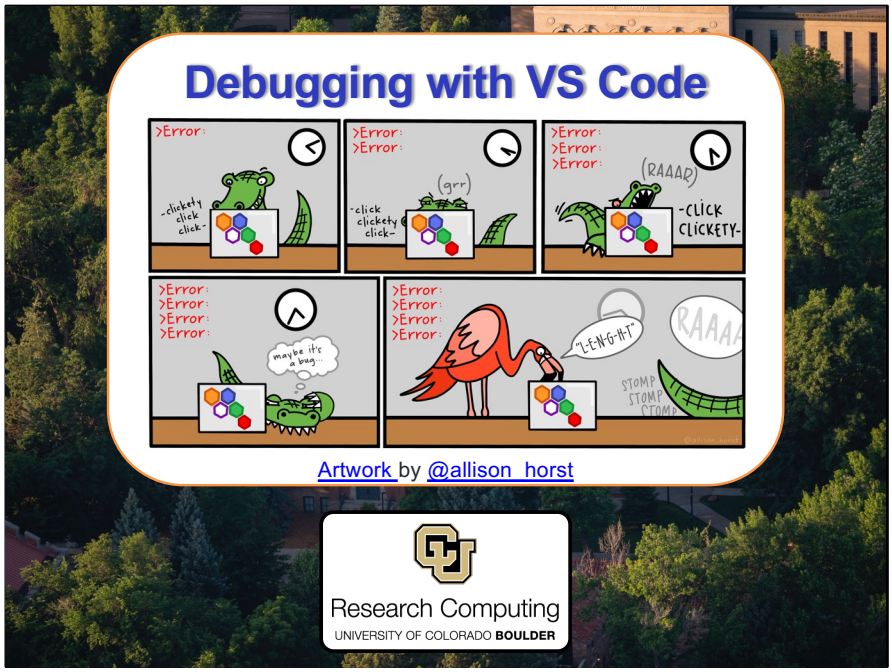


Artwork Source: https://allisonhorst.com/everything-else

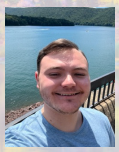
Meet the User Support Team

Layla Freeborn

Brandon Reyes

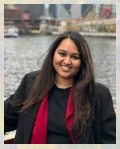
Andy Monaghan

Michael Schneider



John Reiland

Dylan Gottlieb



Mohal Khandelwal

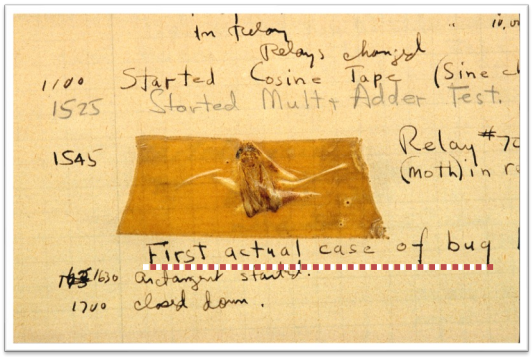
Ragan Lee

QR Code and URL to Slide Deck

Slides

https://github.com/ResearchComputing/rmacc_2025/tree/main/debugging_with_vscode

The First Programming "Bug"



Research Computing
UNIVERSITY OF COLORADO BOULDER

4

- Bugs == Errors
- The phrases (bug and debugging) are often attributed to Grace Hopper and her team – who found a literal bug (a moth) stuck in their computational machine (Mark II) – but the term is actuall an age-old engineering term dating back over century.
- In this workshop, we will discuss the process of debugging and the different techniques that you can use to better find and fix the bugs.

Description (Smithsonian):
'American engineers have been calling small flaws in machines "bugs" for over a century. Thomas Edison talked about bugs in electrical circuits in the 1870s. When the first computers were built during the early 1940s, people working on them found bugs in both the hardware of the machines and in the programs that ran them.

In 1947, engineers working on the Mark II computer at Harvard University found a moth stuck in one of the components. They taped the insect in their logbook and labeled it "first actual case of bug being found." The words "bug" and "debug" soon became a standard part of the language of computer programmers.

Among those working on the Mark II in 1947 was mathematician and computer programmer Grace Hopper, who later became a Navy rear admiral. This log book was probably not Hopper's, but she and the rest of the Mark II team helped popularize the use of the term computer bug and the related phrase "debug."'

Image & Description Source:
https://americanhistory.si.edu/collections/nmah_334663

Debugging Process

Observe a Bug

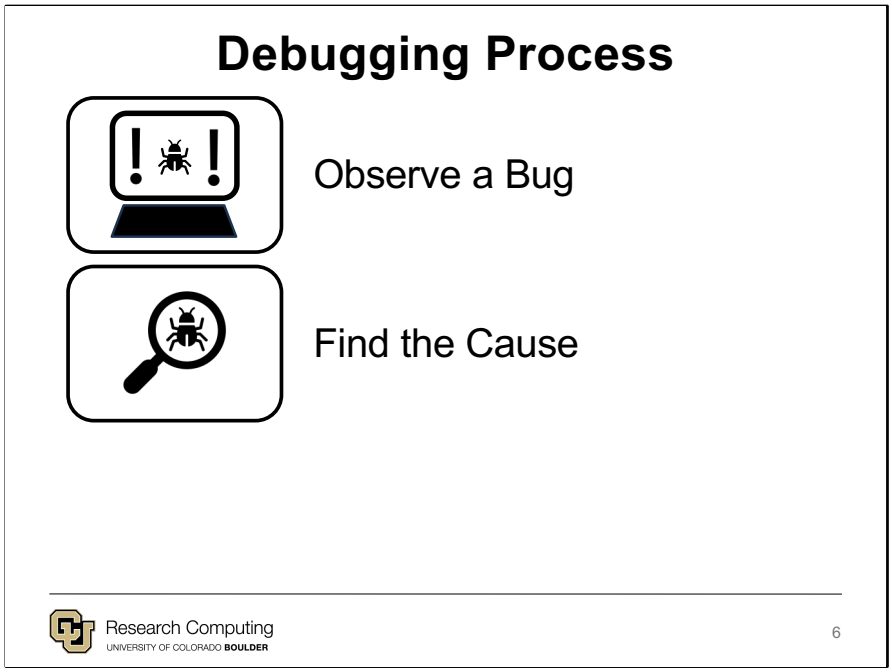
Research Computing
UNIVERSITY OF COLORADO BOULDER

5

Before we can start debugging, we must first recognize the need for debugging – i.e. observe a bug.

In the context of software programming, a bug refers to a moment when a system behaves in an unexpected or unintended manner.

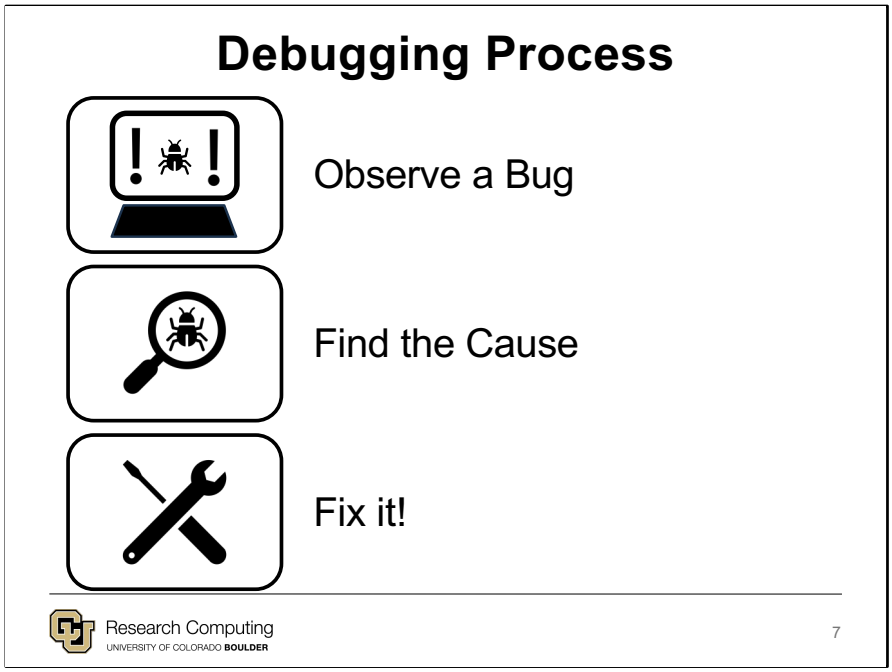
This “unexpected behavior” can run the gambit of incorrect output to the full-blown system crashes!



After observing a bug, we must start searching for its underlying cause – which could be any one of a variety of potential errors.

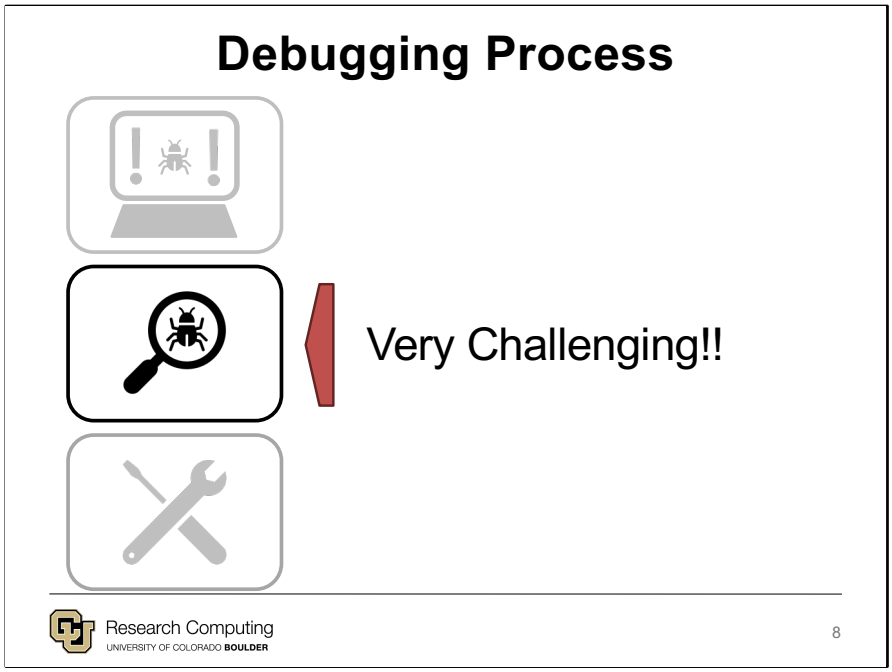
It is important to note that the error, or errors, can exist across three spaces – the project's code, the system running the code, and then engineer's mind.

Today we'll be focusing on the first space, code, but it is important to always remember to check for issues in the system's hardware and your own understanding of the system and the code.

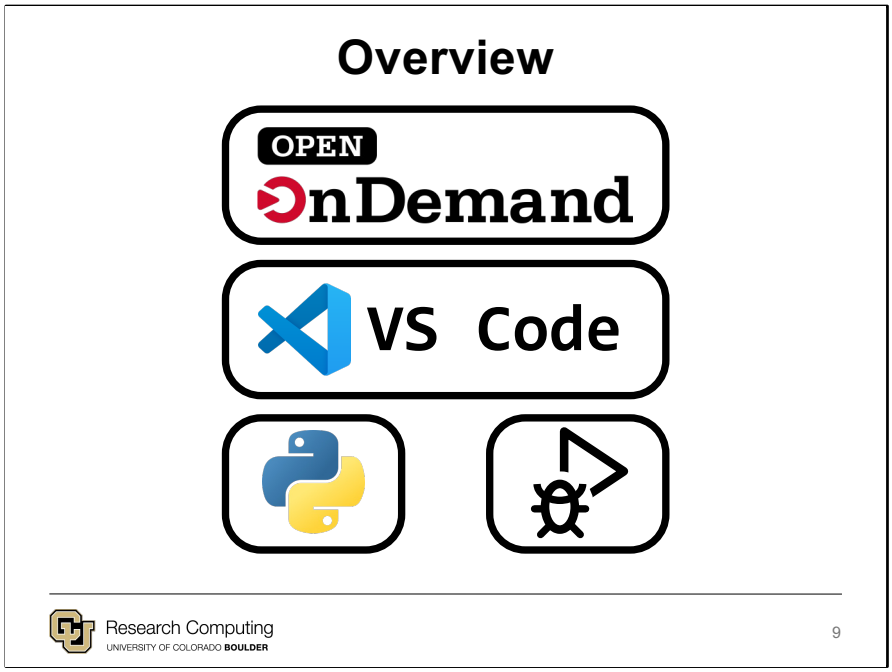


Once the error, or errors, has been identified it must be fixed.

This is the final, and often easiest, step of the debugging process.



The most challenging part of debugging is this second step – finding the error.

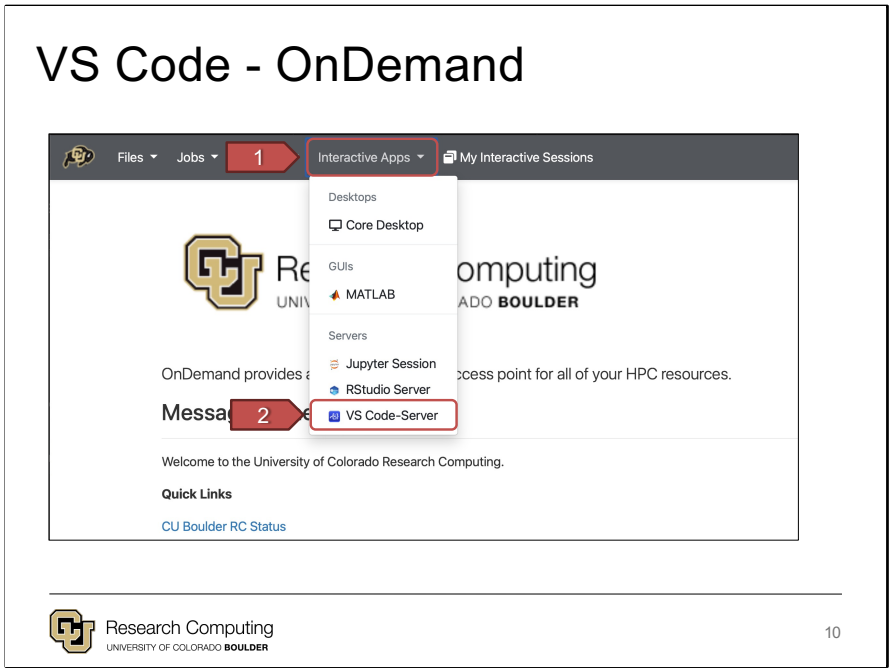


This presentation covers the basics of using VS Code's (Visual Studio Code) built-in debugger and is divided into three parts:

(1) Demonstrate how to start a VS Code Server through OnDemand - https://ondemand.rc.colorado.edu/pun/sys/dashboard

(2) Provide an overview of VS Code's user interface

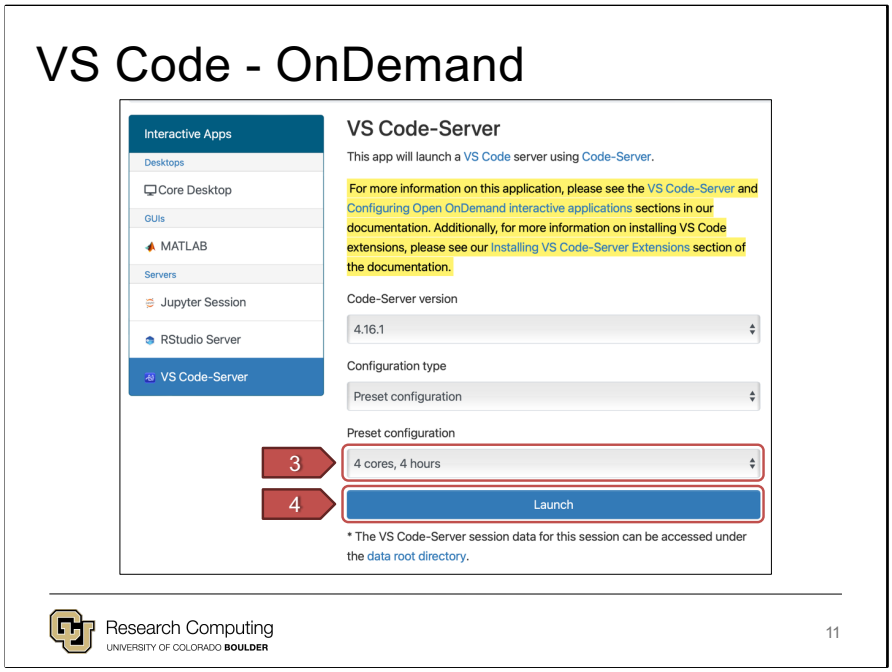
(3) Explain how to prepare and then use the debugger on a Python program



(1) Open the Interactive Apps dropdown (Top Menu Bar)

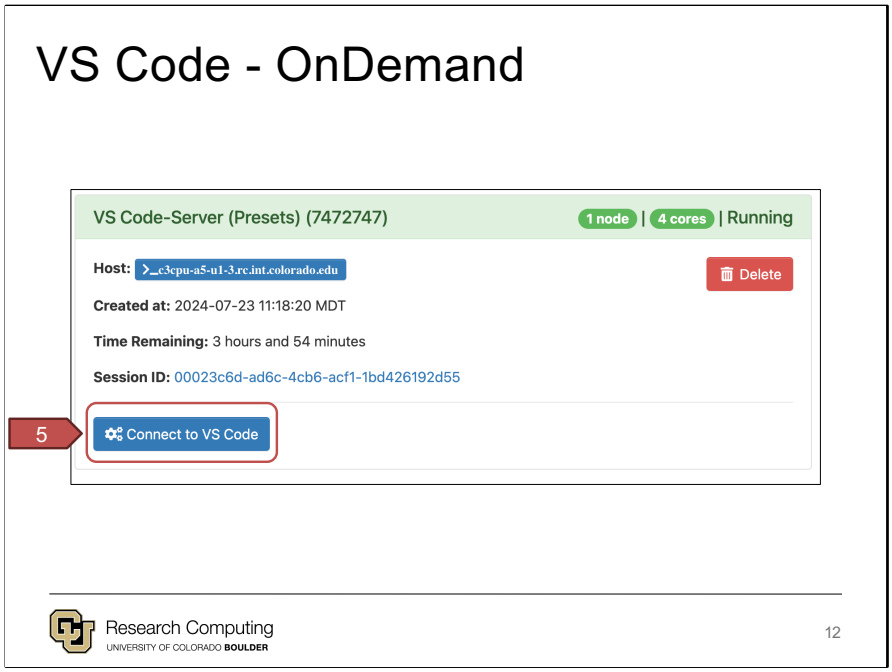
(2) Select “VS Code-Server”

Note:
VS Code OnDemand Documentation:
https://curc.readthedocs.io/en/latest/gateways/OnDemand.html#vs-code-server



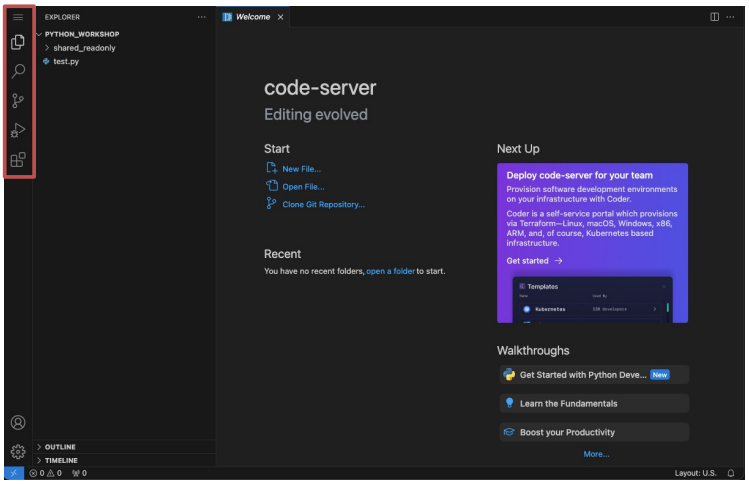
(3) Select your preferred configuration [4 cores, 4 hours]

(4) Launch the VS Code Server



(5) Once the server has started, click “Connect to VS Code”

VS Code - OnDemand



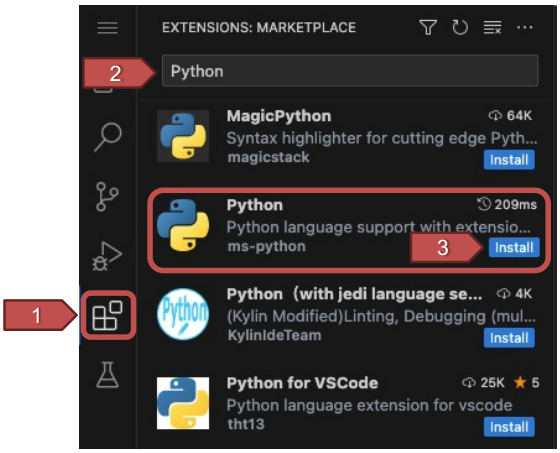
Research Computing
UNIVERSITY OF COLORADO BOULDER

13

To debug a python program, VS Code will need to have the Python Extension installed.

Extensions can be found by clicking the stack of blocks in the top-left of the user interface.

VS Code – Extensions



(1) Click the “Extensions” tab
(2) Type “Python” into search provided search bar.
(3) Click “Install” next to the Python extension

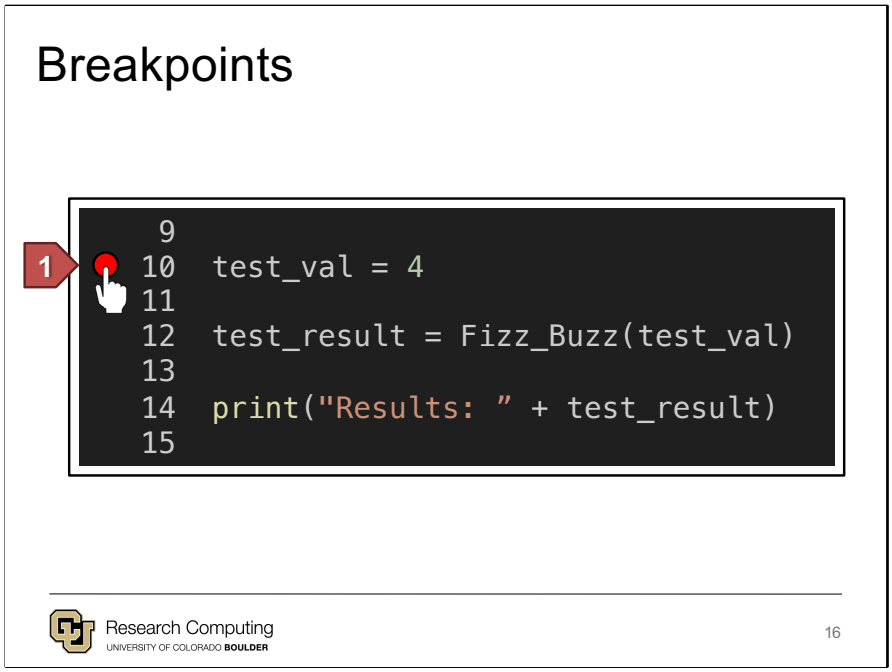
Example Program

FizzBuzz.py

```
1  def Fizz_Buzz(number_to_test):
2          result = ""
3          if(number_to_test % 2 == 0):
4                  result += "Fizz"
5
6          if(number_to_test % 3 == 0):
7                  result += "Buzz"
8          return(result)
9
10 test_val = 4
11
12 test_result = Fizz_Buzz(test_val)
13
14 print("Results: " + test_result)
15
```


Copy+paste the FizzBuzz.py's code into a new file in your VS Code session.

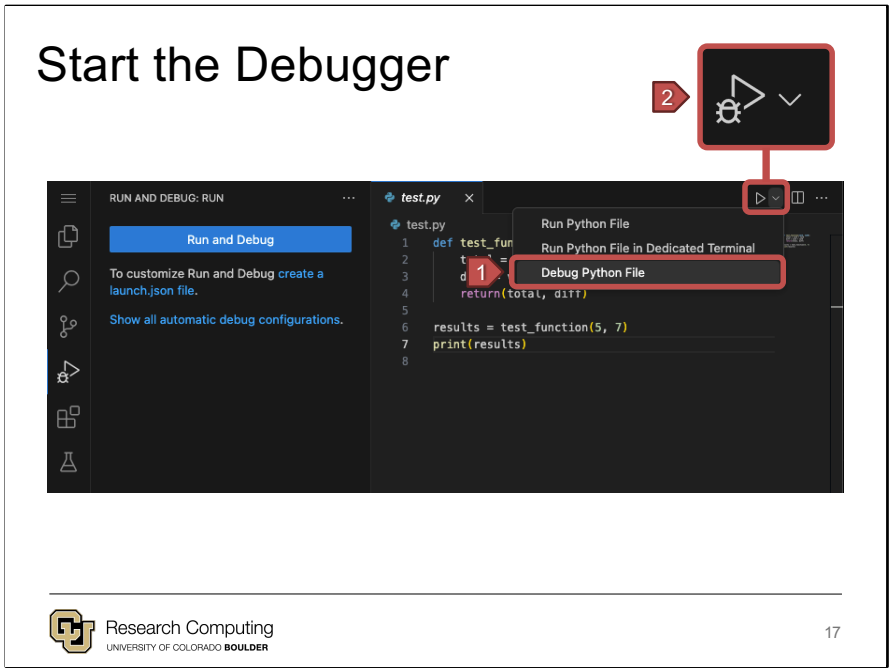
This file is also provided in this presentations Github Repo: https://github.com/ResearchComputing/rmacc_2025/tree/main/debugging_with_vscode



Breakpoints can be used to temporarily pause a program's execution.

While paused, the state of the program's data (variables, registers, etc.) can be observed and even modified.

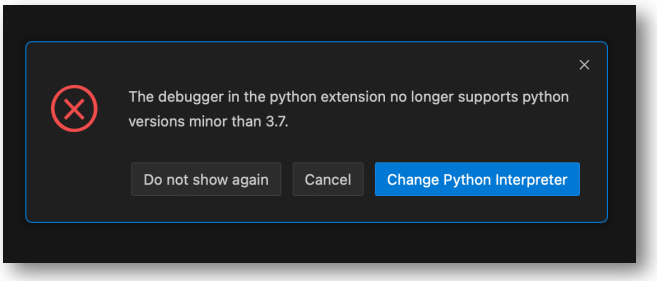
(1) Add a breakpoint on Line 10 test_val=4



(1) Click the dropdown and select “Debug Python File”

(2) The “Run” button should be replaced with the “Debug” button

Python Interpreter Error

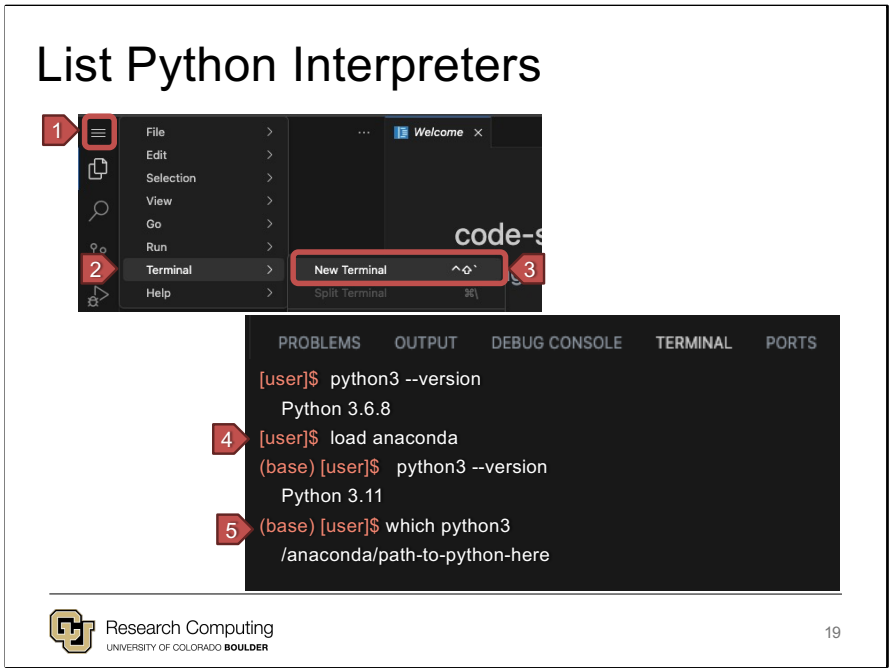


Research Computing
UNIVERSITY OF COLORADO BOULDER

18

The default Python Interpreter for the OnDemand (Viz) nodes is 3.68, which is not supported by the debugger.

If you haven't set the Python Extension to use a different interpreter (like Anaconda), then you will likely see this error message.

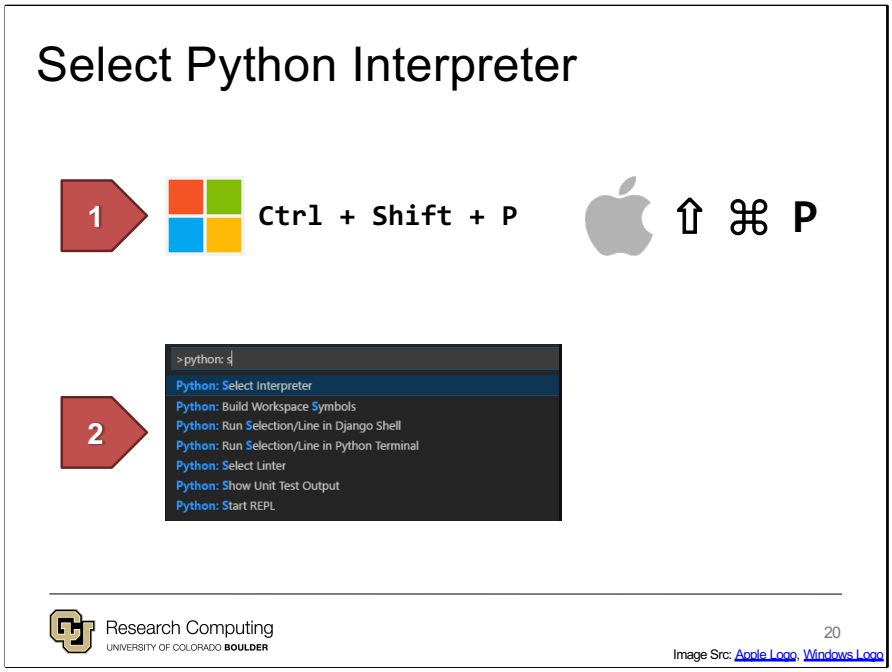


To change the Python Interpreter:
(1-3): Open a New Terminal

In the Terminal:
(4) Load anaconda (base environment)

(5) List the file path to the base Anaconda's python interpreter

Make sure to copy+paste the file path

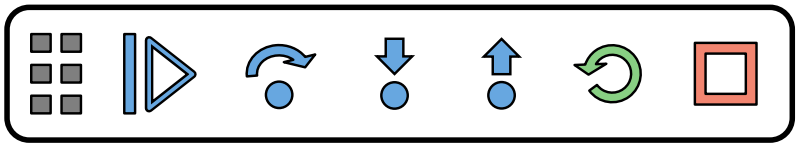


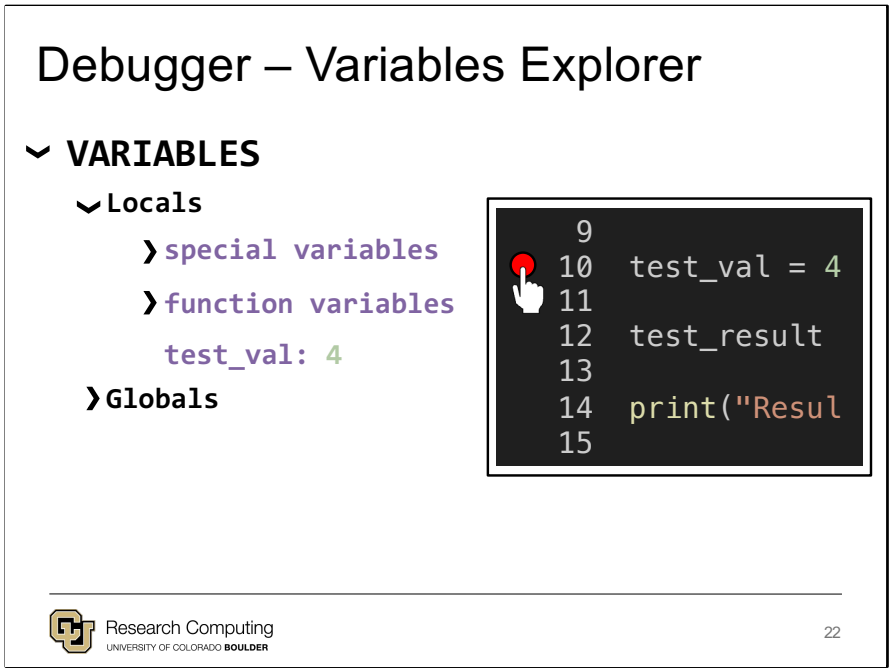
(1) Open the Command Pallete in VS Code (short cut keys for Windows (Left) and Mac (Right)

(2) Pick the python extensions “Select Interpreter” option.

Information on Selecting the Python Interpreter in VS Code: https://code.visualstudio.com/docs/python/environments#_working-with-python-interpreters

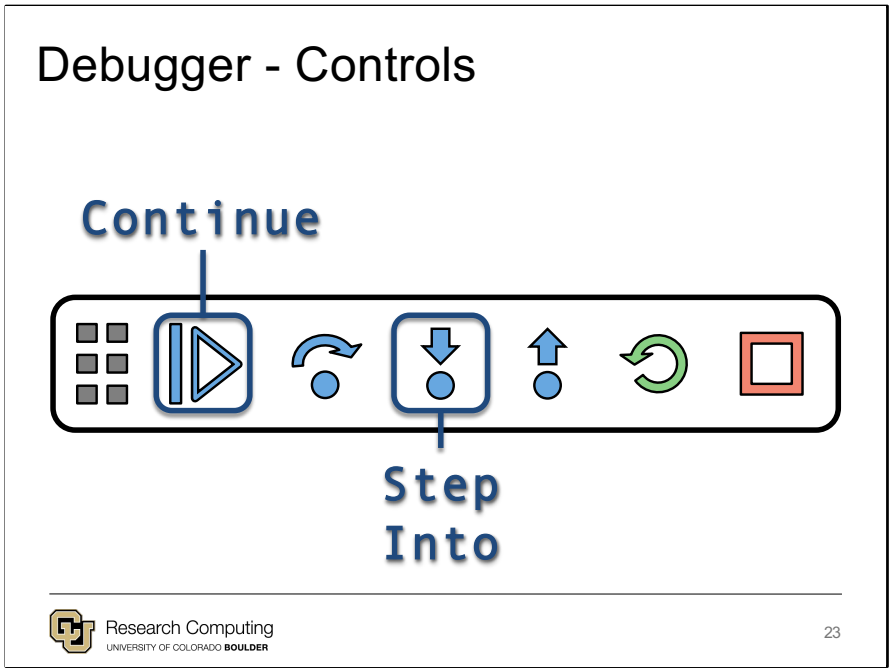
Debugger - Controls





While paused on line 10, the user should see a similar set of variables and values.

Demo: Show how the variables can be viewed and modified.

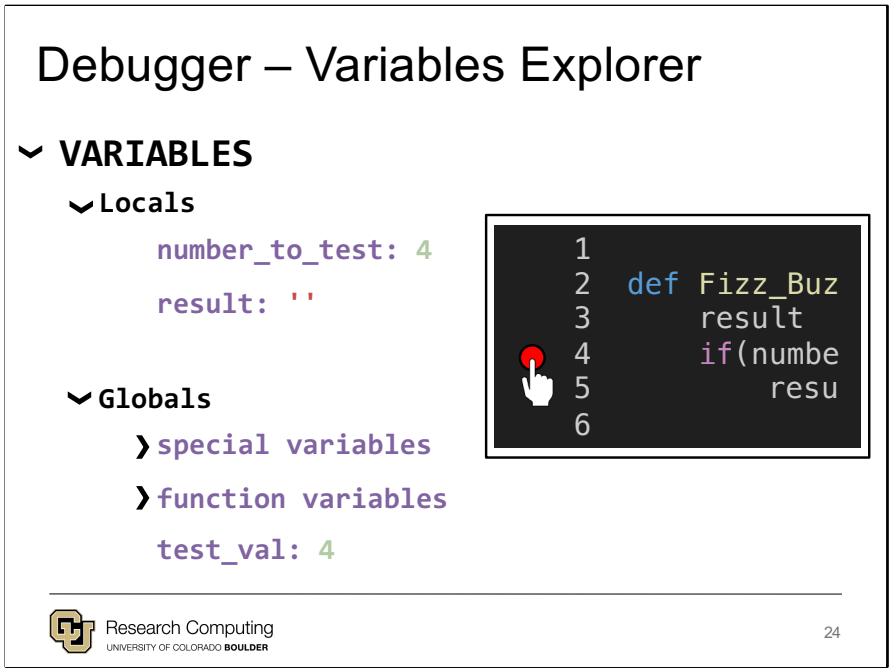


The “Continue” button will move from breakpoint to breakpoint, until reaching the end of the program’s execution (or an error is reached).

The “Step Into” button, enables users to observe the path of a program’s execution line-by-line, including “stepping into” functions called.

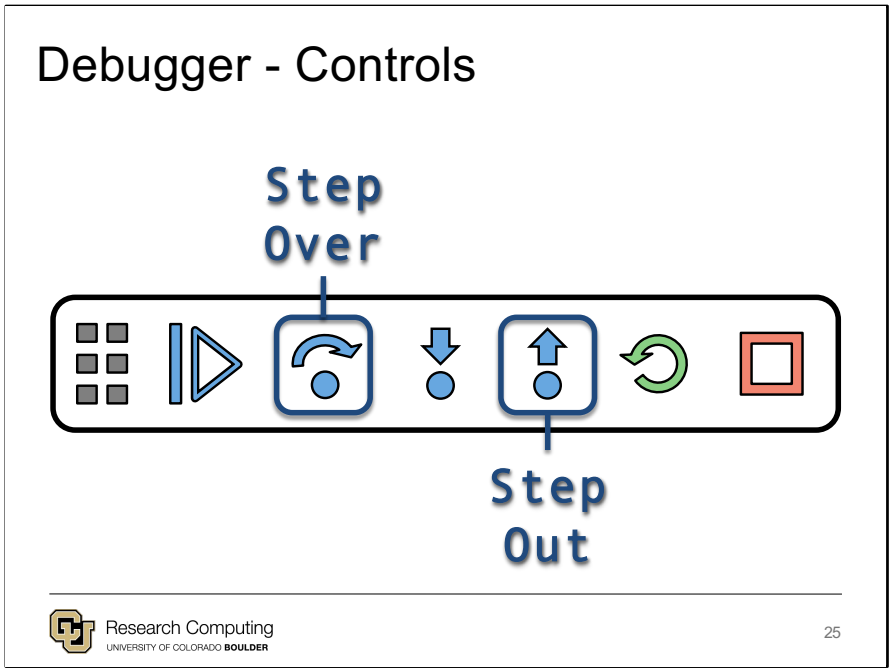
Demo: Show how each option affects the execution of the program.

Debugger – Variables Explorer



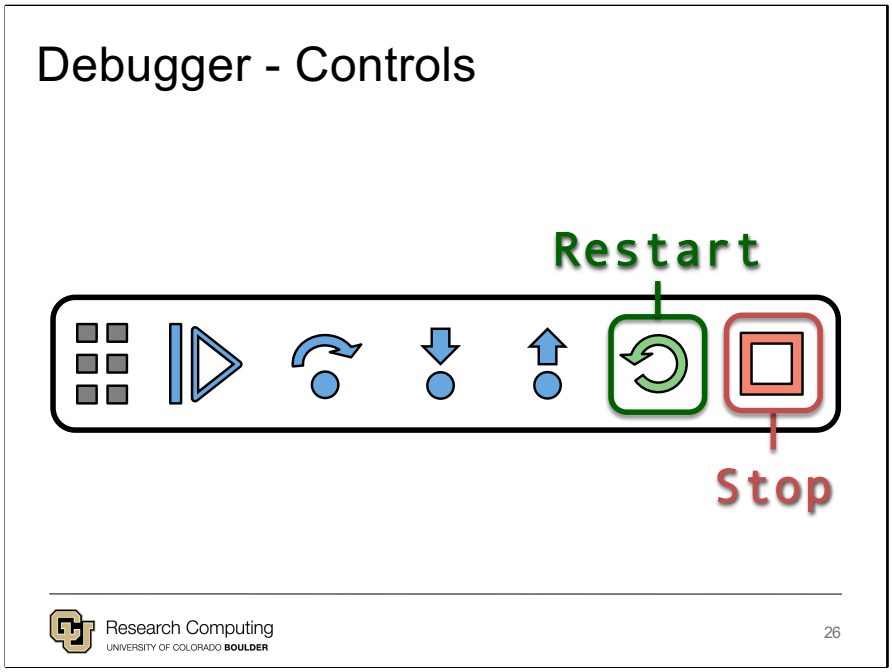
Once inside the “Fizz_Buzz” method, the variable display will update as shown.

Note how test_val is now listed as a ”Global” and the appearance of the method’s variables, “Local”



Step Over – This button enables for coding stepping that stays in the current scope – e.g. no stepping into a method call, loop, or conditional statement

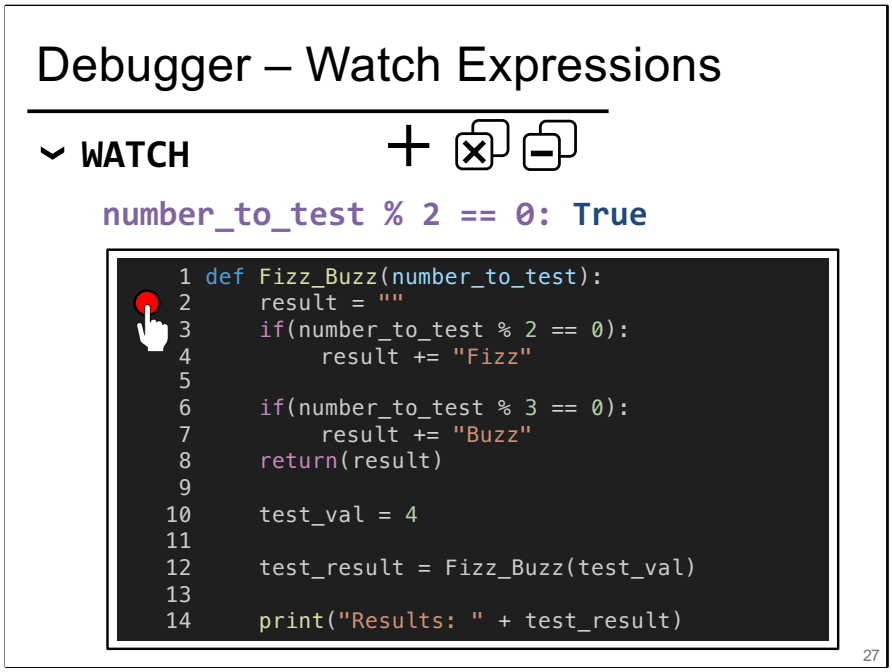
Step Out – This button continues code execution and pauses at the calling/higher level of scope – e.g. the point where a method was call, the end of a loop, or outside a conditional statement.



Restart – This button will end the program's execution and then start over

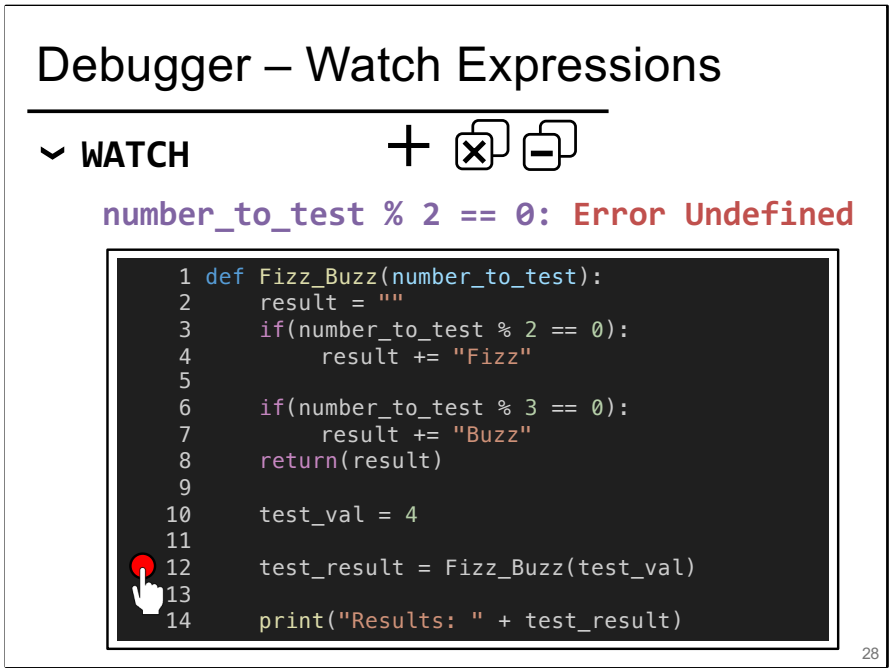
Stop – This will stop both the program's execution and the debugger

Key warning – Any changes made to variables within the debugger are only temporary. That's great for testing, but does mean the data will be gone when restarting or stopping a debugging session



Individual variables (e.g. an object's field) or programmatic expressions can be observed with the "Watch" feature.

The variables or expression being watched will be automatically updated, always providing an up-to-date view of system information.



Be mindful, you will often run into “Undefined” errors with watchlists because the variables used are no longer in-scope.

Next Steps – Check the Docs!

More Breakpoints!

- Conditional, Trigger, Function, Data, Logpoint

Testing Frameworks

- Automated testing of code

https://code.visualstudio.com/docs/debugtest/debugging

Questions or Comments?

Questions

Comments