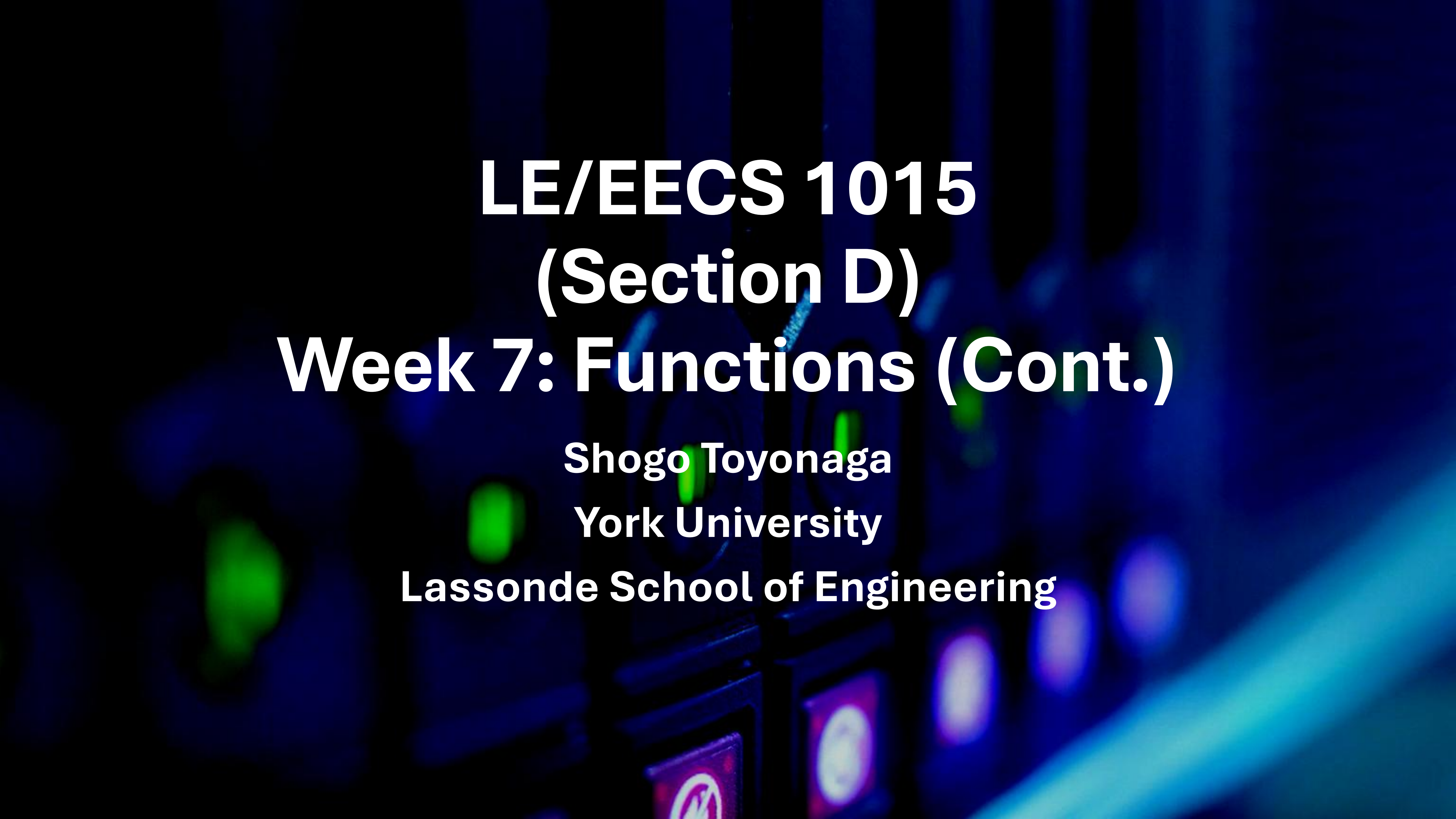


LE/EECS 1015 (Section D) Week 7: Functions (Cont.)

Shogo Toyonaga

York University

Lassonde School of Engineering

This Week…

1. Modules
    - How can we use them?
    - How can we create our own?
    - Why should we use them?
2. Function Calls
    - Positional Arguments
    - Keyword Arguments
    - Default Arguments

Midterm Information

- Midterms will take place in your assigned lab section after reading week.
- 90 Minutes covering material from Week 1 – Week 5 (Inclusive)
- Format:
    1. 10 Multiple Choice
    2. Coding Question
    3. Debugging Question
    4. Coding Question
- **You must use the lab desktops; PLEASE ensure that your EECS account works properly.**
- You will only have access to the following IDEs: PyCharm, Wing Personal, Juypter Notebook
- Closed book; no access to email, printing, external web resources, etc.,

Midterm Information

- You must submit your answers BEFORE the end of the 90 minutes. Unsubmitted files cannot be recovered.

- Your submission will be auto-graded but you will not be able to see your score during the lab test.

- If your code has syntax errors, the question will receive 0 marks. No re-grading requests will be accepted for syntax or formatting errors.

Some Advice from a Past EECS Student

1. Practice, Practice, Practice
    - Review each of the Weekly Quizzes
    - Do the optional weekly activities
    - Redo the labs (seriously)
2. Active Recall
    - https://codingbat.com/python
        - Warmup-1
        - String-1
        - Logic-1
        - Logic-2
3. Tutoring
    - Feynman Technique: Teach the concepts to a beginner. If you can't explain a concept simply enough, it means that you have not mastered the material yet.
4. Office Hours
    - We will continue to hold daily sessions during the reading week. Please take advantage of the opportunity if you need help.

Some Advice from a Past EECS Student

"Nothing is permanent in this wicked world - not even our troubles"

- *Charlie Chaplin*

You will be okay; good luck!!!! 😊

Modules (Introduction)

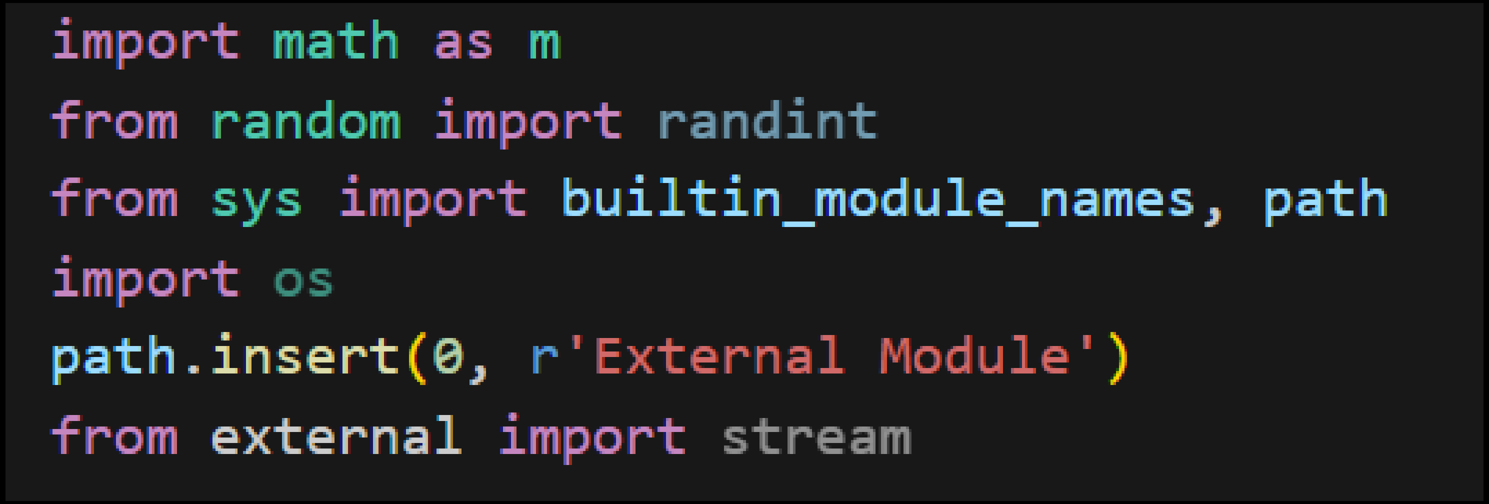
- A module is a file which contains reusable Python definitions such as functions, variables (constants), and statements.

- Python has built-in modules (docs.python.org/3/library/) and third-party modules (pypi.org/) which provide a wealth of capabilities.

- Prior to using a module, it must be imported into your code

- You can create your own modules by creating a Python File in the same directory/folder as your script. You can also manually add a module in a different directory to sys.path.

Modules



```python
import math as m
from random import randint
from sys import builtin_module_names, path
import os
path.insert(0, r'External Module')
from external import stream
```

Modules

- You can view the names that a module defines by using the built-in $dir(\ldots)$ function.

- This can be chained with $help(\ldots)$ to view the associated documentation for a provided name.

Modules: Challenge

1. Look at the Lecture 6 Recitation!

2. Run each of the cells in the Jupyter notebook. If you don’t understand what a method does, refer to the documentation!

3. Add your own methods; have fun!

4. Homework: Install a third-party module and add some demos to the notebook! Be creative :D

Modules (math) – Some Docs….

Name	Meaning
$math.ceil(x,/) \rightarrow int$	Returns the ceiling of x as an integral. In short, it rounds the number up to the nearest integer.
$math.floor(x,/) \rightarrow int$	Returns the floor of x as an integral. In short, it rounds the number down to the nearest integer.
$math.\sin(x,/) \rightarrow float$	Returns the sine of x (Measured in Radians)
$math.\cos(x,/) \rightarrow float$	Returns the cosine of x (Measured in Radians)
$math.\tan(x,/) \rightarrow float$	Returns the tangent of x (Measured in Radians)
$math.degrees(x,/) \rightarrow float$	Convert angle x from radians to degrees
$math.radians(x,/) \rightarrow float$	Converts angle x from degrees to radians
$math.e$	Returns Eulers constant ($e \approx 2.718281 \ldots$)
$math.pi$	Returns $\pi \approx 3.141592 \ldots$
$math.inf$	Returns a floating-point positive infinity value.

Modules (random) – Some Docs….

Name	Meaning
$random.normalvariate(\mu = 0.0, \sigma = 1.0) \rightarrow float$	Returns a random float which is sampled from the Normal Distribution.
$random.randint(a, b) \rightarrow int$	Returns a random integer $r \in [a, b]$.
$random.randrange(start, stop = None, step = 1) \rightarrow int$	Returns a random item from $[start, stop)$, given a step value.
$random.random() \rightarrow float$	Returns the next random floating-point $\in [0.0, 1.0)$.
$random.uniform(a, b) \rightarrow float$	Returns a random float in the range $[a, b)$ or $[a, b]$ depending on the rounding.

Function Calls

- Pythons Positional arguments are passed to the function in the order that they are defined.

- You can force people to use positional arguments by including / as the last parameter in your function signature. For example, running $abs(x = 5)$ will throw an error whereas $abs(5)$ will not.

- Conversely, Keyword arguments are passed to the function using the names of parameters. As such, the order can be arbitrary and does not need to strictly follow the function signature.

- Lastly, default arguments can be included in the function signature for parameters. If a user forgets to provide the parameter value, you are safe!

Thank You!

Shogo Toyonaga
Lassonde School of Engineering