# SDEV 1001

Programming Fundamentals

Debugging with Breakpoints - 1

## Expectations - What I expect from you

- No Late Assignments

- No Cheating

- Be a good classmate

- Don’t waste your time

- Show up to class

## Agenda

On the right is what we will cover today.

Introduction to Debugging in Python
What is a Breakpoint?
Example: Debugging a Simple Calculator
Common Debugger Commands
Inspecting Variables
Stepping Through Code
Example: Debugging a Loop
Tips for Effective Debugging

## Introduction to Debugging in Python

- Debugging is the process of finding and fixing errors in your code.
- All programmers encounter bugs—debugging is an essential skill!
- Python provides built-in tools to help you debug your programs.

## What is a Breakpoint?

- A breakpoint is a marker in your code where execution will pause.
- This allows you to inspect variables, step through code, and understand what your program is doing at that moment.
- In Python, you can set a breakpoint by adding `breakpoint()` in your code (Python 3.7+).

# Example: Debugging a Simple Calculator

```python
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

x = 10
y = 5
breakpoint()
result = add(x, y)
print("Result:", result)
```

- When you run this code, execution will pause at `breakpoint()`.
- You can now inspect the values of `x`, `y`, and step through the code.

# Common Debugger Commands

| Command | Action |
|---|---|
| n or next | Run the next line of code |
| s or step | Step into a function call |
| c or continue | Continue running until the next breakpoint or end |
| l or list | List the surrounding code lines |
| p or print | Print the value of a variable |
| q or quit | Exit the debugger |

## Inspecting Variables

- While paused at a breakpoint, you can check the value of variables:

```
(Pdb) x
10
(Pdb) y
5
(Pdb) result
*** NameError: name 'result' is not defined
```

- You can also change variable values to test different scenarios:

```
(Pdb) x = 20
(Pdb) n
```

## Stepping Through Code

- Use `n` (next) to execute the next line.
- Use `s` (step) to step into a function call.
- Use `c` (continue) to run until the next breakpoint or end of the program.

## Example: Debugging a Loop

```python
numbers = [1, 2, 3, 4, 5]
total = 0
for n in numbers:
    breakpoint()
    total += n
print("Total:", total)
```

- You can inspect `n` and `total` at each iteration.

## Tips for Effective Debugging

- Add breakpoints where you suspect issues.
- Check the values of variables and program flow.
- Use print statements for quick checks, but prefer the debugger for complex issues.
- Remove or comment out breakpoints when done.

## Summary

- Debugging helps you understand and fix your code.
- Use `breakpoint()` and the Python debugger to pause and inspect your program.
- Practice stepping through code and inspecting variables to become a better programmer!

**IMPORTANT NOTE:** This is one of the important skills and tools you'll learn in this course.

# Example

Let’s go run a few examples together