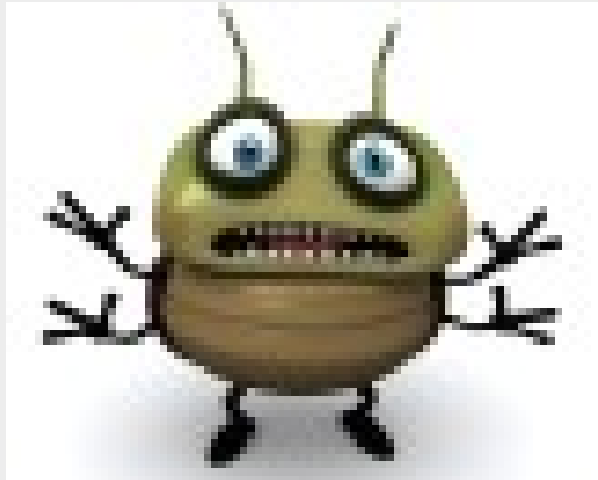
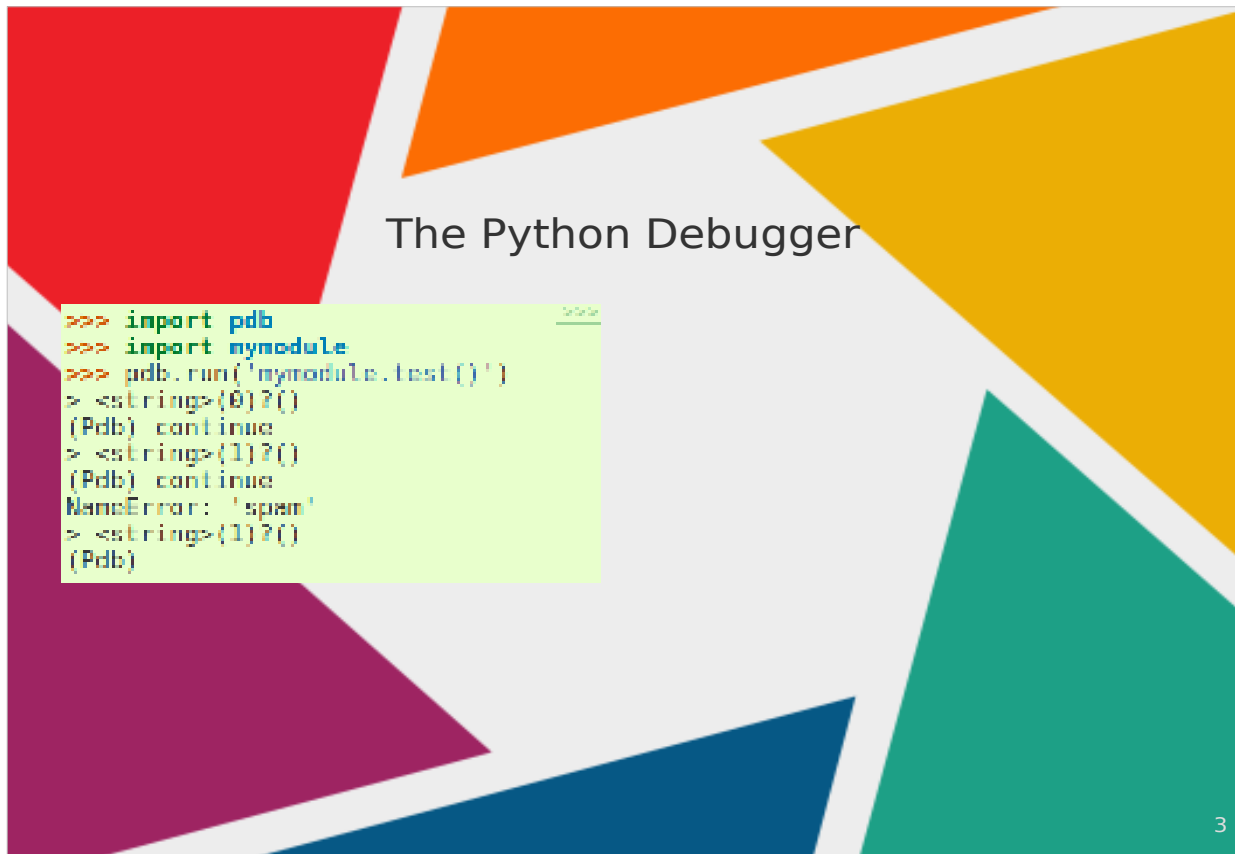


Debugging with PyCharm

What does it mean to debug a program?



To debug simply means to locate and remove program bugs, errors or abnormalities. It could be a period you put in the wrong spot or something you forgot to comment out.

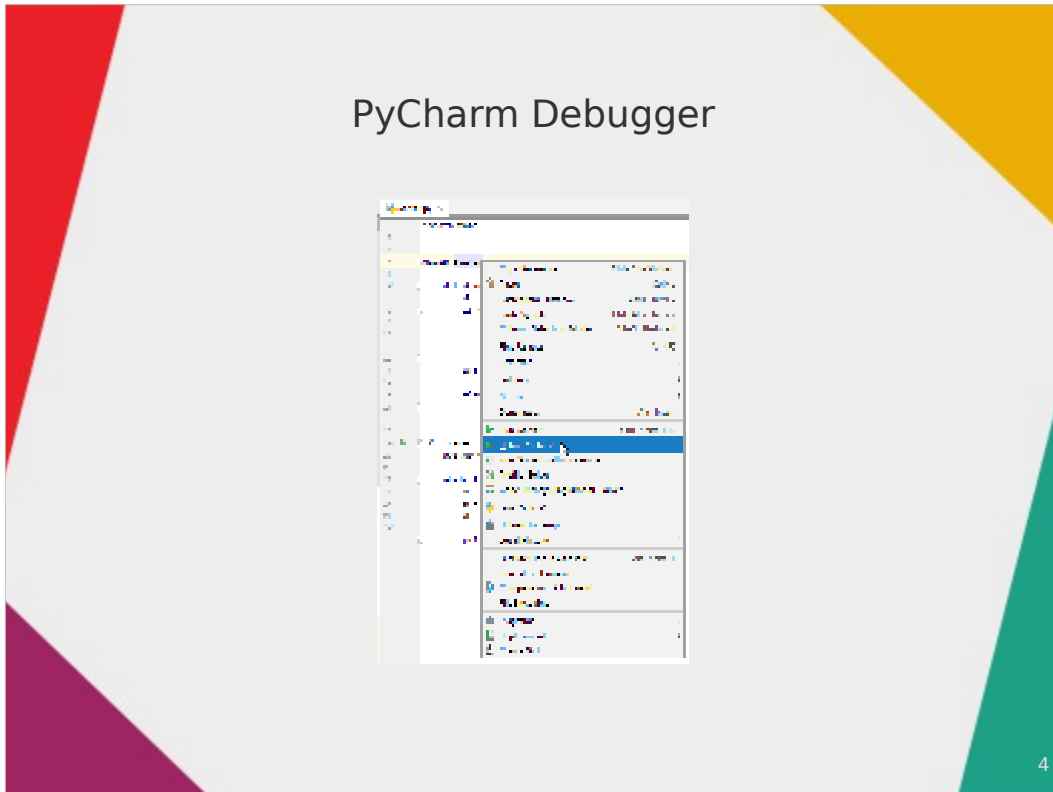


The Python Debugger

```
>>> import pdb
>>> import mymodule
>>> pdb.run('mymodule.test()')
> <string>(0)?()
(Pdb) continue
> <string>(1)?()
(Pdb) continue
NameError: 'spam'
> <string>(1)?()
(Pdb)
```

3

Python has its own built-in debugger called pdb. It can be called by an import statement. There is a python debugger for each version of python 2.7, 3.5, 3.6 etc.



In order to debug a program look to the top right of your pycharm IDE. There you'll see the name of your program, a play button to run your program and a green bug to the right of it.

BREAKPOINTS

During debugging sometimes we add breakpoints:

In software development, a breakpoint is an intentional stopping or pausing place in a program, put in place for debugging purposes. It is also sometimes simply referred to as a pause. More generally, a breakpoint is a means of acquiring knowledge about a program during its execution.

With PyCharm, users can create several types of breakpoints.

Python Line breakpoint: these breakpoints are assigned to lines of source code and are used to target a particular section for debugging.

Temporary Line breakpoint: These breakpoints are assigned to lines of source code and are used to target a particular section for debugging but once this breakpoint is hit the breakpoints are immediately removed.

There are also Django breakpoints, exception breakpoints, and javascript breakpoints

PyCharm supports debugging for Python and Django applications, classes, and files. The debugging functionality is incorporated in PyCharm, you only need to configure its settings.

Depending on the plugins enabled, PyCharm can also support debugging for other languages, for example, JavaScript.

The JavaScript debugging functionality is incorporated in PyCharm, you only need to configure its settings.

You can configure the debugger settings to support other languages: To configure settings required for debugging, perform the following general steps

In the Project Structure, configure the roots, dependencies and libraries to be passed to the interpreter.

In the Settings/Preferences dialog box, configure the debugger options:

Under the Build, Execution and Deployment section, click Debugger, and configure the debugger options.

Under the Build, Execution and Deployment section, click Python Debugger, and configure the Python debugger options.

Pausing and Resuming the Debugger Session

Introduction

When a breakpoint is hit, or when a running thread or an application is paused manually, the debugging session is suspended.

Pausing the debugger session

Do any of the following:

On the main menu, choose Run | Pause Program.

Click on the Debug toolbar.

Note that the button is not available for Run/Debug Configuration: Node.js, Run/Debug Configuration: Attach to Node.js/Chrome, and Run/Debug Configuration: NodeUnit.

Resuming the debugger session

Do any of the following:

On the main menu, choose Run | Resume Program.

Click on the Debug toolbar.

Monitoring the Debug Information

The information on a debugging session is displayed in the dedicated tabs of the Debug tool window named after the selected run/debug configuration.

For each session, use the Console tab to view the debugger messages and application output, and the Debug tab to monitor threads and frames.

python debugToolWindow

Monitor debugger overhead

The debug process is part of the runtime and, therefore, may impact performance. Every evaluation of an expression, or stepping over the code use the same memory as the debugged application, and may cause large overhead.

PyCharm lets you view this overhead so that you can quickly detect what causes it and reduce it by removing unnecessary breakpoints, disabling automatic evaluation of expressions, turning off async stacktraces, etc.

To invoke the Overhead pane, click the overhead view icon in the top-right corner of the Debug tool window:

Using a remote interpreter

Configuring a remote interpreter on Windows

Note that any remote interpreter will do.

Click Ctrl+Alt+S to open the Settings dialog on the Windows machine (the whole process is described in the section [Accessing Settings](#)).

Next, click the Project Interpreter node. On this page, click the gear button (gear icon):

pygear button

Then choose the remote interpreter:

pyremote interpreter choose

Next, in the Configure Remote Python Interpreter dialog box, click the Deployment Configuration radio-button:

Just a side note...If you click OK in the Configure Remote Python Interpreter dialog box, then the system interpreter `/usr/bin/python` specified in the Python interpreter path field, will be used. However, if you click `browseButton` next to the Python interpreter path field, you can specify, say, virtual environment, that has been created on the remote computer beforehand.

py remote interpreter via deployment configuration
Then click the browse button (browseButton) next to the
Deployment configuration field.

The Add server dialog box opens. There, enter the server
name (let it be MySFTPConnection) and choose the server
type from the drop-down list. Select SFTP to tell PyCharm
to transfer files over the SSH connection. Next, the
deployment settings dialog opens for MySFTPConnection.
In the

<https://www.jetbrains.com/help/pycharm/working-with-run-debug-configurations.html>