

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

Debugging with VS Code

Instructor:

Research Computing

- Website: www.rc.colorado.edu
- Helpdesk: rc-help@colorado.edu

Slides:

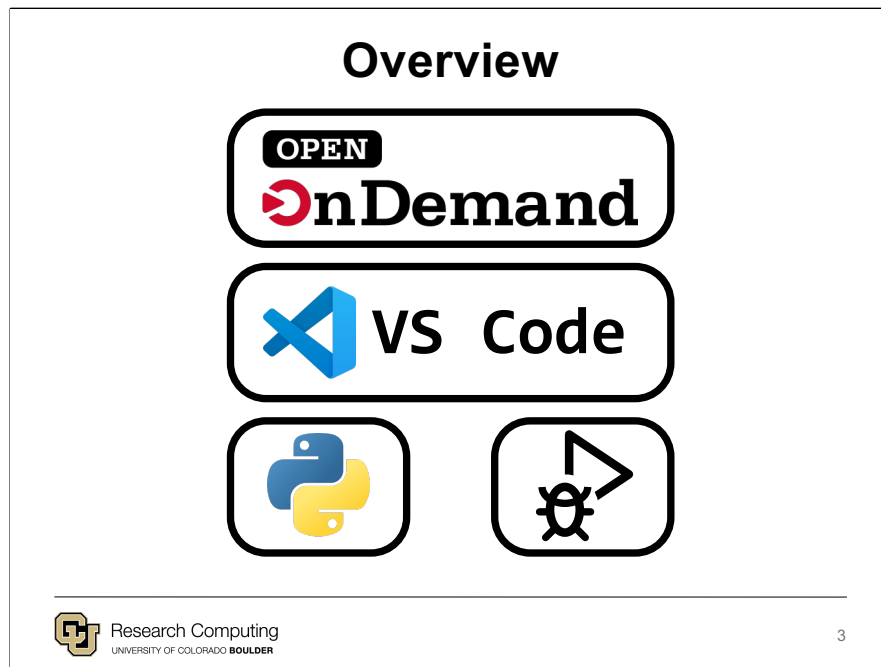
<https://github.com/ResearchComputing/debugging-with-vs-code-shortcourse>

Survey:

<http://tinyurl.com/curc-survey18>



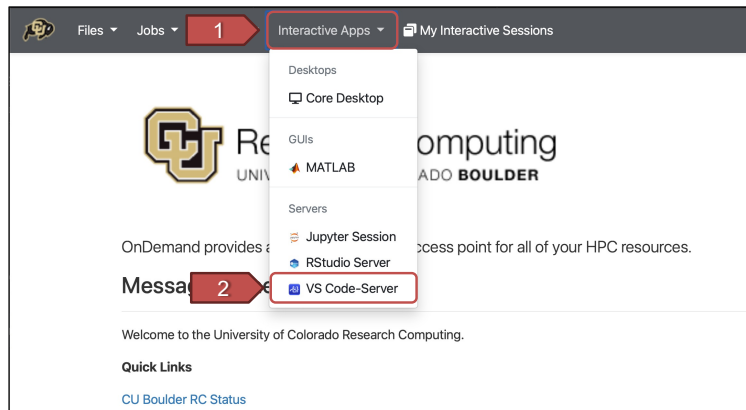
Research Computing
UNIVERSITY OF COLORADO BOULDER



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

VS Code - OnDemand



(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>

VS Code - OnDemand

The screenshot shows the 'VS Code-Server' configuration page. On the left, a sidebar lists 'Interactive Apps' with categories: Desktops (Core Desktop), GUIs (MATLAB), Servers (Jupyter Session, RStudio Server, and VS Code-Server). The main panel is titled 'VS Code-Server' and contains the following fields:

- Code-Server version:** A dropdown menu showing '4.16.1'.
- Configuration type:** A dropdown menu showing 'Preset configuration'.
- Preset configuration:** A dropdown menu showing '4 cores, 4 hours'.
- Launch:** A blue button to start the session.

Two red arrows with numbers indicate the steps: Arrow 3 points to the 'Preset configuration' dropdown, and Arrow 4 points to the 'Launch' button. A note at the bottom states: '* The VS Code-Server session data for this session can be accessed under the data root directory.'

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

(4) Launch the VS Code Server

VS Code - OnDemand

The screenshot displays the VS Code OnDemand interface. At the top, a green header bar shows 'VS Code-Server (Presets) (7472747)' and '1 node | 4 cores | Running'. Below this, the 'Host' is listed as 'c3cpu-a5-u1-3.rc.int.colorado.edu' with a 'Delete' button. The 'Created at' time is '2024-07-23 11:18:20 MDT' and the 'Time Remaining' is '3 hours and 54 minutes'. The 'Session ID' is '00023c6d-ad6c-4cb6-acf1-1bd426192d55'. A red arrow with the number '5' points to a blue button labeled 'Connect to VS Code' which is also enclosed in a red rectangular box.

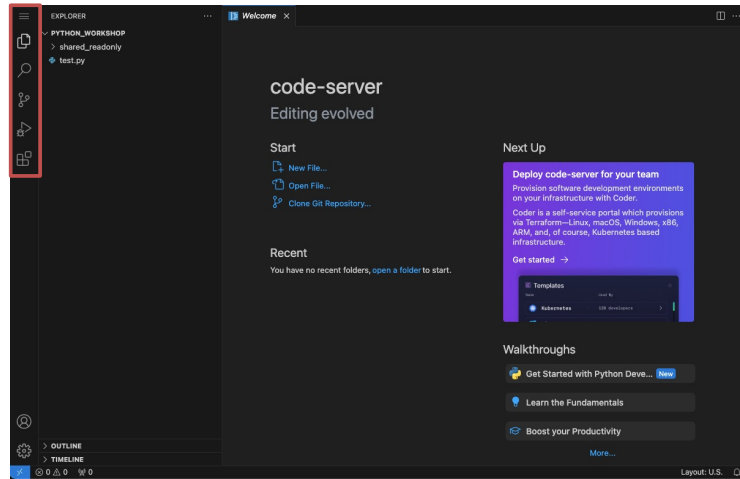


Research Computing
UNIVERSITY OF COLORADO BOULDER

6

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

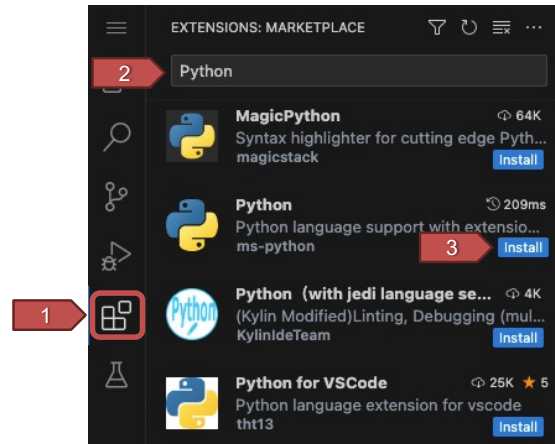
VS Code - OnDemand



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
```



Research Computing
UNIVERSITY OF COLORADO BOULDER

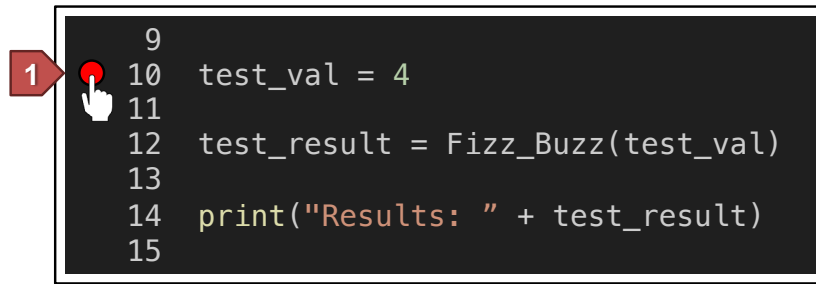
9

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/debugging_with_vs_code_shortcourse

Breakpoints

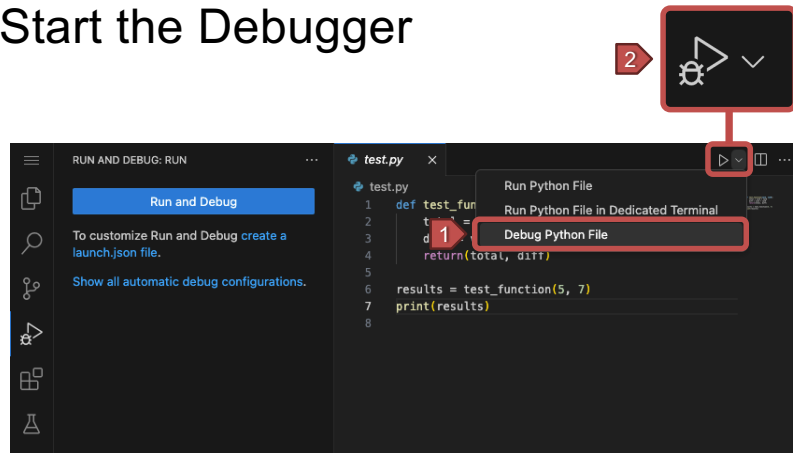


Breakpoints can be used to temporarily pause a program's execution.

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

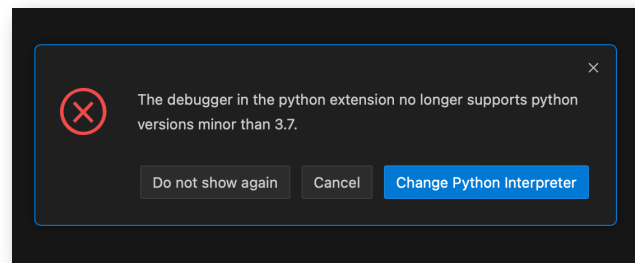
(1) Add a breakpoint on Line 10 `test_val=4`

Start the Debugger



- (1) Click the dropdown and select “Debug Python File”
- (2) The “Run” button should be replaced with the “Debug” button

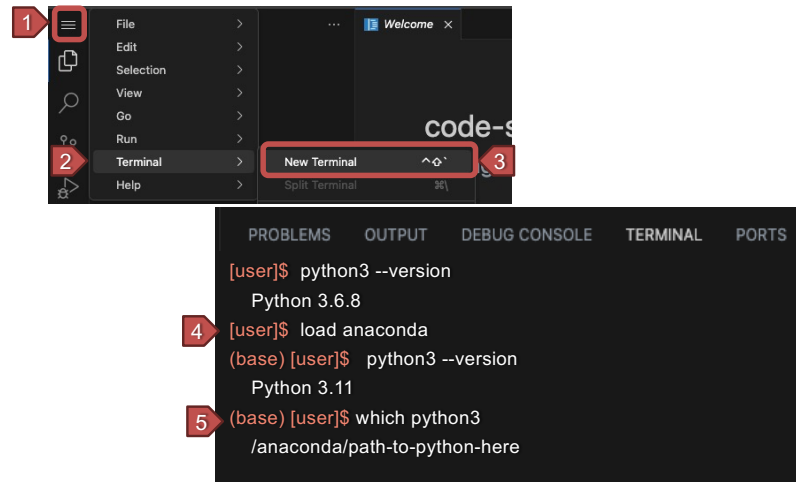
Python Interpreter Error



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.

List Python Interpreters



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

Select Python Interpreter

1

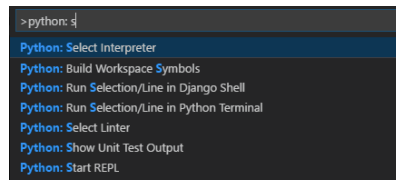


Ctrl + Shift + P



P

2



Research Computing
UNIVERSITY OF COLORADO BOULDER

14

Image Src: [Apple Logo](#), [Windows Logo](#)

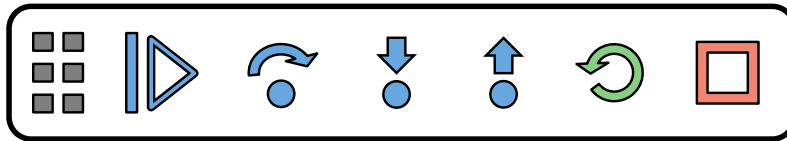
(1) Open the Command Palette in VS Code (short cut keys for Windows (Left) and Mac (Right))

(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



Debugger – Variables Explorer

✓ VARIABLES

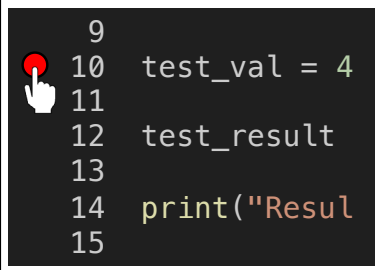
✓ Locals

› special variables

› function variables

test_val: 4

› Globals



```
9
10 test_val = 4
11
12 test_result
13
14 print("Resul
15
```



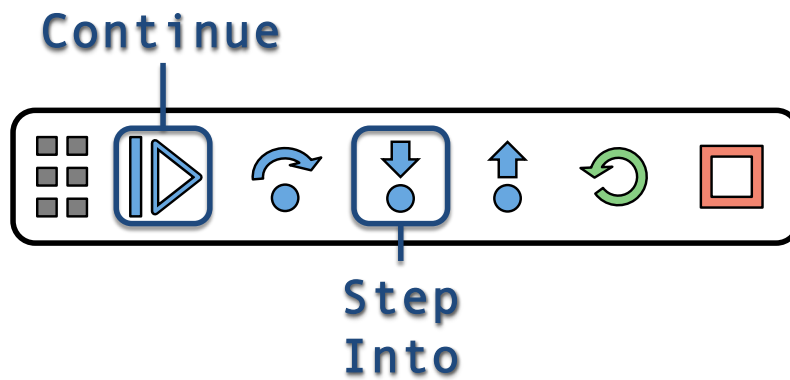
Research Computing
UNIVERSITY OF COLORADO BOULDER

16

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.

Debugger - Controls



Research Computing
UNIVERSITY OF COLORADO BOULDER

17

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

✓ VARIABLES

✓ Locals

number_to_test: 4

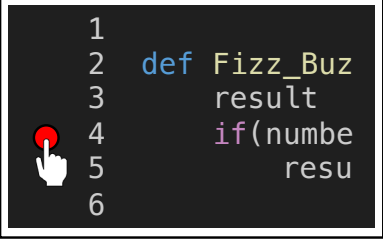
result: ''

✓ Globals

› special variables

› function variables

test_val: 4



```
1
2 def Fizz_Buz
3     result
4     if(numbe
5         resu
6
```



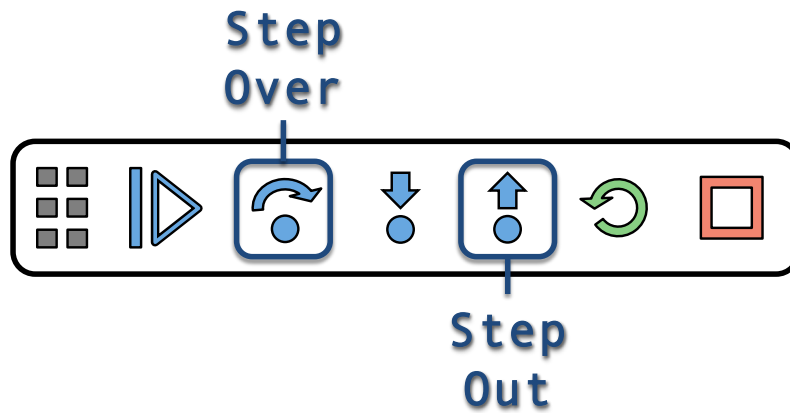
Research Computing
UNIVERSITY OF COLORADO BOULDER

18

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”

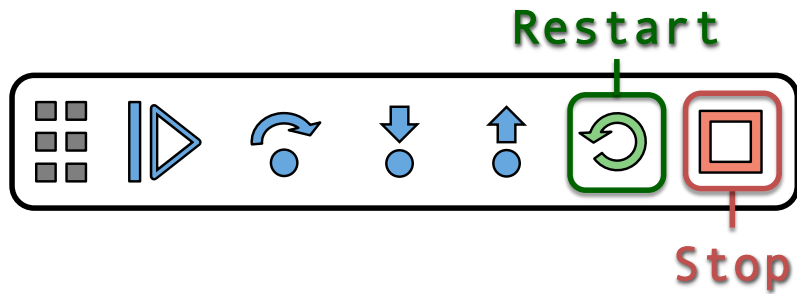
Debugger - Controls



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.

Debugger - Controls



Research Computing
UNIVERSITY OF COLORADO BOULDER

20

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

Debugger – Watch Expressions

✓ WATCH



`number_to_test % 2 == 0: True`



```
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)
```

21

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.

Debugger – Watch Expressions

✓ WATCH



`number_to_test % 2 == 0`: Error Undefined

```
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)
```

22

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