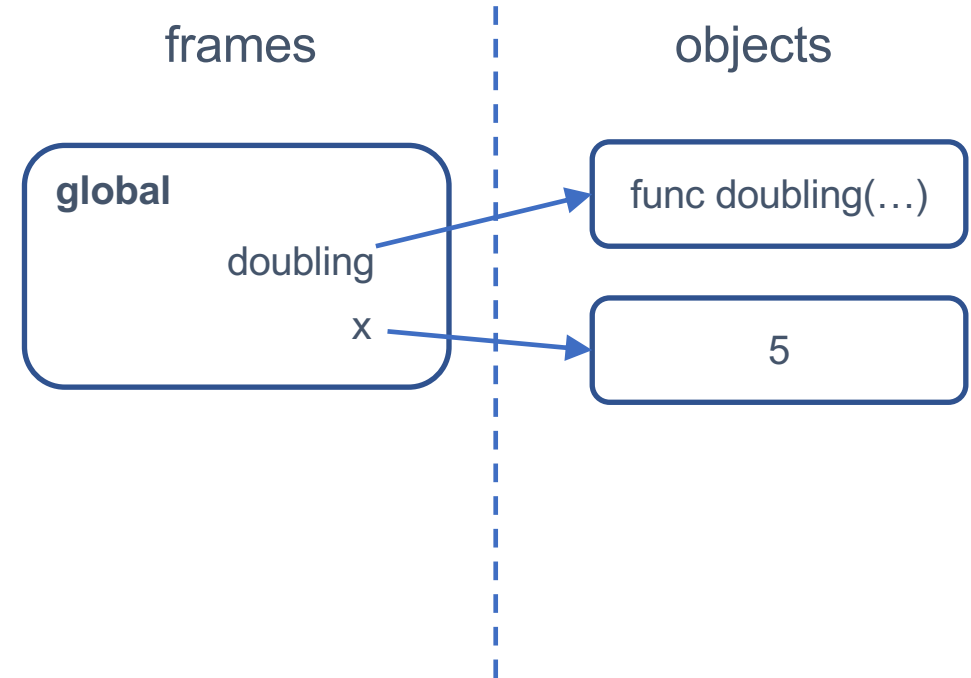
Memory Model for Function Call

- **def doubling(x):**
- **return 2*x**
-
- $x = 5$
- $x = \text{doubling}(x+5)$



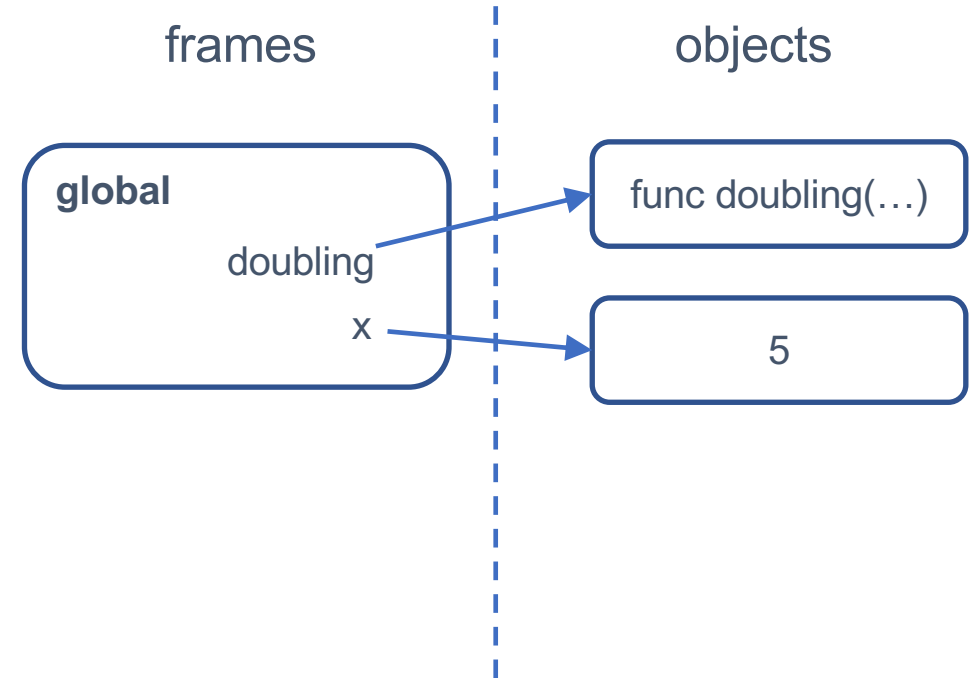
Memory Model for Function Call

- `def doubling(x):`
- `return 2*x`
-
- **`x = 5`**
- `x = doubling(x+5)`



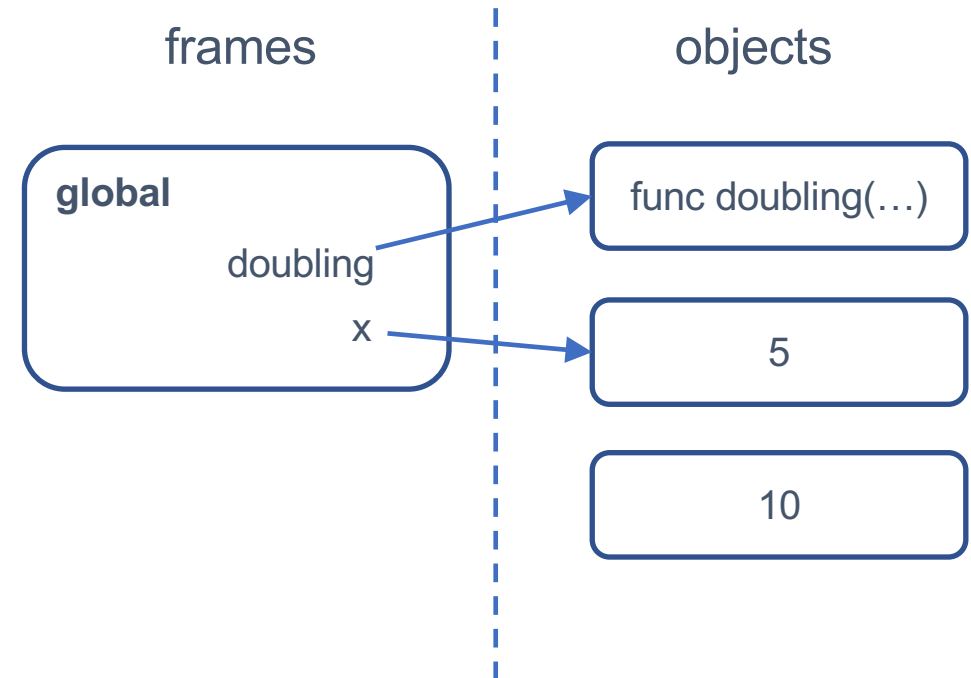
Memory Model for Function Call

- `def doubling(x):`
- `return 2*x`
-
- `x = 5`
- `x = doubling(x+5)`



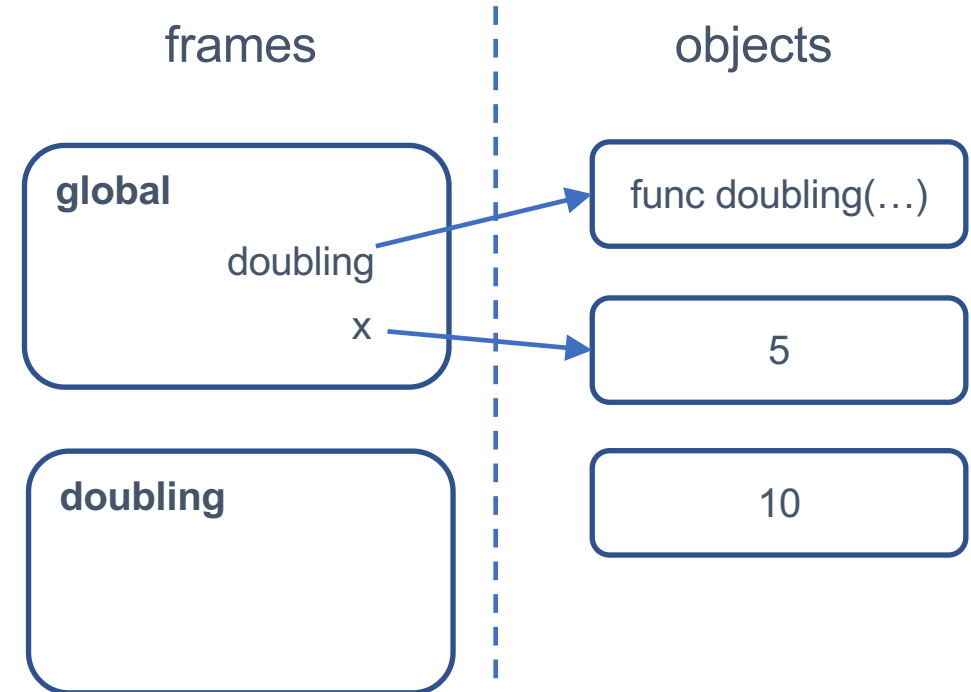
Memory Model for Function Call

- `def doubling(x):`
- `return 2*x`
-
- `x = 5`
- `x = doubling(x+5)`
 - Evaluate argument and get a value (10)
 - Using x in the global namespace



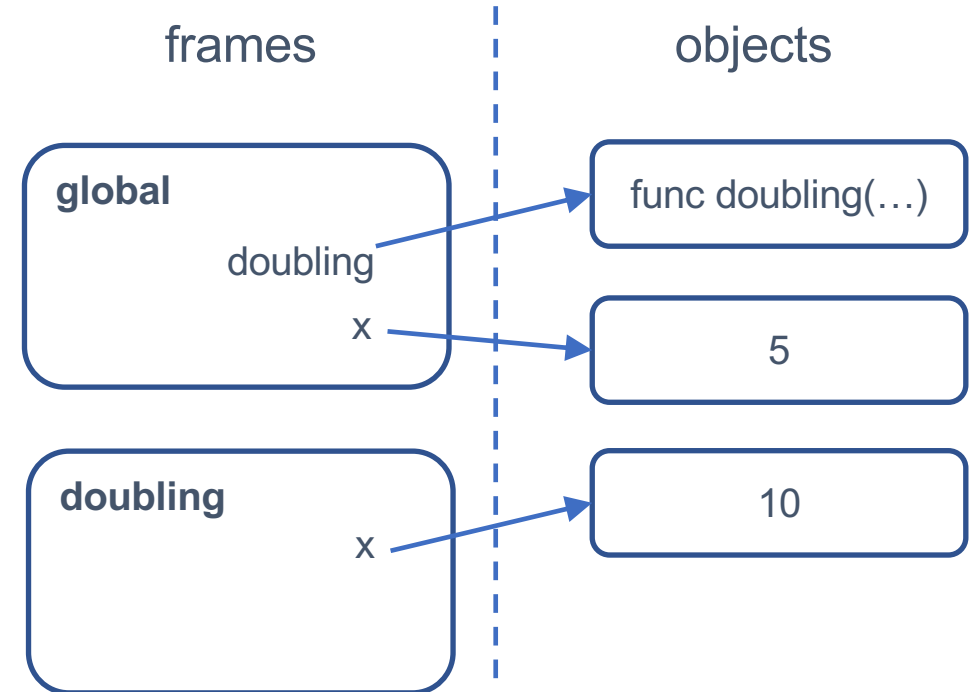
Memory Model for Function Call

- `def doubling(x):`
- `return 2*x`
-
- `x = 5`
- `x = doubling(x+5)`
 - Evaluate argument and get a value (10)
 - Using x in the global namespace
 - Creating a namespace



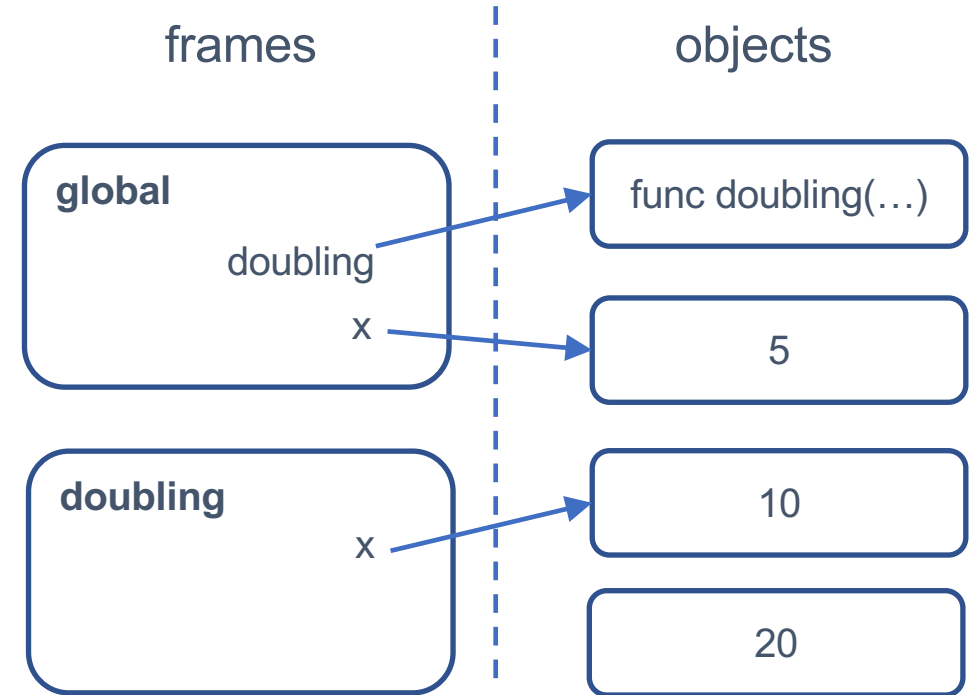
Memory Model for Function Call

- `def doubling(x):`
- `return 2*x`
-
- `x = 5`
- `x = doubling(x+5)`
 - Evaluate argument and get a value (10)
 - Using x in the global namespace
 - Creating a namespace
 - Assign the argument value to the function parameter



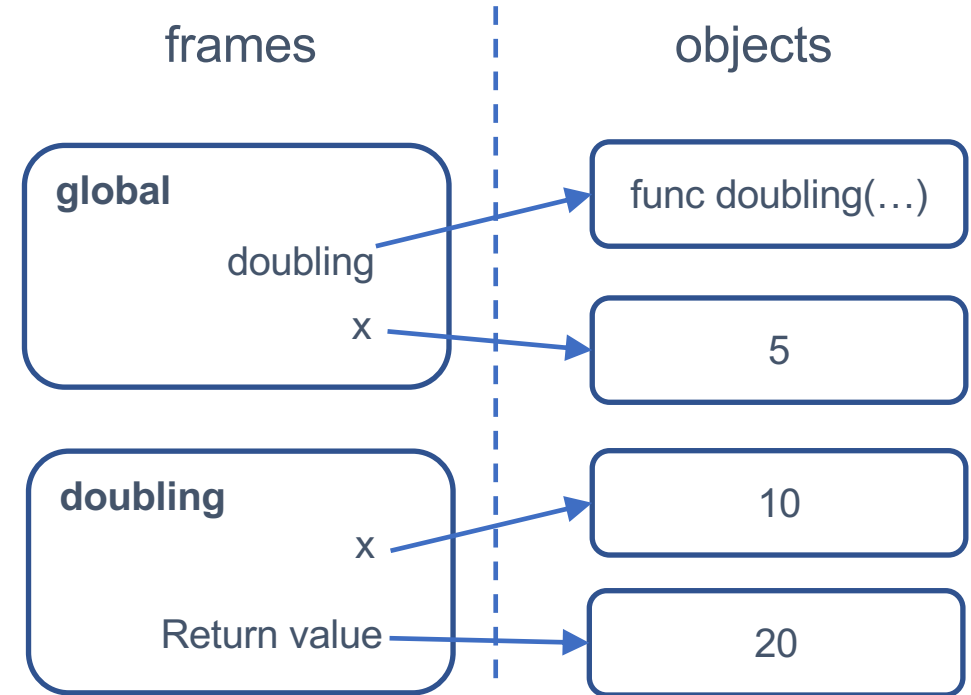
Memory Model for Function Call

- `def doubling(x):`
- `return 2*x`
-
- `x = 5`
- `x = doubling(x+5)`
 - Evaluate argument and get a value (10)
 - Using x in the global namespace
 - Creating a namespace
 - Assign the argument value to the function parameter
 - Get the return value of the function



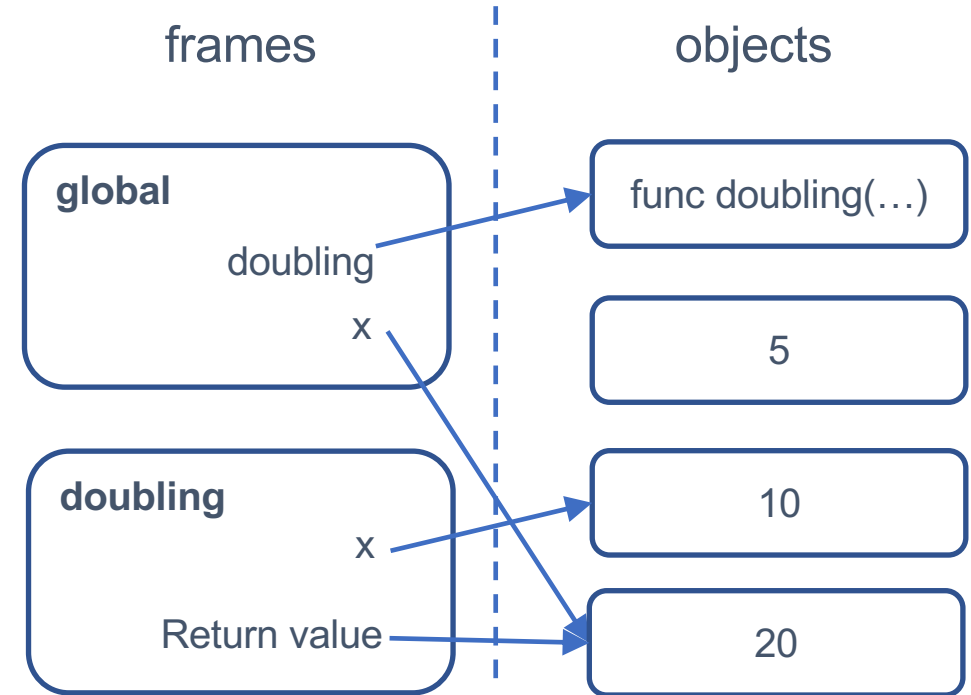
Memory Model for Function Call

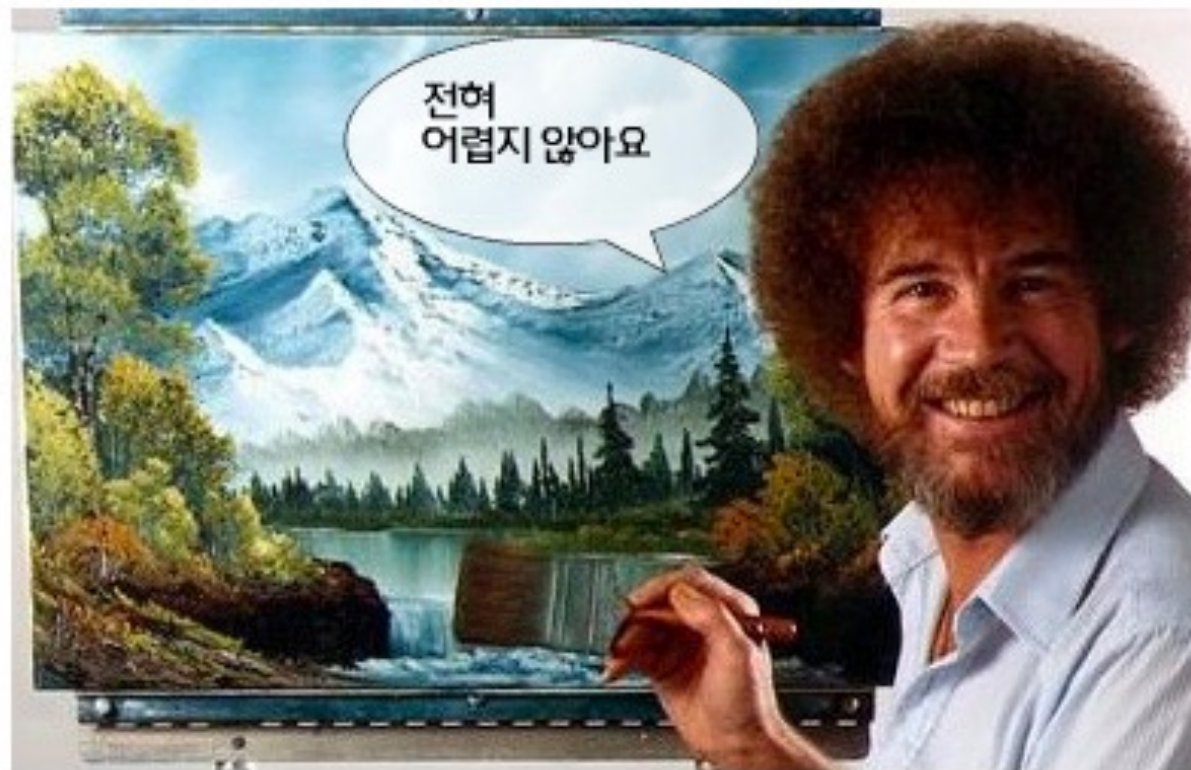
- `def doubling(x):`
- `return 2*x`
-
- `x = 5`
- `x = doubling(x+5)`
 - Evaluate argument and get a value (10)
 - Using x in the global namespace
 - Creating a namespace
 - Assign the argument value to the function parameter
 - Get the return value of the function



Memory Model for Function Call

- `def doubling(x):`
- `return 2*x`
-
- `x = 5`
- **x** = `doubling(x+5)`
 - Evaluate argument and get a value (10)
 - Using x in the global namespace
 - Creating a namespace
 - Assign the argument value to the function parameter
 - Get the return value of the function
 - Terminate the function and assign the result to x





어때요 ? 참 쉽죠 ?

Function Design

Lecture 2-3

Hyung-Sin Kim



SNU Graduate School of Data Science

To write your own functions...

Guidelines for Designing New Functions

- Recall what coding is...
 - Coding is writing an algorithm using programming language
 - Meaning that you must have an algorithm (a plan) in advance!
- Writing a good essay/paper requires planning
 - Topic, background material, outline, body
- Writing a good function does need a plan!



Guidelines for Designing New Functions

- Whenever writing a new function, you need to answer the following questions
 - **Name:** What do you name the function?
 - **Param:** What are the parameters, and what types of information do they refer to?
 - **Body:** What calculations are you doing with that information?
 - **Return:** What information does the function return?
 - **Test:** Does it work like you expect it to?

Example of Function Design

- We want to write a function for calculating difference between two days
- Step 1: Determine function name according to what it does
 - days_difference
- Step 2: Determine parameters and return value
 - Parameters: day1 (int), day2 (int)
 - Return value: difference between the two days
- Step 3: Make some test cases (your expectation)
 - days_difference(200, 224) ➡ 24
 - days_difference(27, 27) ➡ 0
 - days_difference(18, 30) ➡ 12

Example of Function Design

- Step 4: Write the function header
 - `def days_difference(day1: int, day2: int) -> int:`
- Step 5: Write a short description
 - `# Return the number of days between day1 and day2, which are both in the range 1-365 (thus indicating the day of the year)`
 - This is a very important step, both for you and your co-workers!
- Step 6: Write the function body
 - `return day2 - day1`
- Step 7: Test
 - Confirm if your function works well for the test cases you made

Summary

- Function structure (header and body)
- Namespace and local variable
- What happens when you call a function
- Guidelines for writing a new function

Thanks!