

EN-US



Using the Debugger

Lab overview

A *software bug* refers to an error, flaw, or failure in a computer program that causes an incorrect or unexpected result. A *debugger* is a computer program that is used to test and find bugs (debug) other programs. You can use a debugger to step through the code. The Python Debugger (pdb) is an interactive source code debugger for Python programs. In this lab, you will use the pdb to step through the scripts you wrote in previous labs.

In this lab, you will:

- Explore the basic features of the Python Debugger
- Use the Python Debugger to step through Python scripts

Estimated completion time

30 minutes

Accessing the AWS Cloud9 IDE

1. Start your lab environment by going to the top of these instructions and choosing **Start Lab**.

A **Start Lab** panel opens, displaying the lab status.

2. Wait until you see the message *Lab status: ready*, and then close the **Start Lab** panel by choosing the X.
3. At the top of these instructions, choose **AWS**.

Activate Windows
Go to Settings to activate Windows.

EN-US

3. At the top of these instructions, choose **AWS**.

The AWS Management Console opens in a new browser tab. The system automatically logs you in.

Note: If a new browser tab does not open, a banner or icon at the top of your browser typically indicates that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and choose **Allow pop ups**.

4. In the AWS Management Console, choose **Services > Cloud9**. In the **Your environments** panel, locate the **reStart-python-cloud9** card, and choose **Open IDE**.

The AWS Cloud9 environment opens.

Note: If a pop-up window opens with the message *.c9/project.settings have been changed on disk*, choose **Discard** to ignore it. Likewise, if a dialog window prompts you to *Show third-party content*, choose **No** to decline.

Creating your Python exercise File

5. From the menu bar, choose **File -> New from template -> Python File**.

This actions creates an untitled file.

6. Delete the sample code from the template file.

7. Choose **File -> Save As...**, provide a suitable name for the exercise file (for example, *debugger.py*), and save it under the **/home/ec2-user/environment** directory.

Note: The **.py** is the extension for Python files.

Activate Windows
Go to Settings to activate Windows.

EN-US



Accessing the terminal session

8. In your AWS Cloud9 IDE, choose the + icon and select **New Terminal**.

A terminal session opens.

9. To display the present working directory, enter `pwd`. This command points to **/home/ec2-user/environment**.

10. In this directory, locate the file that you created in the previous section.

Exercise 1: Exploring the basic features of the AWS Cloud9 Python Debugger

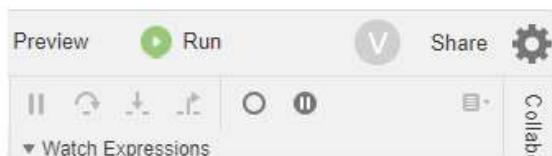
Cloud9 offers an interactive source code debugger for several languages, including Python. In this exercise, you cover some of the basic commands for debugging the **debugger.py** file.

Complete the following steps to explore the basic features of the Python Debugger.

11. From the navigation pane of the IDE, choose the **.py** file that you created in the previous *Creating your Python exercise file* section. Copy the following code and paste it in the file:

```
name = "John"
print("Hello " + name + ".")
age = 40
print(name + " is " + str(age) + " years old.")
```

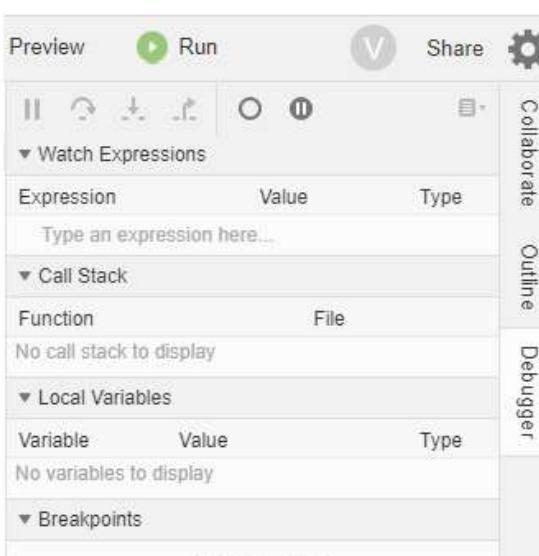
12. On the right of the interface, choose the **Debugger** tab to open the debugger.



Activate Windows
Go to Settings to activate Windows.

EN-US

12. On the right of the interface, choose the **Debugger** tab to open the debugger.



The screenshot shows the AWS Lambda console interface. At the top, there are tabs for Preview, Run, Share, and a gear icon. Below these are buttons for Pause, Stop, and a play/pause icon. A dropdown menu is open, showing options like Watch Expressions, Call Stack, Local Variables, and Breakpoints. To the right of the dropdown, there are tabs for Collaborate, Outline, and Debugger, with the Debugger tab currently selected. The main area displays sections for Watch Expressions, Call Stack, Local Variables, and Breakpoints, each with a table-like structure for viewing data.

13. Choose the gutter to the left of the number 1 to add a break point, and choose the gutter to the left of the number 4 to add another break point.

Your console should look similar to the following figure.



A screenshot of the AWS Lambda code editor window titled "17-debugger-1.p". The code is as follows:

```
1 name = "John"
2 print("Hello " + name + ".")
3 age = 40
4 print(name + " is " + str(age) + " years old.")
5
```

Two red circular markers are placed in the gutter to the left of the numbers 1 and 4, indicating set breakpoints.

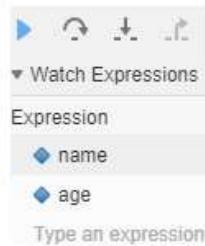
Activate Windows
Go to Settings to activate Windows.

14. In the **Debugger** window, add two watch expressions: **name** and **age**.



EN-US

14. In the **Debugger** window, add two watch expressions: **name** and **age**.

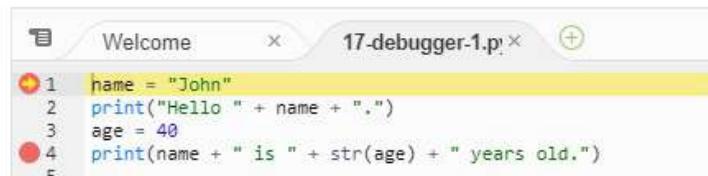


15. To run the program, choose the **Run** button. This opens a runner tab and runs the program.

16. On the runner tab, choose the **Run in Debug Mode** button.



17. Run the program again. The program stops at the break point.

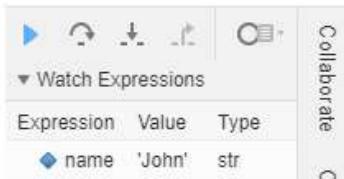


18. Choose the **Step Over** icon at the top of the **Debugger** window.

EN-US

18. Choose the **Step Over** icon at the top of the **Debugger** window.

19. Line 1 is run, and the value of the **name** variable is displayed in the **Debugger** window.



20. Choose the blue arrow at the top of the **Debugger** window. The program resumes and stops at line 4 where the other break point is set. The value of the **age** variable is displayed.



21. Choose the blue arrow at the top of the **Debugger** window to resume and end the program.

Exercise 2: Using the Python Debugger

Using the debugging basics you learned in Exercise 1, try stepping through some of the other labs to practice using the Python Debugger.

Congratulations! You have used some of the basic features of the Python Debugger.

Activate Windows
Go to Settings to activate Windows.

[End Lab](#)

EN-US



20. Choose the blue arrow at the top of the **Debugger** window. The program resumes and stops at line 4 where the other break point is set. The value of the **age** variable is displayed.

The screenshot shows the Python Debugger interface. At the top is a toolbar with icons for step, run, and stop. Below it is a dropdown menu. The main area is titled "Watch Expressions". A table lists two variables: "name" with value "'John'" and type "str", and "age" with value "40" and type "int".

Expression	Value	Type
name	'John'	str
age	40	int

21. Choose the blue arrow at the top of the **Debugger** window to resume and end the program.

Exercise 2: Using the Python Debugger

Using the debugging basics you learned in Exercise 1, try stepping through some of the other labs to practice using the Python Debugger.

Congratulations! You have used some of the basic features of the Python Debugger.

End Lab

Congratulations! You have completed the lab.

22. Choose **End Lab** at the top of this page, and then select Yes to confirm that you want to end the lab.

A panel indicates that *DELETE has been initiated... You may close this message box now.*

23. A message *Ended AWS Lab Successfully* is briefly displayed, indicating that the lab has ended.

Activate Windows
Go to Settings to activate Windows.