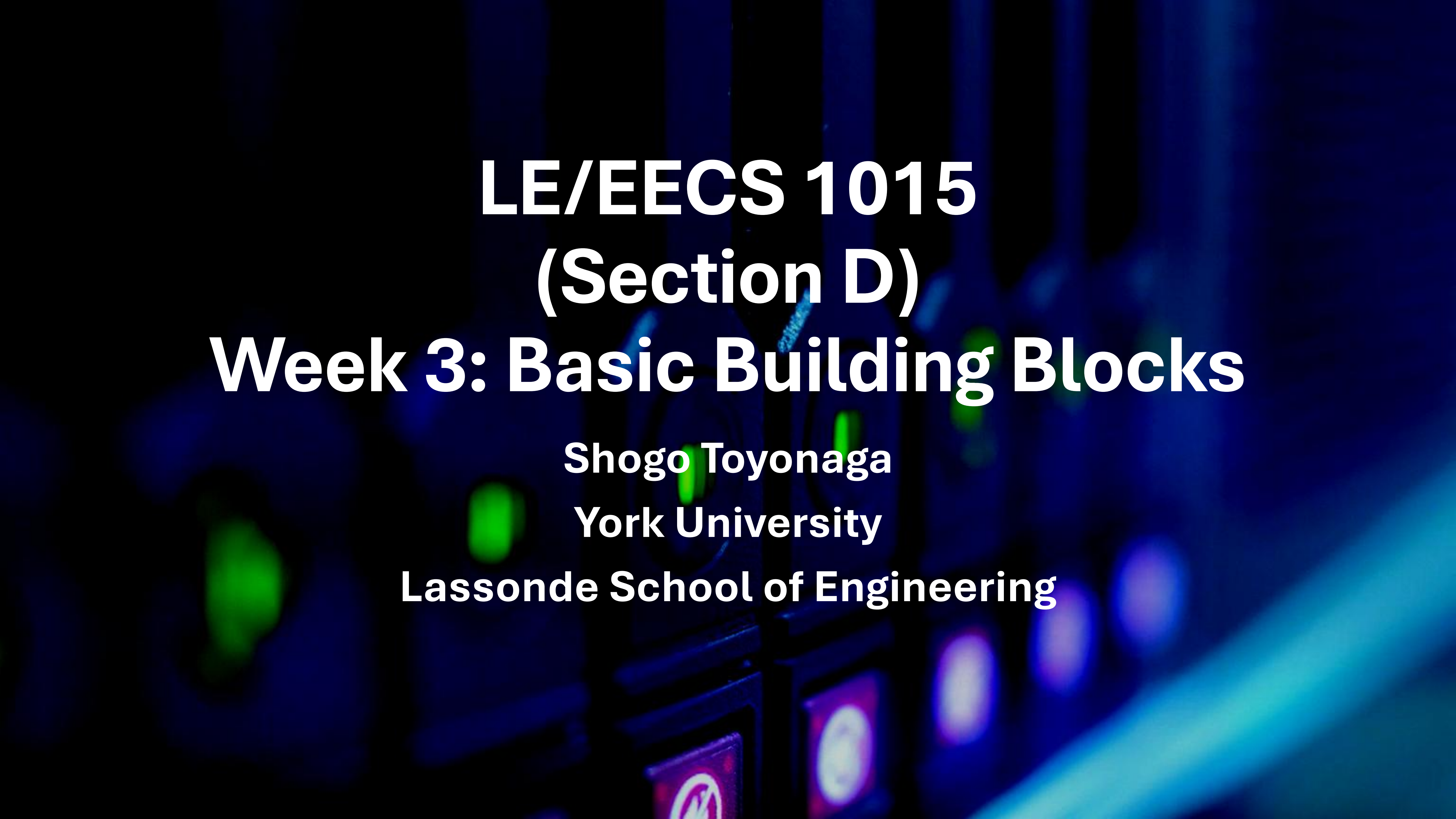


LE/EECS 1015 (Section D) Week 3: Basic Building Blocks

Shogo Toyonaga

York University

Lassonde School of Engineering

This Week…

1. Booleans
    - True
    - False
    - Order of Operations (¬, ∧, ∨)
    - Logical Equivalences (LE/EECS 1019 🥹)
2. Strings
    - Representation
    - Operations
    - Indexing
    - Slicing
    - String Methods
    - Formatting

Goals of Lab 3

1. Strengthen your computational thinking skills

2. Write simple scripts with String operations

3. Extract string sequences with slicing

4. Practice debugging with breakpoints

Lab 3 – What You Do….

Task	Points
Follow the Steps ($x \# y \equiv x^2 - y^2$)	30
Debugging ($x \oplus y$)	30
Implementation (Child Tickets)	10
Implementation (Phone Number)	10
Implementation ($FL \rightarrow L, F$)	4
Implementation ($L, F \rightarrow L$)	8
Implementation ($L, F \rightarrow F$)	8

Lab 3 – Useful Resources

- String methods in Python are easy 〰️ (Bro Code)
- Python string slicing ✂️
- Python 3 String Methods (Official Documentation)
- ALL 47 STRING METHODS IN PYTHON EXPLAINED (Indently)

Booleans

- Are a type of data type which can be: $\{True, False\}$
- Typically associated with expressions that use comparison or logical operators
- If you are casting to a Boolean, any datatype that is (1) not None, or (2) greater than 0 will evaluate to True.

Logical Operators (Order of Operations)

English Interpretation	Operator	Explanation
- ***Less Than*** - ***Less Than or Equal To*** - ***Greater Than*** - ***Greater Than or Equal To*** - ***Equal To*** - ***Not Equal To***	$<, \leq, >, \geq, ==, !=$	Comparison Operators
- ***not p***	$\neg p$	Logical Bitwise Negation
- ***p and q***	$(p \land q)$	Logical Boolean And
- ***p or q***	$(p \lor q)$	Logical Boolean Or
- ***If p then q***	$p \rightarrow q$	Conditional Statement
- ***p if and only if (iff) q***	$p \leftrightarrow q$	Biconditional Statement

Logical Operators: Negation

p	$\neg p$
T	F
F	T

Logical Operators: AND

p	q	$(p \land q)$
T	T	T
T	F	F
F	T	F
F	F	F

Logical Operators: OR

$\boldsymbol{p}$	$\boldsymbol{q}$	$(\boldsymbol{p} \vee \boldsymbol{q})$
T	T	T
T	F	T
F	T	T
F	F	F

Logical Equivalences

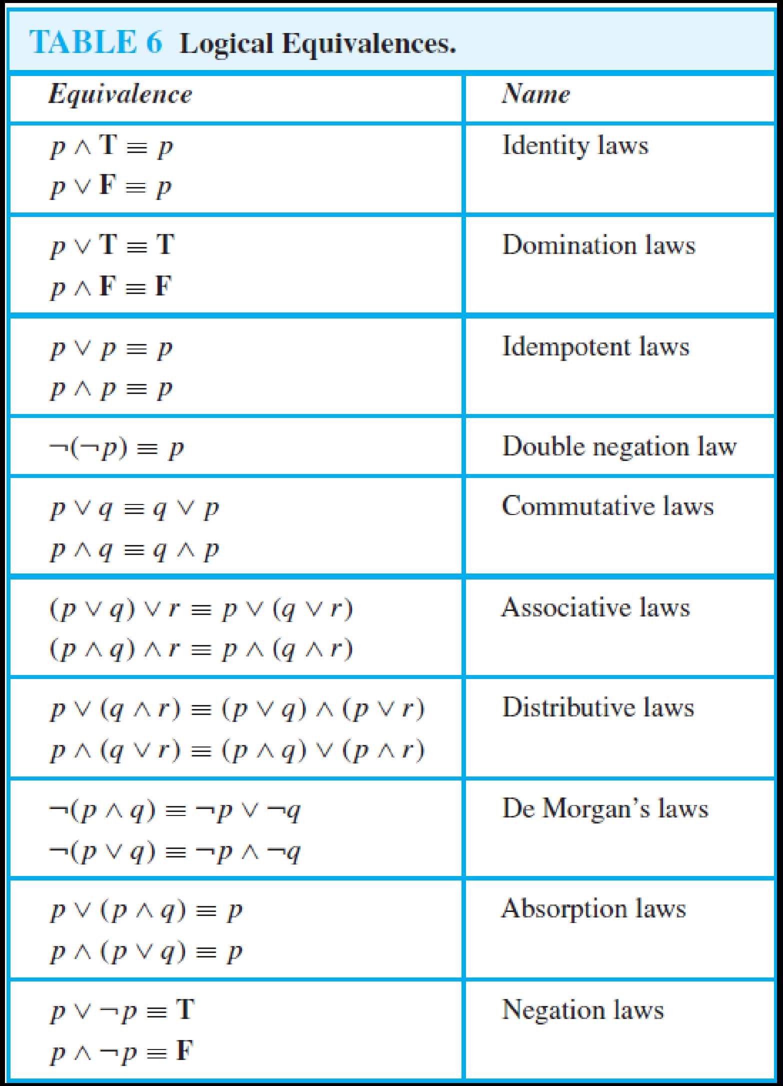


TABLE 6 **Logical Equivalences.**

Equivalence	*Name*
$p \wedge \mathbf{T} \equiv p$ $p \vee \mathbf{F} \equiv p$	Identity laws
$p \vee \mathbf{T} \equiv \mathbf{T}$ $p \wedge \mathbf{F} \equiv \mathbf{F}$	Domination laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent laws
$\neg(\neg p) \equiv p$	Double negation law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative laws
$(p \vee q) \vee r \equiv p \vee (q \vee r)$ $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	Associative laws
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	Distributive laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgan's laws
$p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$	Absorption laws
$p \vee \neg p \equiv \mathbf{T}$ $p \wedge \neg p \equiv \mathbf{F}$	Negation laws

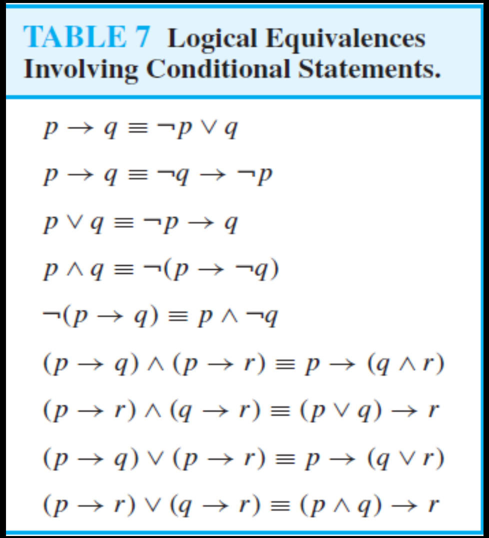


TABLE 7 **Logical Equivalences Involving Conditional Statements.**

$p \rightarrow q \equiv \neg p \vee q$
$p \rightarrow q \equiv \neg q \rightarrow \neg p$
$p \vee q \equiv \neg p \rightarrow q$
$p \wedge q \equiv \neg(p \rightarrow \neg q)$
$\neg(p \rightarrow q) \equiv p \wedge \neg q$
$(p \rightarrow q) \wedge (p \rightarrow r) \equiv p \rightarrow (q \wedge r)$
$(p \rightarrow r) \wedge (q \rightarrow r) \equiv (p \vee q) \rightarrow r$
$(p \rightarrow q) \vee (p \rightarrow r) \equiv p \rightarrow (q \vee r)$
$(p \rightarrow r) \vee (q \rightarrow r) \equiv (p \wedge q) \rightarrow r$

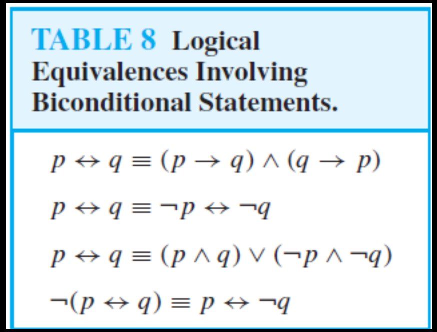


TABLE 8 **Logical Equivalences Involving Biconditional Statements.**

$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$
$p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$
$\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$

Evaluation with Logical Operation

- Lazy Computation / Lazy Evaluation is used to evaluate logical expressions
  - For example, if the LHS of the AND operator is False, we return False. You do not need to evaluate the RHS in this case.
  - For example, if the LHS of the OR operator is True, we return True. You do not need to evaluate the RHS in this case.

- Domination Law is a quick way to build intuition on Lazy Computation with Logical Operations.

Strings

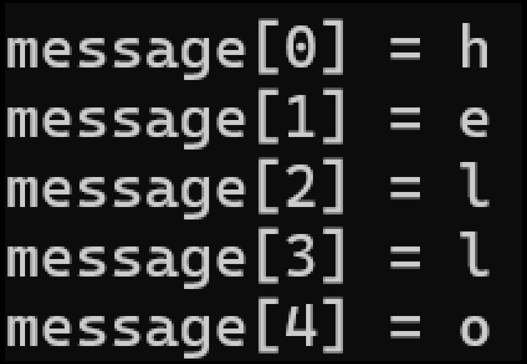
- A string is an immutable sequence of characters. A character in this context can be a letter, a digit, a punctuation mark, or whitespace.
- We can represent strings using:
  - Single Quote Notation
  - Double Quote Notation
  - Multi-Line Triple Double Quote Notation
  - R-Strings
  - F-Strings (Curly Braces for Substitution!)
- Do note that certain characters in certain scenarios will need to be represented by using escape characters.

String Operations

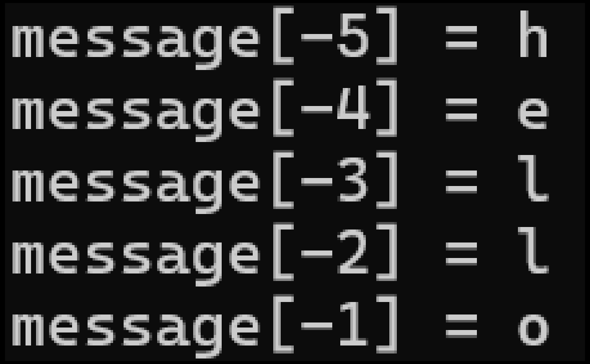
- Concatenation of two string data types uses ‘+’ as the operator
- Repetition of a string n times uses ‘$*$’ as the operator
- Comparisons of strings are based on lexicographic ordering.
- String membership can be assessed by using the, ‘in’ operator
- String length can be assessed dynamically by using the, ‘len’ function.

String Indexing

- You can select a character form a string by using the bracket operator. That is, we provide the index
- Indexing is sometimes thought of providing an, “offset” from the first element
- Python supports positive and/or negative indexing. Negative indices start from the last element (e.g., -1)



```
message[0] = h
message[1] = e
message[2] = l
message[3] = l
message[4] = o
```



```
message[-5] = h
message[-4] = e
message[-3] = l
message[-2] = l
message[-1] = o
```

String Slicing

- Slicing represents retrieving the segment (or subset) of a string.
  - $message[n: m]$ returns the subset of the string from index range $[n, m)$. The default step-size is set to increment by 1
  - $message[n: m: q]$ returns the subset of the string from index range $[n, m)$ using a step-size of q
- Negative Indexing can be used here to define the start or end of the string slice!

String Methods

- Strings implement the common sequence operations along with additional methods that have been documented online: https://docs.python.org/3/library/stdtypes.html#string-methods
- Example Built-In String Methods (There are many, many more!):
  - $str.count()$
  - $str.endswith()$
  - $str.find(), str.index()$
  - $str.format()$
  - $str.isdigit()$
  - $str.partition()$
  - $str.replace()$

Thank You!

Shogo Toyonaga
Lassonde School of
Engineering