

Review

- String type
 - text values and operations
 - Print and Input
- Bool type
 - True/False and Logical operations
- Relational operators and “in” operator
 - Bool type output
- Control where to execute depending on conditions

Modules - Importing

Lecture 4-1

Hyung-Sin Kim



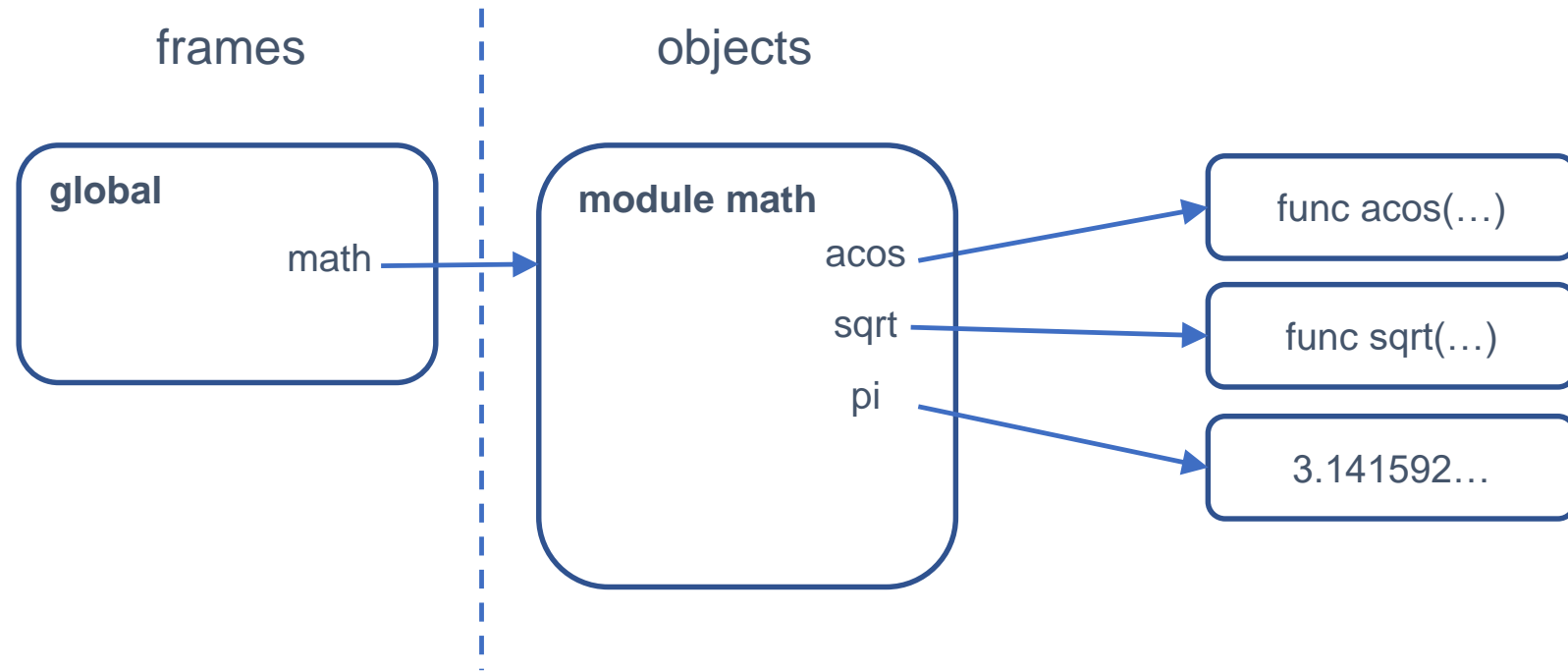
SNU Graduate School of Data Science

Module

- A package of (reusable) variables and functions, grouped together in a single file
- Once you **import** a module, you can use all the functions and variables in the module
 - `>>> import math`

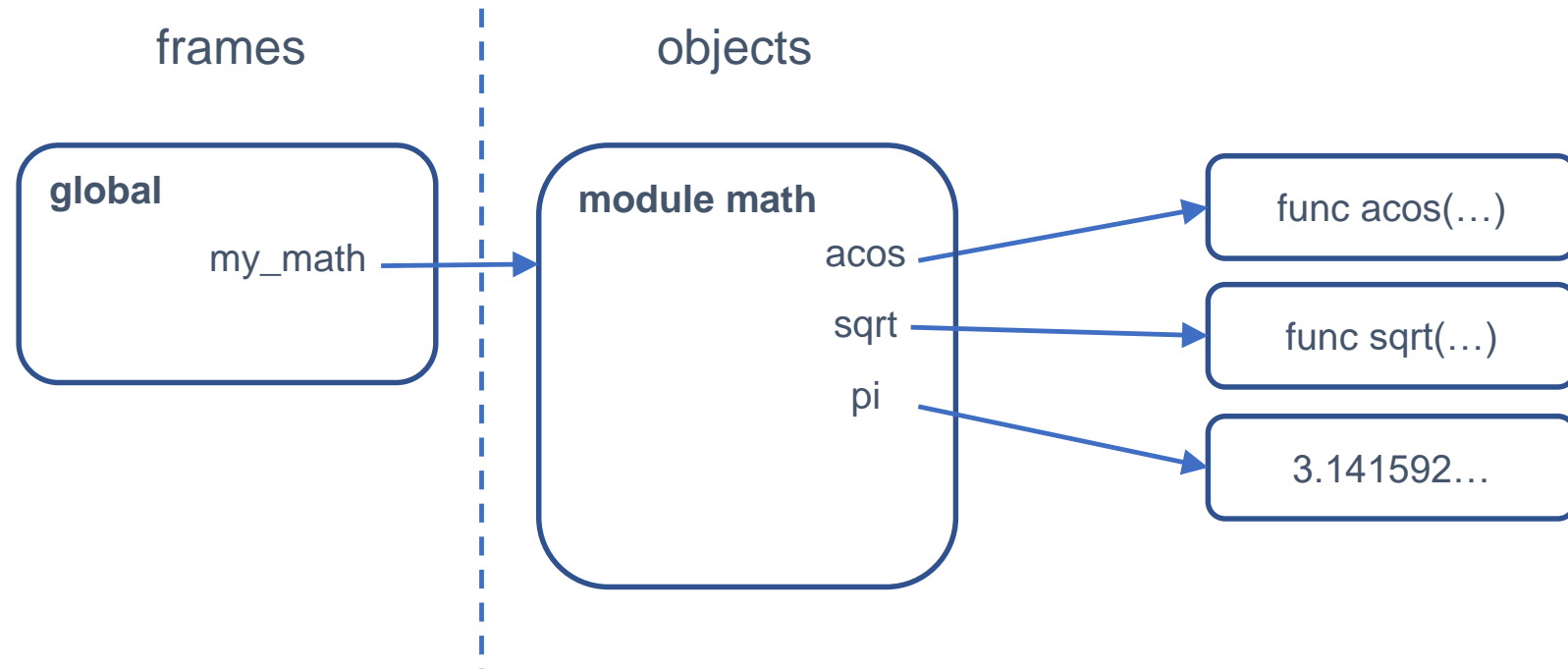
Module – What Happens when Importing?

- `>>> import math`
 - Creates a variable called **math** that refers to the math module **object**



Module – What Happens when Importing?

- `>>> import math as my_math`
 - Creates a variable called **my_math** that refers to the math module object



Module – Using Functions and Variables

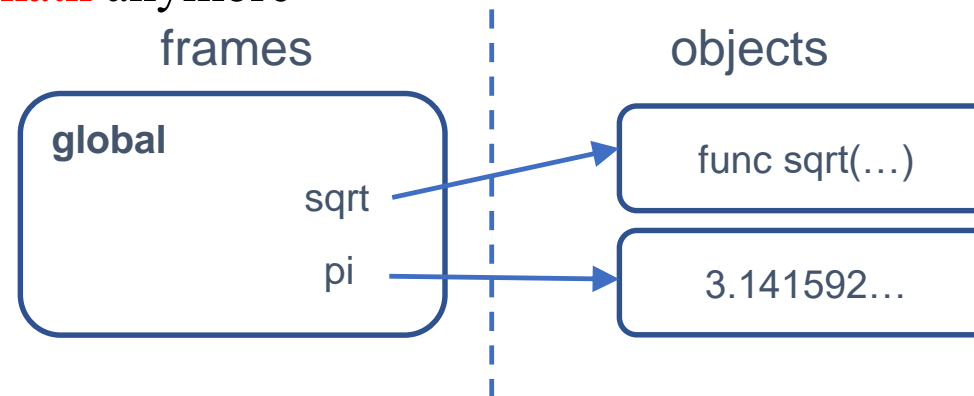
- After importing, we can use functions and variables in math module as follows:
 - `>>> math.xxxxx`
 - `>>> math.pi` → 3.141592653589793
 - `>>> math.sqrt(9)` → 3.0
- **Dot** is an operator like + and *
 - `>>> A.B`
 - Step 1: Look up the object that the variable A refers to
 - Step 2: In that object, find the name B

Module – Warning

- You can even change the value of a module's variable, but DON'T DO THIS!
 - `>>> math.pi = -17482`
 - Likewise, DON'T MANIPULATE a built-in function's name! (e.g., `max = min`)
- Python does not allow programmers to “freeze” values, which is a significant **flaw**
 - Other programming languages like C provides **constants** as well as variables and do not allow the constants to be changed

Module – Importing Specific Things

- It is **inconvenient** to always type **math** to use something in the module **math**
- We can import specific functions and variables into the current namespace
 - `>>> from math import sqrt, pi`
 - Now we don't have **math** but **sqrt** and **pi** directly in the current namespace
 - Don't have to and cannot use **math** anymore
 - `>>> sqrt(9) ➡ 3.0`



Module – Importing Specific Things

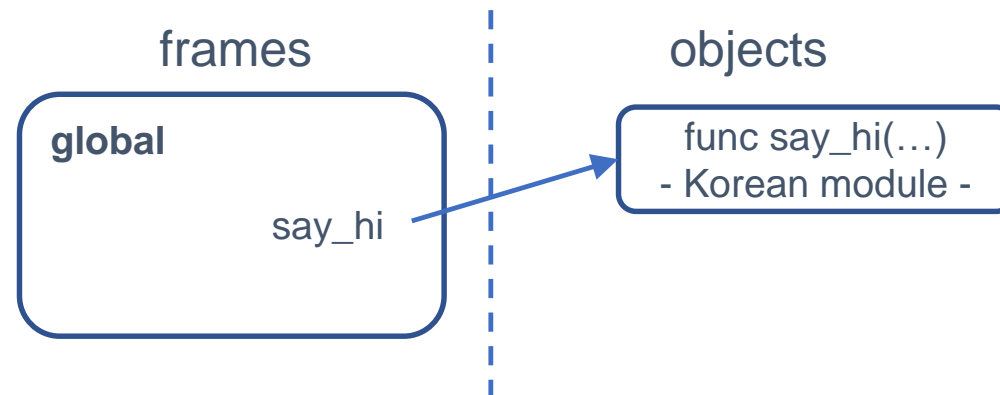
- Importing a single function **does not save** memory compared to importing an entire module
 - All things in math module are loaded on memory anyway
 - But the difference is how the module is mapped to names in the global namespace
 - <https://www.youtube.com/watch?v=vhzdwoJangI>
- Therefore, importing specific things really for the ease of typing ...

Module – Importing Specific Things

- What if you do “from math import *”?
 - You can import all the functions and variables from math module and use them directly. You can save some typing. Great!
- But this is **NOT** a good idea!
 - Python library contains **several hundreds** modules
 - Multiple modules might have functions or variables having **same name**

Module – Importing Specific Things

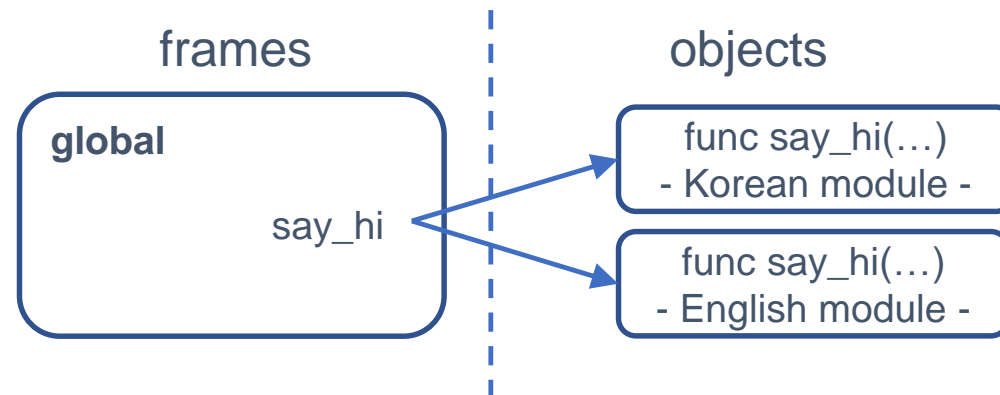
- What's wrong with that?
 - >>> from Korean import say_hi (안녕하세요)



Module – Importing Specific Things

- What's wrong with that?
 - >>> from Korean import say_hi (안녕하세요)
 - >>> from English import say_hi (Hello)
 - Say_hi function in Korean module is just gone, replaced by that in English module
 - >>> say_hi() ➡ Hello
 - **NOT** wise to take this risk by doing...
 - >>> from Korean import *
 - >>> from English import *

You have no idea what (bad things) may happen later...



Summary

- Module: One step forward to reusable programming
- A.B
- Memory and module import
 - Entire module vs. specific function/variable
- Namespace and module import
 - Entire module vs. specific function/variable

Modules – Writing

Lecture 4-2

Hyung-Sin Kim



SNU Graduate School of Data Science

Module – Make Your Own Module

- Open a new text file and make the file name “temperature.py”
 - Now Jupyter knows that it is Python file

- Write some variables and functions in the file

```
1 a = 5.0
2 b = 9.0
3 c = 32.0
4
5 def convert_to_celsius(fahrenheit: float) -> float:
6     return (fahrenheit - c) * a / b
7
8 def convert_to_fahrenheit(celsius: float) -> float:
9     return celsius * b / a + c
```

- Save the file

Module – Make Your Own Module

- Now you have a temperature module
 - `import temperature`
 - Play with variables and functions in your own module!
- Good and convenient to group related variables and functions in a module!

Classes

Lecture 4-3

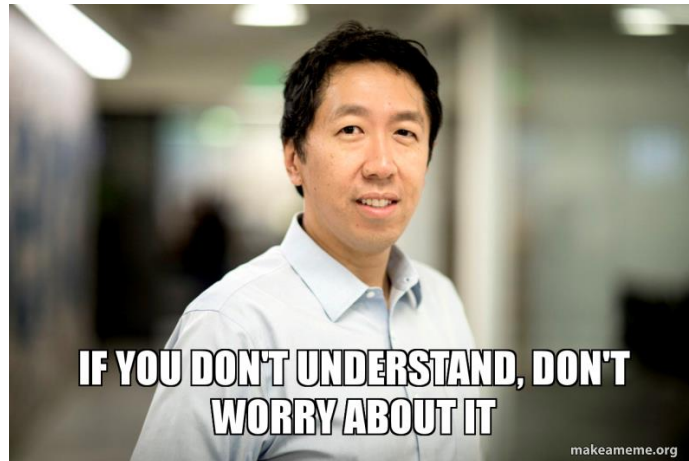
Hyung-Sin Kim



SNU Graduate School of Data Science

Class

- Class is another type of **object** that has variables and functions similar to module
 - Functions in a class are called **methods**
 - A method takes its class object as the first and default argument
 - You can call a class's methods like calling a module's functions
 - `<<class object>>.<<method>>`
- There are more differences between class and module but...



Class

- All the types we've seen so far are actually classes
 - int class, float class, str class, bool class
 - You can see their definitions by typing, for example **help(int)**
- Class vs. Class object
 - A class is a **blueprint**, definition of its attributes and methods (but not real yet)
 - A class object is a class' **instance** (realization of a blueprint)
 - Example: TESLA model Y
 - TESLA designed **model Y's blueprint** and produces numerous **model Y objects** according to the blueprint.
 - I can buy a **model Y object** but not **the model Y blueprint**
 - **The model Y blueprint** has an attribute 'color.'
 - My **model Y object's** color attribute is blue
 - Your **model Y object's** color attribute is red
 - I can drive a **model Y object** without knowing its **blueprint**



Class

- Python has built-in classes, such as int, float, str, and bool
- We can create objects of these classes and easily use their methods and attributes, without worrying about how they are implemented
 - Yes, this is an example of **abstraction**!
- Example: **str** class
 - `>>> name = "Hyung-Sin Kim"` (name points at an **object** of str class, "Hyung-Sin Kim")
 - When calling str methods, you don't input name as the argument!
 - `>>> name.lower()` ➡ "hyung-sin kim"
 - `>>> name.upper()` ➡ "HYUNG-SIN KIM"
 - `>>> name.find("S")` ➡ 6 (location where "S" appears first in "Hyung-Sin Kim")
 - `>>> name.count("n")` ➡ 2 (# of "n" in "Hyung-Sin Kim")

Class

- You will write your own classes (**blueprint**) later
 - **Object**-oriented programming
- But now please be familiar with methods in **str class**
 - Manipulating strings fluently is super important!
 - https://www.w3schools.com/python/python_ref_string.asp
 - Count, find, lower, upper, lstrip, rstrip ...
- You can understand how str methods work except split(), which will be described later (output type is List)

It is OK to not understand 100% for now.

*You will be more familiar with this concept
as you use various classes and
learn object-oriented programming*

Summary

- Class vs. Class object
- How to use methods in a class

Thanks!