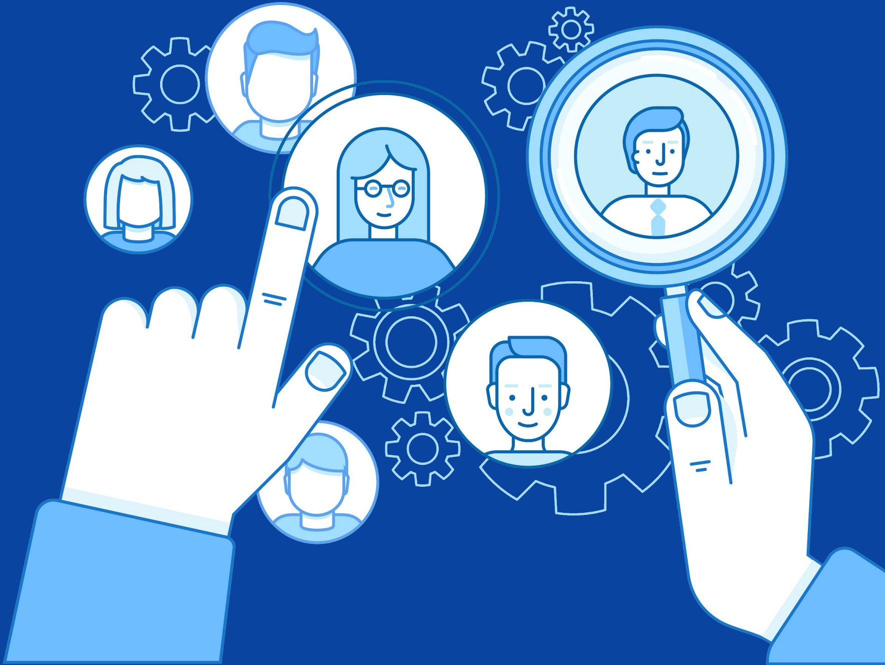
Digital Career Institute

Python Course - Debugging



Digital Career Institute

Goal of the Submodule

The goal of this submodule is to help the learners debug code. By the end of this submodule, the learners should be able to understand

- How to understand and interpret stack trace.
- How to use **pdb** (Python debugger).
- Using breakpoints.
- Reading variables.

Topics

- What is debugging
- Rubber duck debugging.
- Print statement as a debugger.
- Interpreting StackTrace.
- Introduction to the **`pdb`** module in Python.
- Using the **`breakpoint`** module in Python 3.7 and above.

Glossary

Term	Definition
Stacktrace or backtrace	It is a list of function calls and statements that a program was in the middle of executing when an error occurs.
Debug	A process of identifying and removing errors from a program.

What is debugging ?

Digital Career Institute

Debugging - What it is?

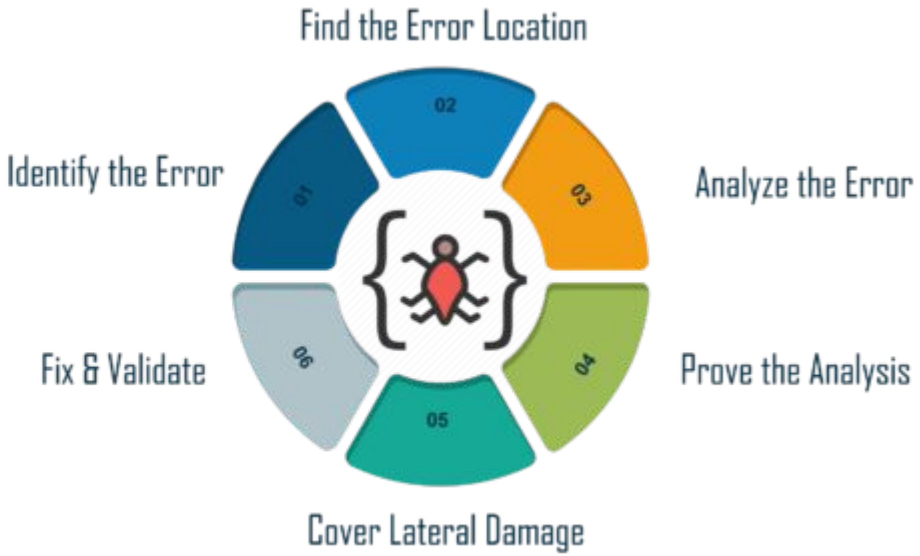
- The terms "bug" and "debugging" are popularly attributed to Admiral Grace Hopper in the 1940s. While she was working on a Mark II computer at Harvard University, her associates discovered a moth stuck in a relay and thereby impeding operation, whereupon she remarked that they were "debugging" the system

What it is Debugging?

https://www.youtube.com/watch?v=lRYs9s-QK8k&ab_channel=FunRobotics

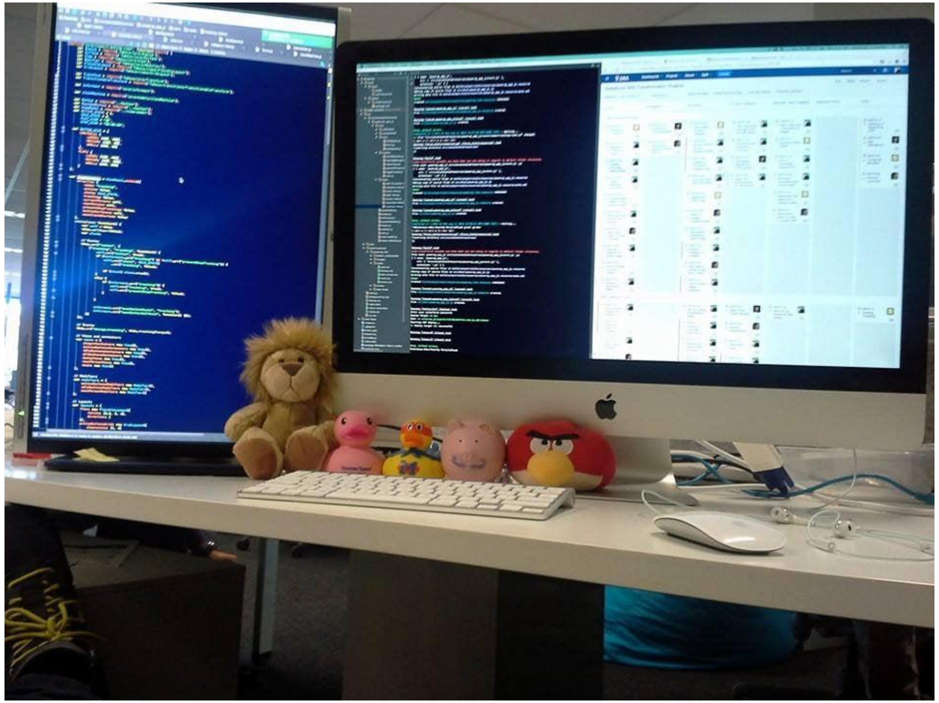
What it is Debugging?

Digital Career Institute



Rubber Duck Debugging

Rubber Duck Debugging - What it is?



- Try to explain your code to another person preferably a non-programmer or an inanimate object (like a Rubber duck)

Rubber Duck Debugging - What it is?

Digital Career Institute

Here's an example of how a programmer might use rubber duck debugging:

- The programmer encounters a problem in their code.
- They take a rubber duck or other inanimate object and place it on their desk.
- The programmer explains the code or problem to the rubber duck in detail, step by step.
- As they explain the code, the programmer may notice issues in their logic or identify the root cause of the problem.
- After finishing the explanation, the programmer can make changes to the code to fix the problem.

Debugging with print Statement

Debugging with print Statement

- We can use print statements to test our hypothesis about why a program may not be working a.k.a. print - debugging.
- There are more professional ways than print, but let's have a quick recap on how to do print debugging.

Debugging with print Statements

```
def sum_of_list(list):
    print(list)

    flatten_list = []

    for item in list:

        print(item)

        flatten_list.append(item)

    return sum(flatten_list)
```

Printing variables to see values

1. If you suspect the entire list might have a problem you can print it.

2. You can print another area you suspect could have a problem in our for loop.

Problems with print Statements

- Not a lot of context into the program is provided.
- It is more tedious to keep rerunning your code, moving print statements about as you look for the error.
- The print statement could also be your bug.

Debugging with Stack Trace

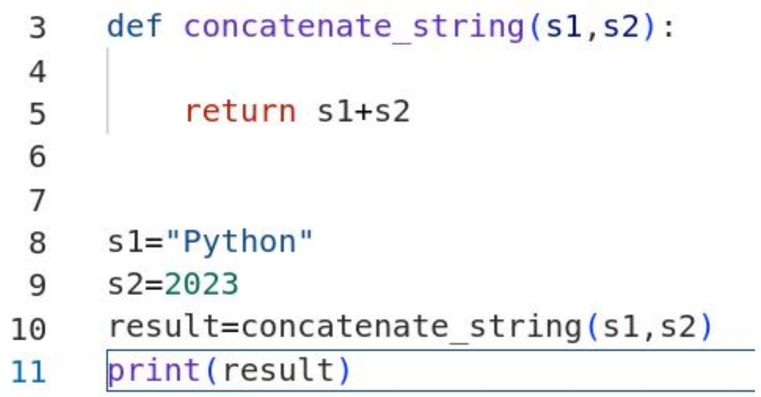
Digital Career Institute

Debugging with stack trace

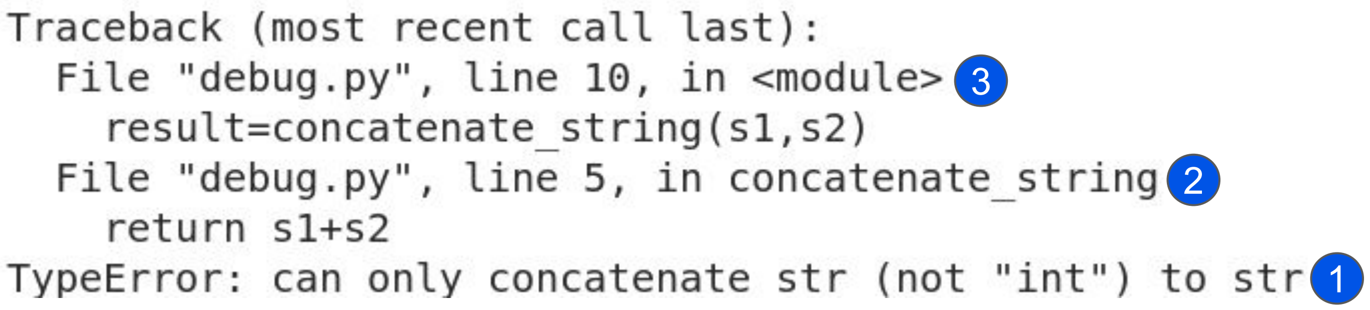
- In Python, a stack trace (also known as a traceback) is a report that is generated automatically when an exception occurs during program execution.

- The stack trace shows the sequence of function calls that led up to the error. It includes the names of the functions, their line numbers, and the file names where the functions were defined.

Debugging with stack trace



```
3    def concatenate_string(s1,s2):
4
5        return s1+s2
6
7
8    s1="Python"
9    s2=2023
10   result=concatenate_string(s1,s2)
11   print(result)
```



```
Traceback (most recent call last):
  File "debug.py", line 10, in <module> 3
    result=concatenate_string(s1,s2)
  File "debug.py", line 5, in concatenate_string 2
    return s1+s2
TypeError: can only concatenate str (not "int") to str 1
```

Stack trace advantages vs disadvantages

Digital Career Institute

Advantages	Disadvantages
● Stack trace can be very helpful when it comes to small programs.	● Stack trace can be difficult to read when using complex programs with nested functions.
● You can quickly identify the error.	● Stack traces can reveal sensitive information about your code or system.
● provide a detailed record of the sequence of function calls.	● In some cases, stack traces can be misleading or inaccurate.

Debugging with pdb

How to start the debugger

- PDB stands for "Python Debugger," and it's a powerful tool for debugging Python code.
- The major advantage of pdb is it runs purely in the command line, thereby making it great for debugging code on remote servers when we don’t have the privilege of a GUI-based debugger.
- pdb supports:
  - Setting breakpoints
  - Stepping through code
  - Source code listing
  - Viewing stack traces

How to start the debugger

1. Using a python statement that sets the breakpoint.

2. Starting from the command line.

```
python -m pdb <filename>.py
```

Your output is shown below

```
  /path-to-folder/main.py(34)<module>()
-> def sum_array(list):
(pdb)
```

Debugging with Breakpoint within the Script

```
def sum_of_list(list):
    import pdb; pdb.set_trace()

    flatten_list = []

    for item in list:

        flatten_list.append(item)

    return sum(flatten_list)
```

1

Use **pdb** if Python is below version 3.7

1. If you suspect the **`sum_of_list`** function might have a problem, add the **`import`** statement and execute as shown in example.

Debugging with Breakpoint within the Script

```
def sum_of_list(list):
    breakpoint()

    flatten_list = []

    for item in list:

        flatten_list.append(item)

    return sum(flatten_list)
```

`breakpoint` works for Python 3.7 +

1. If you suspect the list might have a problem, add a **`breakpoint()`** to the beginning of the function.

Using Interactive Python Debugger - ipdb

```
def sum_of_list(list):
    import ipdb; ipdb.set_trace()

    flatten_list = []

    for item in list:

        flatten_list.append(item)

    return sum(flatten_list)
```

1

ipdb benefits 1

- It offers highlighting
- Installed via pip
  - `$ pip install ipdb`

Changing your Debug Environment Defaults

- You can make **`ipdb`** your default debugger by setting it in your environment.
- You can use this in your **`bash`** or **`zsh`** profile setup

```
export PYTHONBREAKPOINT=ipdb.set_trace
```

Debugger Commands (Help/Exit)

method	Description
`h`	Without argument, print the list of available commands.
`q`	Quit from the debugger. The program being executed is aborted. You can also use `<Ctrl> + D` keys.

Debugger Commands (Stepping)

Digital Career Institute

method	Description
`n`	Continue execution until the next line in the current function is reached or it returns.
`s`	Execute the current line of code, stop at the first possible occasion either in the function that is called or on the next line in the current function.
`r`	Continue execution until current function returns a response or value.
`c`	Continue execution of your code and stop only when a another breakpoint is encountered.
`j <line_no>`	Skip to a specific line number in your code you want executed. *Tip: This can be a good time saver when used in loops.*

Debugger Commands (Frame navigation)

method	Description
`u`	Move the current frame count one level up in the stack trace (This is like navigating to the line before your current line of code)
`d`	Move the current frame count one level down in the stack trace

Debugger Commands (Display)

method	Description
`p <expression>`	Evaluate the value of an expression in the current context and print its value. For example, `p arg`
`w`	Print a stack trace, with the most recent frame at the bottom. See snippet below with active line at 26. An arrow will indicate the current frame. This determines the context of most commands. See indicator at 1

```
  /path-to-folder/main.py(34)<module>()
-> print(sum_array([1, 2, 3]))
>   /path-to-folder/main.py(26)sum_array()
-> flattened_list = []
```

1

Debugger Commands (Display)

method	Description
`pp`	Pretty print the value of an expression - add some formatting and styling.
`l`	List source code for the current file. Without any arguments, the list is 11 lines around the current line. A range can be provided too.
`a`	Print the argument list of the current function.

At the Core of the Lesson

Lessons learned:

- Commands to use when you are in the debug mode.
- Running programs in Debug mode

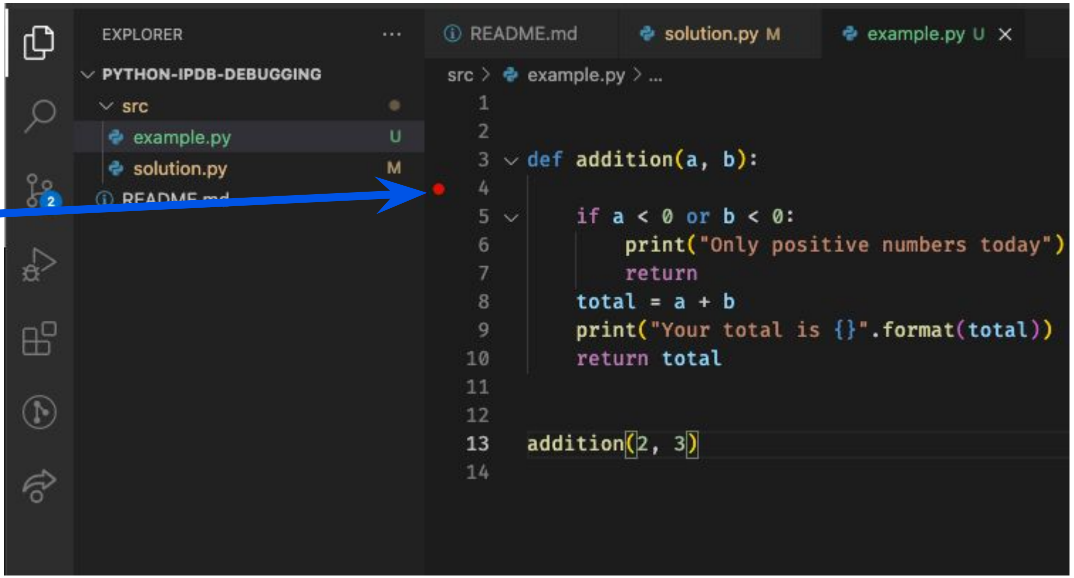
Digital Career Institute

Debugging using the VS Code IDE

Digital Career Institute

VS Code Break Points - Step 1

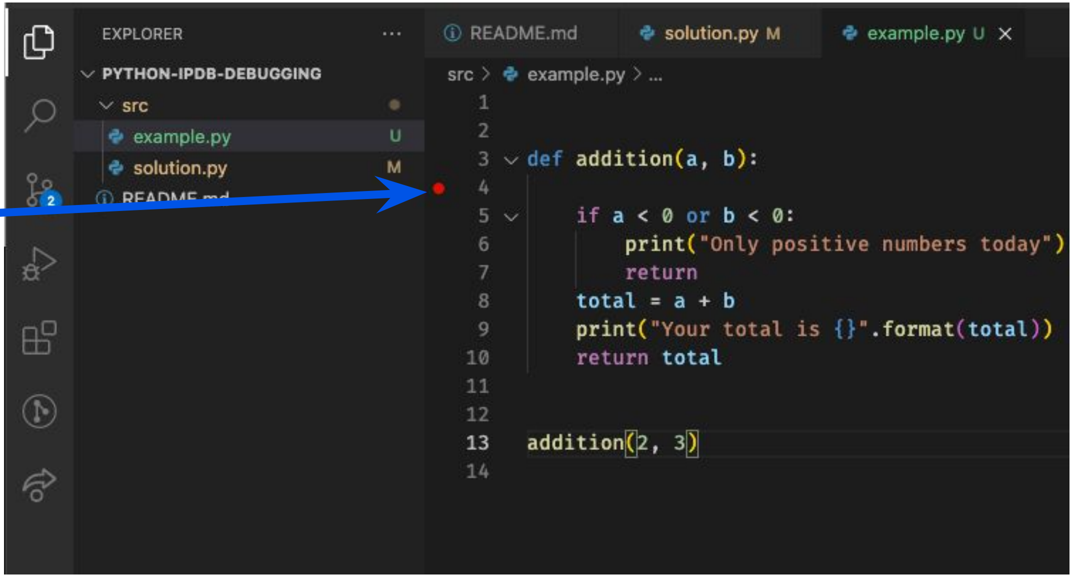
- Move your mouse closer to the line numbers on the gutter.
- A light red circle will show
- Click the circle that turns red in line of code you are interested in debugging.



```python


def addition(a, b):

    if a < 0 or b < 0:
        print("Only positive numbers today")
        return
    total = a + b
    print("Your total is {}".format(total))
    return total


addition(2, 3)

```

VS Code Break Points - Step 2

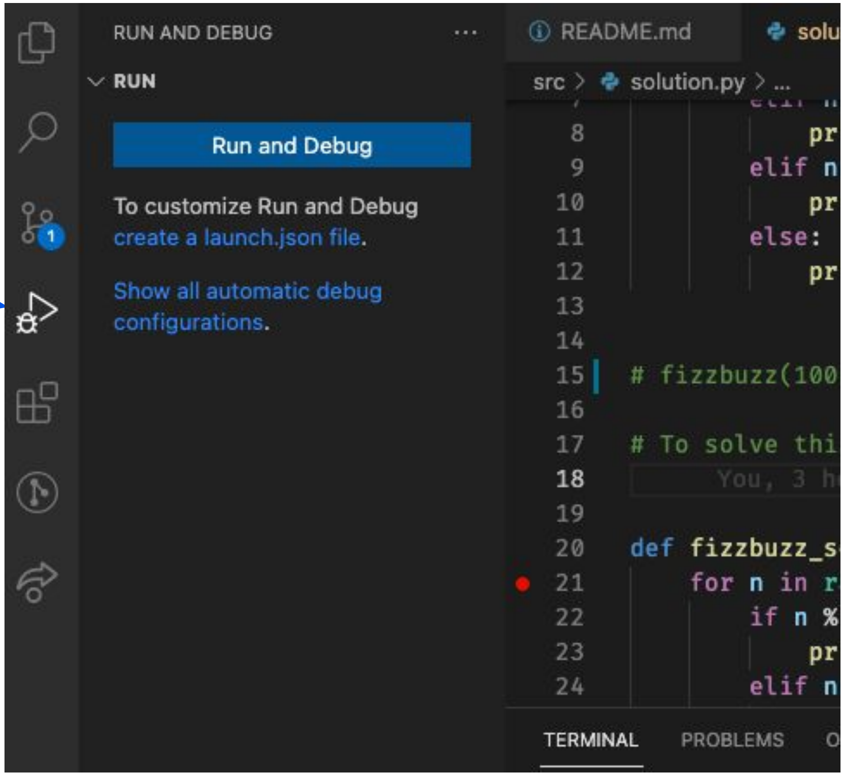
- After clicking the red circle, it will remain red until you disable it.



```python


def addition(a, b):

    if a < 0 or b < 0:
        print("Only positive numbers today")
        return
    total = a + b
    print("Your total is {}".format(total))
    return total


addition(2, 3)

```

VS Code Break Points - Step 3

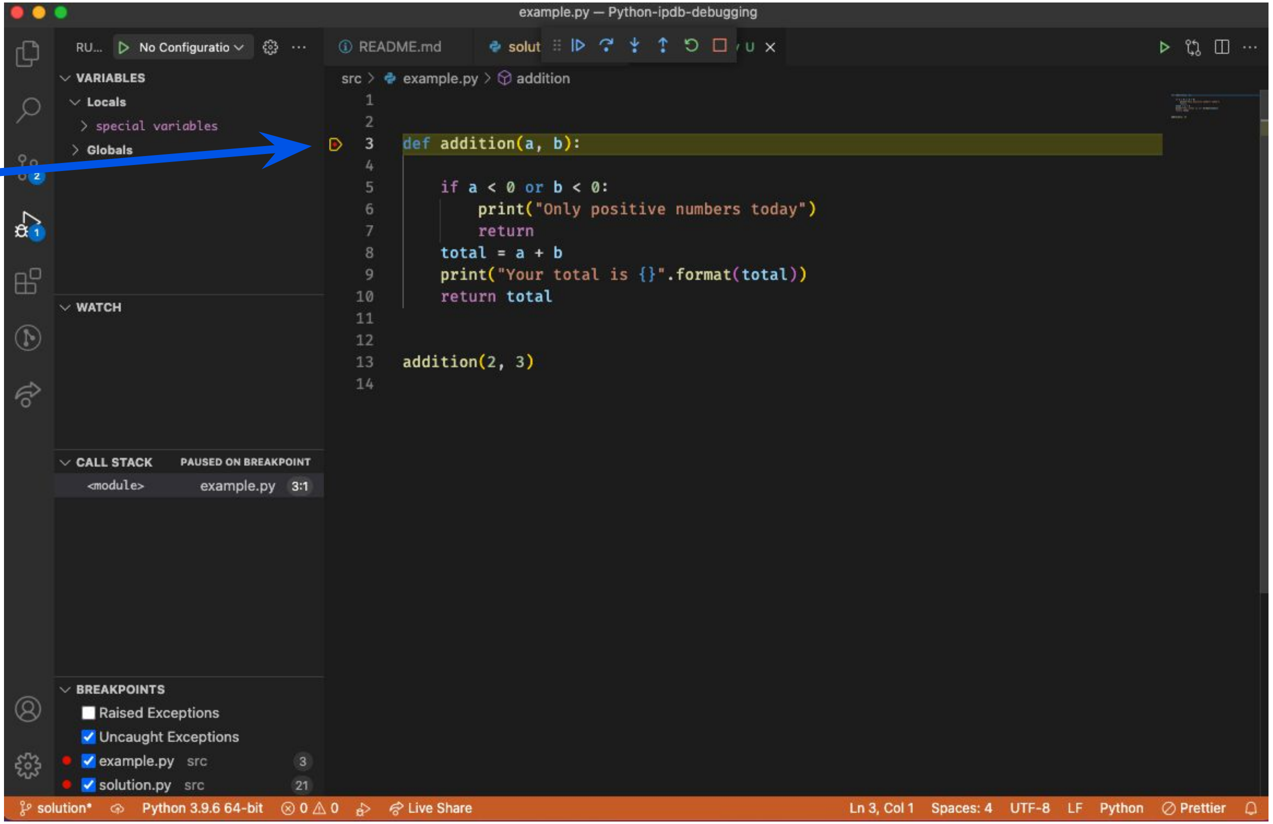
- Click the icon with a little "bug" and play button next to it (on your toolbar).
- Then click "Run and Debug".
- Choose the "Python file" option.


RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show all automatic debug configurations.
README.md
solu
src > solution.py > ...
TERMINAL
PROBLEMS

```
            pr
        elif n
            pr
        else:
            pr



# fizzbuzz(100

# To solve thi
          You, 3 h

def fizzbuzz_s
    for n in r
        if n %
            pr
        elif n
```

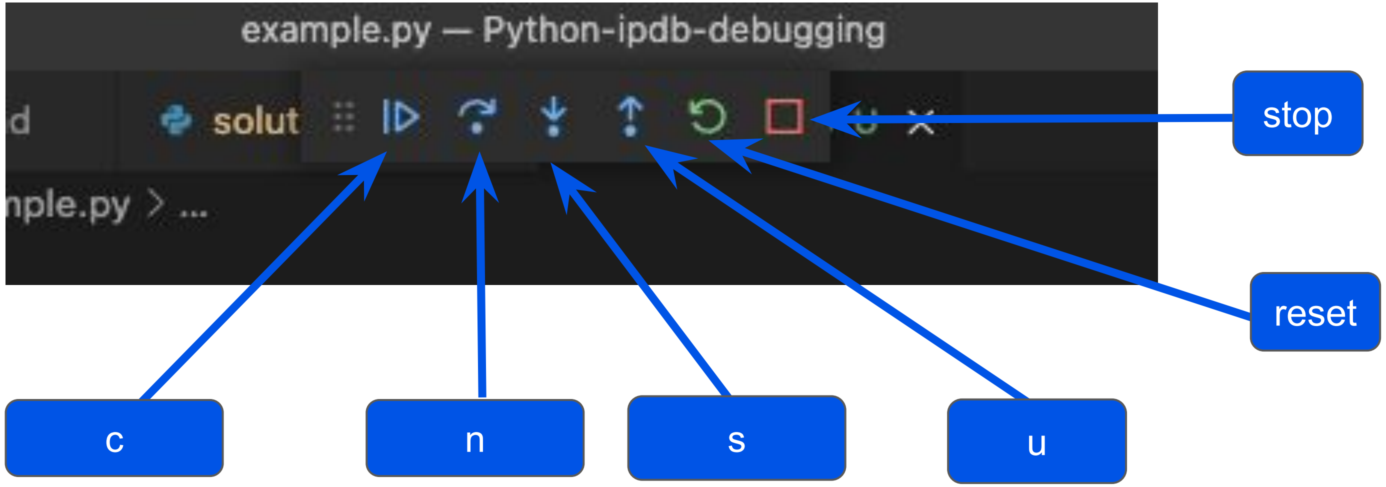
VS Code in Debug Mode

- Code stops at this point.


example.py — Python-ipdb-debugging
RU... No Configuratio
README.md
solut
VARIABLES
Locals
special variables
Globals
WATCH
CALL STACK PAUSED ON BREAKPOINT
<module> example.py 3:1
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
example.py src 3
solution.py src 21
src > example.py > addition
solution* Python 3.9.6 64-bit ⊗ 0 ⚠ 0 Live Share Ln 3, Col 1 Spaces: 4 UTF-8 LF Python Prettier

```python

def addition(a, b):

    if a < 0 or b < 0:
        print("Only positive numbers today")
        return
    total = a + b
    print("Your total is {}".format(total))
    return total


addition(2, 3)

```

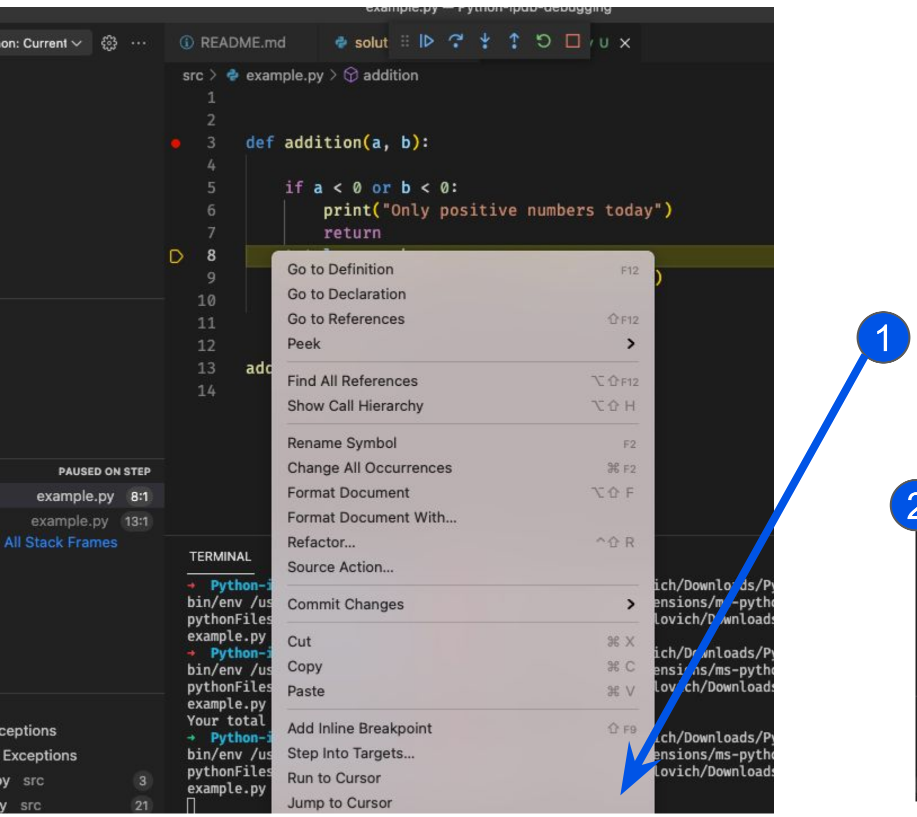
VS Code Visual Break Points



- These functions are the same we encountered while debugging with the CLI.

Advanced Debugging

Changing Variables

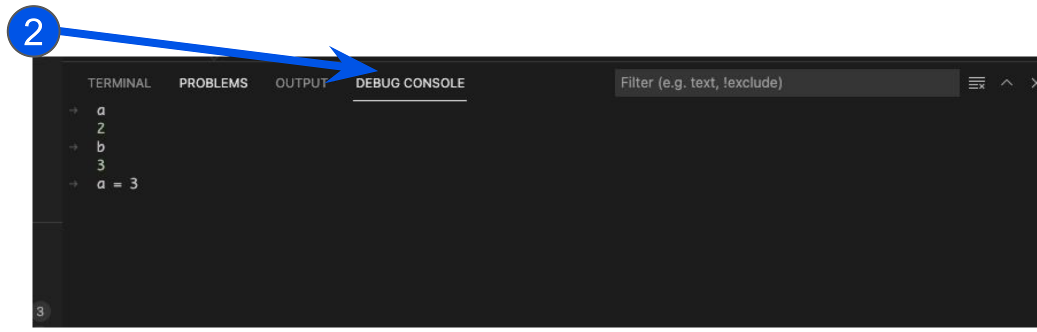


```
def addition(a, b):

    if a < 0 or b < 0:
        print("Only positive numbers today")
        return
```

```
→ Python-i                ich/Downloads/Py
bin/env /us               ensions/ms-pytho
pythonFiles               lovich/Downloads
example.py
→ Python-i                ich/Downloads/Py
bin/env /us               ensions/ms-pytho
pythonFiles               lovich/Downloads
example.py
Your total
→ Python-i                ich/Downloads/Py
bin/env /us               ensions/ms-pytho
pythonFiles               lovich/Downloads
example.py
```

Jump to Cursor

1. Right click in the current frame of code and select **Jump to Cursor.**
2. To change variables, click on the **DEBUG CONSOLE**.



```
→ a
  2
→ b
  3
→ a = 3
```

When to use a debugger?

Digital Career Institute

IDE Debugging

1. Good for larger programs.
2. Usually contains more features than CLI-based debugger.

Command Line Debugging

1. Great when programs are small.
2. Good choice if you are always in the command line.

- When debugging, using of either the IDE or CLI or shell is a personal choice.

Pitfalls & Tips

1. Do not leave your code with breakpoints in it.

2. Use git pre-commit hooks to check if you accidentally left breakpoints in your code.
    - **`pip install pre-commit`**
    - Run the install command in your repo (project).
        - **`$ pre-commit install`**
    - Create a special YAML format, save it as. **`.pre-commit-config.yml`**
    - Copy it from this gist: https://gist.github.com/miclovich/6949c5a7eb8d7996c842c9b301732edd

Documentation

Digital Career Institute

Documentation

- https://docs.python.org/3/library/pdb.html
- https://code.visualstudio.com/docs/python/debugging
- https://realpython.com/lessons/print-and-breakpoint/
- https://www.python-engineer.com/posts/python-debugger-and-breakpoint/

THANK YOU

Contact Details
DCI Digital Career Institute gGmbH

Digital Career Institute
DCI