

# LE/EECS 1015 (Section A: LAB 04) Week 2: Lab #2

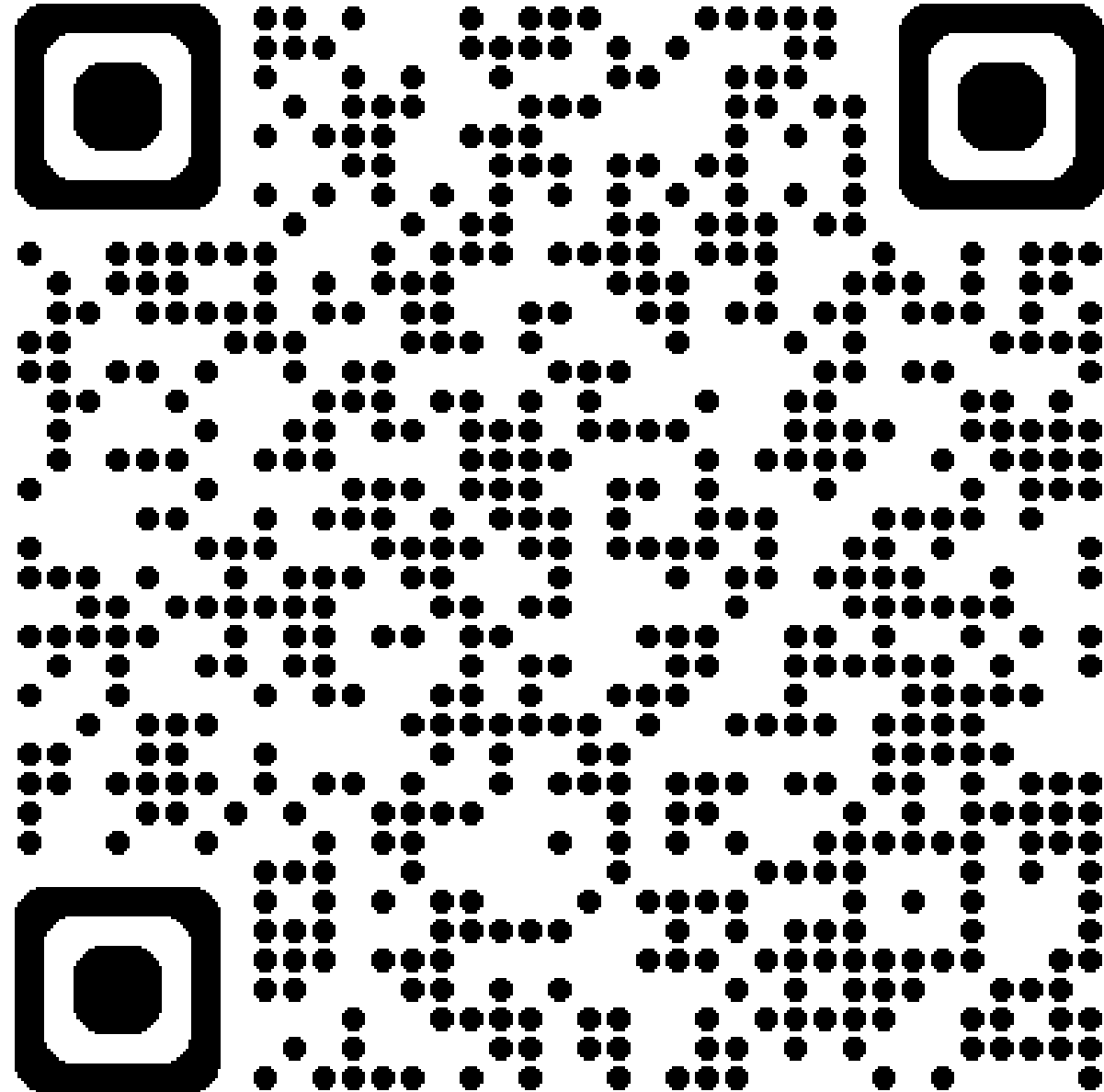
Shogo Toyonaga  
York University  
Lassonde School of Engineering



# Logistics

All of my slides & tutorial notebooks will be available through the following GitHub repository:

[https://github.com/Shogz-Labs/EECS1015\\_F24\\_Assets](https://github.com/Shogz-Labs/EECS1015_F24_Assets)



# Goals of Lab 2

1. Using what you have learnt from *Basic building blocks (I-II)*, write a simple script with primary components
2. Learning to **debug** with Wing IDE (Recommended) or PyCharm

# Debugging

“Debugging is the process of finding and fixing errors or bugs in the source code of any software”

Wing IDE & PyCharm have tools to help you find **bugs** and squash them.

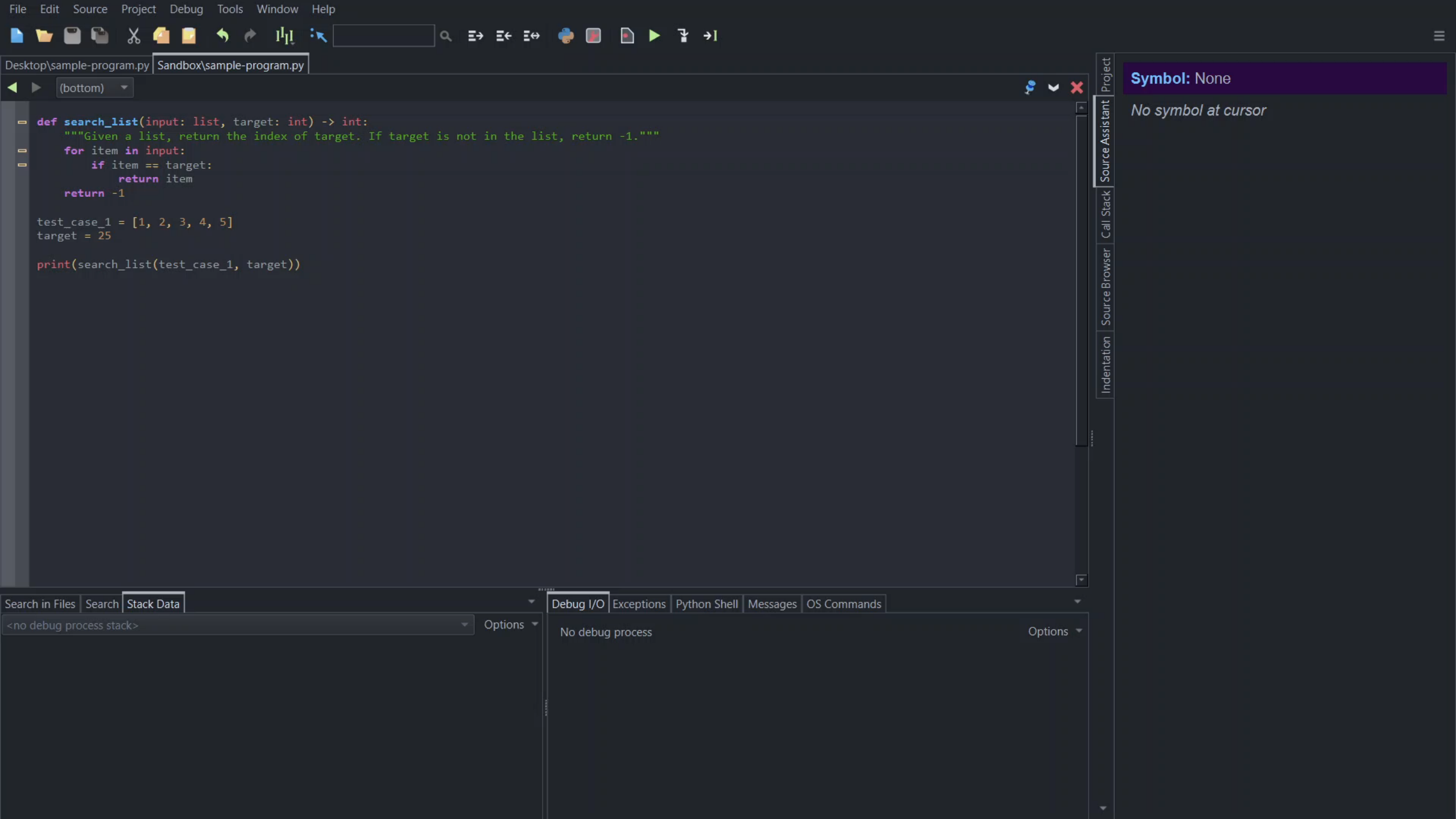
- **Breakpoints** allow you to interrupt the execution of your code at a certain place to view data values and assignments.
- The **stack data** allows us to see variable values and assignments at runtime. This can help us see line-by-line where things go wrong.

# Debugging (Terminology & Operations)

Step into current execution point,  
or start debugging at first line (F7)

Step over current statement (F6)

Step out of current function or  
method (F8)



```
def search_list(input: list, target: int) -> int:
    """Given a list, return the index of target. If target is not in the list, return -1."""
    for item in input:
        if item == target:
            return item
    return -1

test_case_1 = [1, 2, 3, 4, 5]
target = 25

print(search_list(test_case_1, target))
```

Symbol: None

No symbol at cursor

# Concept Review

## 1. Fundamentals of High-Level Programming

- Literals, Operators, Expressions
  - BEDMAS!
- Variables, Assignment, Memory Model
- Statements, Functions
- Application Programming Interface (APIs)
- Booleans & Logical Operations (LE/EECS 1019)
- Strings (Representation, Escape Characters, Operations, Methods, & Formatting)

## 2. Fundamental Principles

- Documentation (Readability, Simplicity, and Comments)
  - Especially important because of dynamic typing!
- Design Patterns
  - Precondition, Loop Invariant, Postcondition

# Concept Review (String I)

- String indexing always starts from **0**! In general, we always index the first item in an array, list, etc., starting from  $[0..n)$  for  $n$  items.
  - Python supports **negative indexing**.
- You can obtain a substring of a string by using the following syntax: *string\_variable[start:end:step]* where  $[start, end)$  and *step* is 1 by default
  - The start, end, or step can be **omitted** to indicate the **default** value.



# Concept Review (String II)

- Strings are **immutable** (cannot be changed after being defined)
- Strings are **objects**
- The String **class** supplies multiple supported methods:
  - e.g., `str_variable_name.capitalize()`

# Concept Review (String Methods)

1. `str.capitalize(self, /)` – Make the first character have upper case and the rest lower case.
  2. `str.casefold(self, /)` – Return a version of the string suitable for caseless comparisons.
  3. `str.lower(self, /)` – Return a copy of the string converted to lowercase.
  4. `str.swapcase(self, /)` – Convert uppercase characters to lowercase and lowercase characters to uppercase.
  5. `str.title(self, /)` – Return a version of the string where each word is title-cased.
  6. `str.upper(self, /)` – Return a copy of the string converted to uppercase.
  7. `str.isalnum(self, /)` – Return True if the string is an alpha-numeric string, False otherwise.
  8. `str.isnumeric(self, /)` – Return True if the string is a numeric string, False otherwise.
  9. `str.isalpha(self, /)` – Return True if the string is an alphabetic string. False otherwise.
  10. `str.islower(self, /)` – Return True if the string is a lowercase string, False otherwise.
  11. `str.isupper(self, /)` – Return True if the string is an uppercase string, False otherwise.
- 
1. `str.count(substring)` – Return the number of non-overlapping occurrences of the substring in str.
  2. `str.find(string)` – Return the lowest index in str where string is found. Return -1 on failure.
  3. `str.endswith(string)` – Returns true if str ends with the specified suffix, False otherwise.
  4. `str.index(string)` – Return the lowest index in str where substring string is found. Raises ValueError when string is not found in str.
  5. `str.startswith(string)` – Return True if str starts with string. False otherwise.
  6. `str.replace(self, old, new, count=-1, /)` – Return a copy with all occurrences of substring old replaced by new.
  7. `str.strip(self, chars=None, /)` – Return a copy of the string with leading and trailing whitespaces removed. If chars is given and not None, remove characters in chars instead.

# What You Will Need





## Lab 2 – Objectives

1. Follow the Steps (/30)
2. Debugging (/30)
3. Implementation (Donuts) (/10)
4. Implementation (Grade Calculator) (/10)
5. Implementation (BMI Calculator) (/10)
6. Implementation (MinMax Average) (/10)

# Thank You!

Shogo Toyonaga

Lassonde School of Engineering

