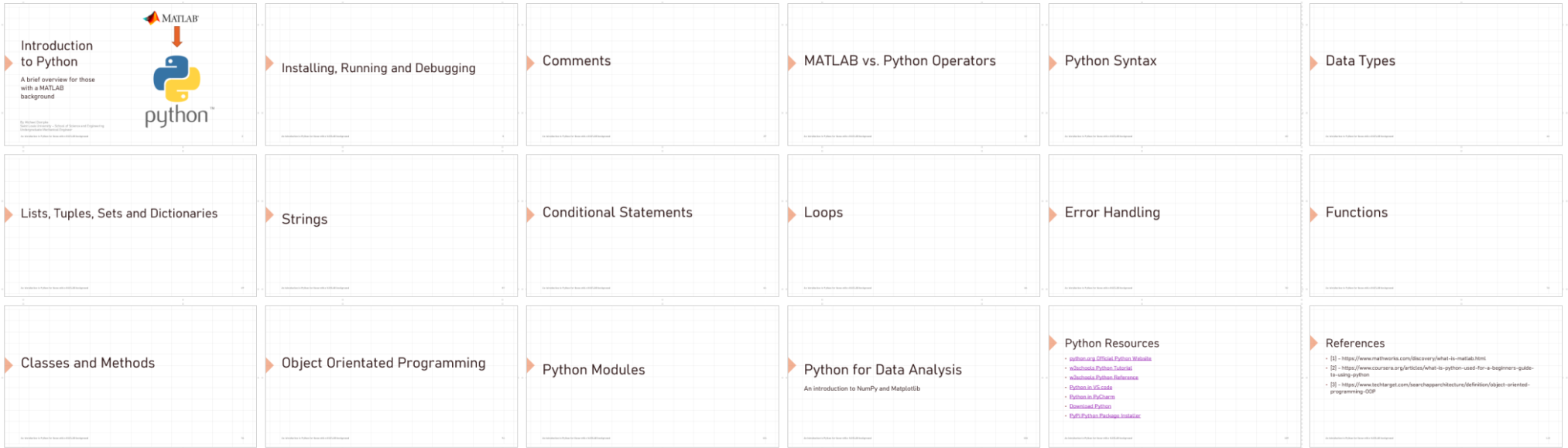
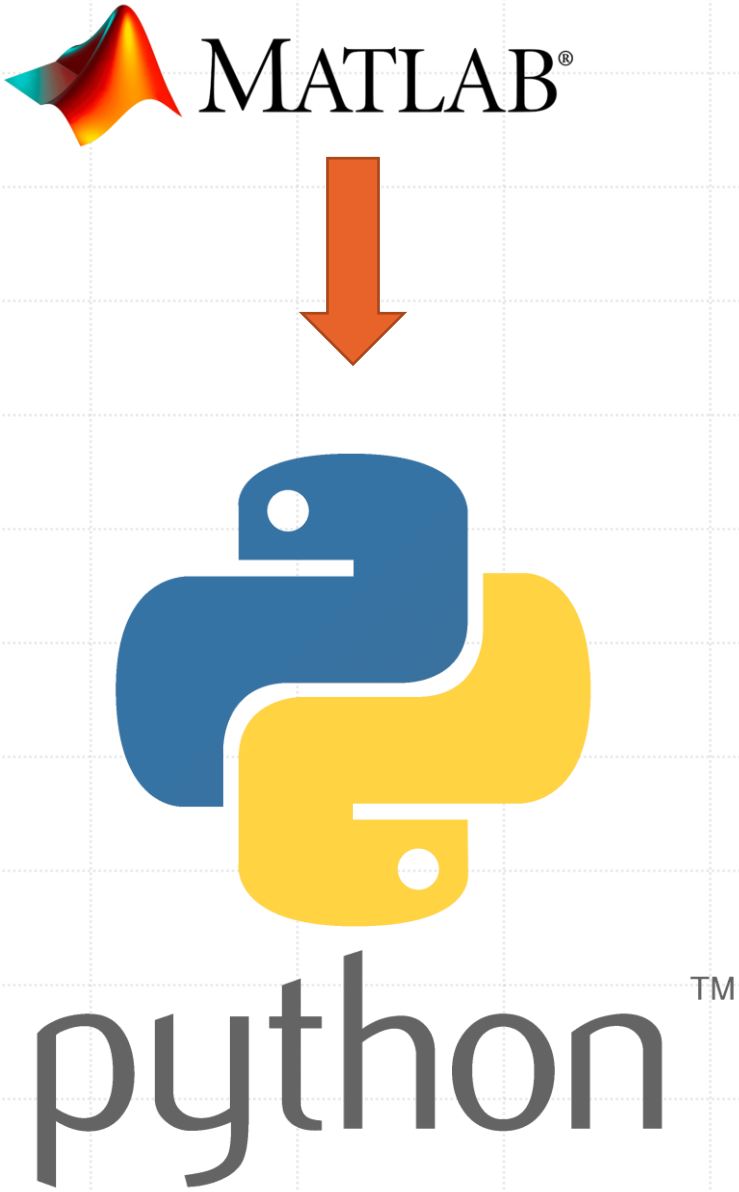
Overview


Introduction to Python
A brief overview for those with a MATLAB background
MATLAB
python
Installing, Running and Debugging
Comments
MATLAB vs. Python Operators
Python Syntax
Data Types
Lists, Tuples, Sets and Dictionaries
Strings
Conditional Statements
Loops
Error Handling
Functions
Classes and Methods
Object Orientated Programming
Python Modules
Python for Data Analysis
An introduction to NumPy and Matplotlib
Python Resources
python.org Official Python Website
w3schools Python Tutorial
w3schools Python Reference
Python in VS code
Python in PyCharm
Download Python
PyPI Python Package Installer
References
[1] – https://www.mathworks.com/discovery/what-is-matlab.html
[2] – https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python
[3] – https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP

Introduction to Python

A brief overview for those with a MATLAB background



By Michael Dompke
Saint Louis University – School of Science and Engineering
Undergraduate Mechanical Engineer

Introduction

- This Presentation is designed to act as a reference for those who are transitioning from MATLAB to Python. A well as to provide the building blocks to start developing code for ARES (Artificial Reasoning for Exploration and Space) and general use.

What's the Difference?

MATLAB

- Analyze data
- Develop algorithms
- Create models and applications
- MATLAB® is a programming platform designed specifically for engineers and scientists to analyze and design systems and products that transform our world. The heart of MATLAB is the MATLAB language, a matrix-based language allowing the most natural expression of computational mathematics.[1]
- Proprietary and Expensive Programming Software
- More $$ for use of Toolbox's

Python

- Python, one of the most popular programming languages in the world, has created everything from Netflix's recommendation algorithm to the software that controls self-driving cars. Python is a general-purpose language, which means it's designed to be used in a range of applications, including **data science**, **software and web development**, **automation**, and generally getting stuff done.[2]
- Free Programing Language
- Free Addons (Libraries)

Installing, Running and Debugging

MATLAB vs Python Interface

MATLAB

- All in one
  - Programing Language
  - Editor
  - IDE (Integrated development environment)
  - Debugger
- Must be ran in MATLAB program

Python

- Separate
  - Programing Language
  - IDE
- Can run in
  - Terminal
  - Text Editor(If Supported)
    - ATOM
    - Sublime Text
  - IDE(If Python is Supported and Configured)
    - PyCharm
    - Visual Studio Code
    - Visual Studio

Installing python

- Go to
  - https://www.python.org/downloads/
  - Download the version of python that you are looking for
    - As of 6/15/2022 ARES uses python 3.9

https://www.python.org/downloads/

python™

Donate | Search | GO | Socialize

About | Downloads | Documentation | Community | Success Stories | News | Events

Download the latest version for Windows

Download Python 3.10.5

Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python? Prereleases, Docker images

Looking for Python 2.7? See below for specific releases

Active Python Releases

For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.10	bugfix	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537
2.7	end-of-life	2010-07-03	2020-01-01	PEP 373


PyCharm
Python IDE for Professional Developers
JET BRAINS

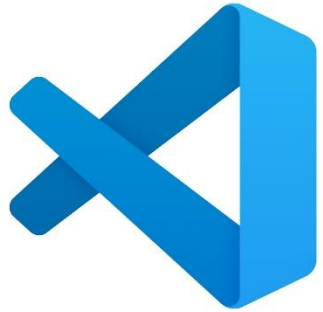
PyCharm

- PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.
- Getting Started: https://www.jetbrains.com/help/pycharm/quick-start-guide.html
- Recommended Edition
  - Community or EDU
- Available on
  - Windows
  - MacOS
  - Linux

Helpful Links PyCharm

- Quick Start: https://www.jetbrains.com/help/pycharm/quick-start-guide.html
- Debugging: https://www.jetbrains.com/help/pycharm/debugging-your-first-python-application.html

Visual Studio Code

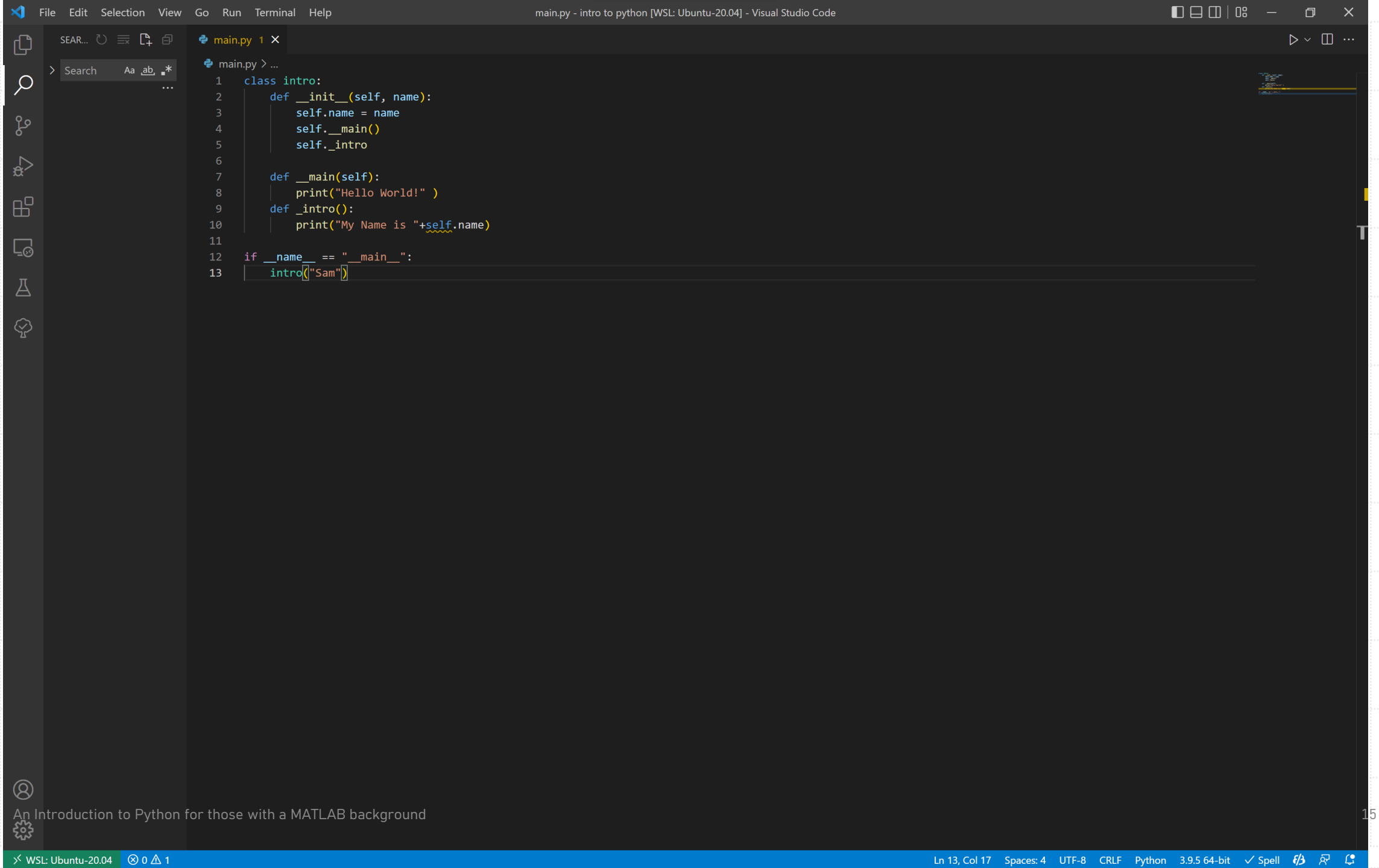


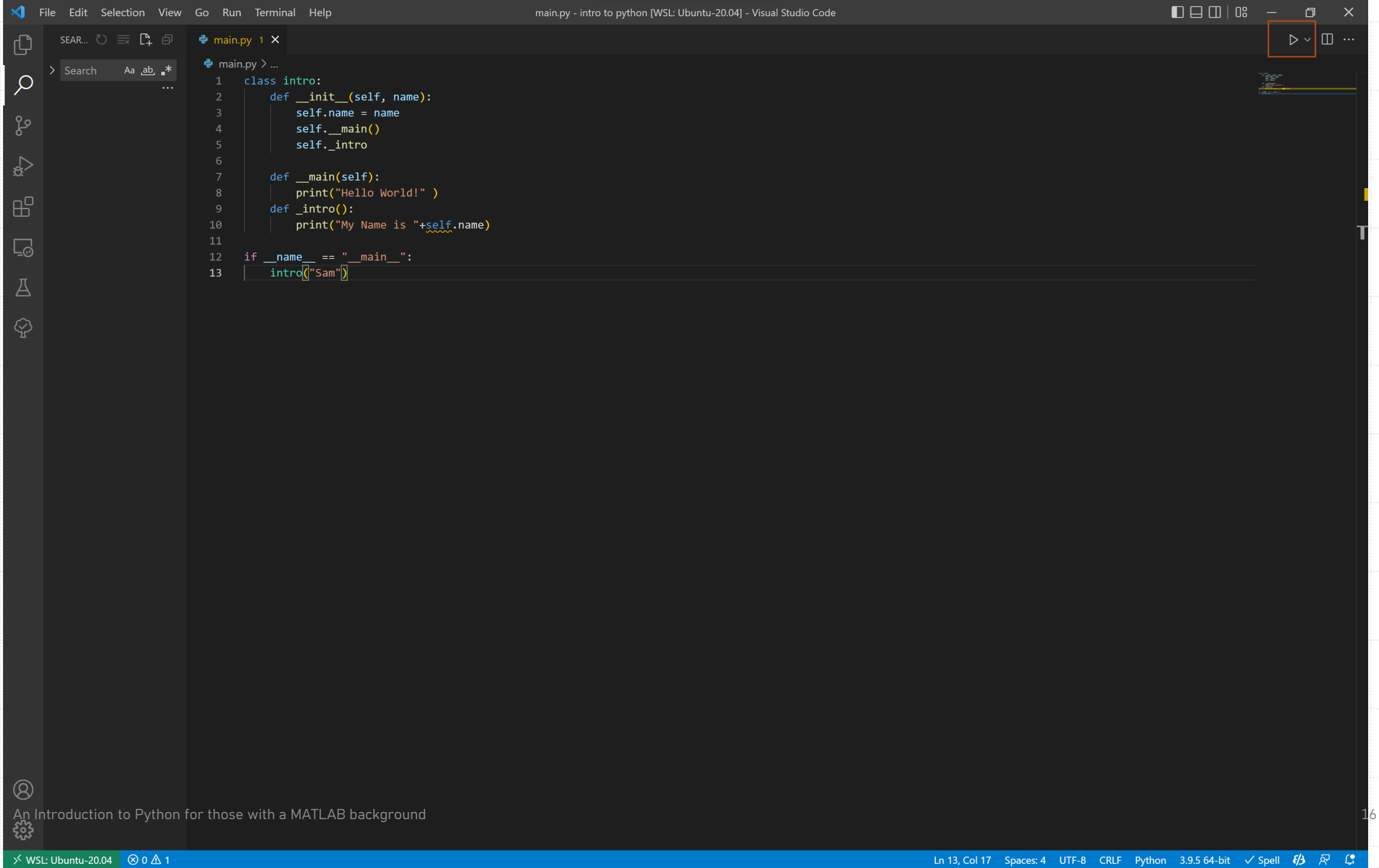
- Working with Python in Visual Studio Code, using the Microsoft Python extension, is simple, fun, and productive. The extension makes VS Code an excellent Python editor, and works on any operating system with a variety of Python interpreters. It leverages all of VS Code's power to provide auto complete and IntelliSense, linting, debugging, and unit testing, along with the ability to easily switch between Python environments, including virtual and conda environments.
- Available on:
  - Windows
  - MacOS
  - Linux

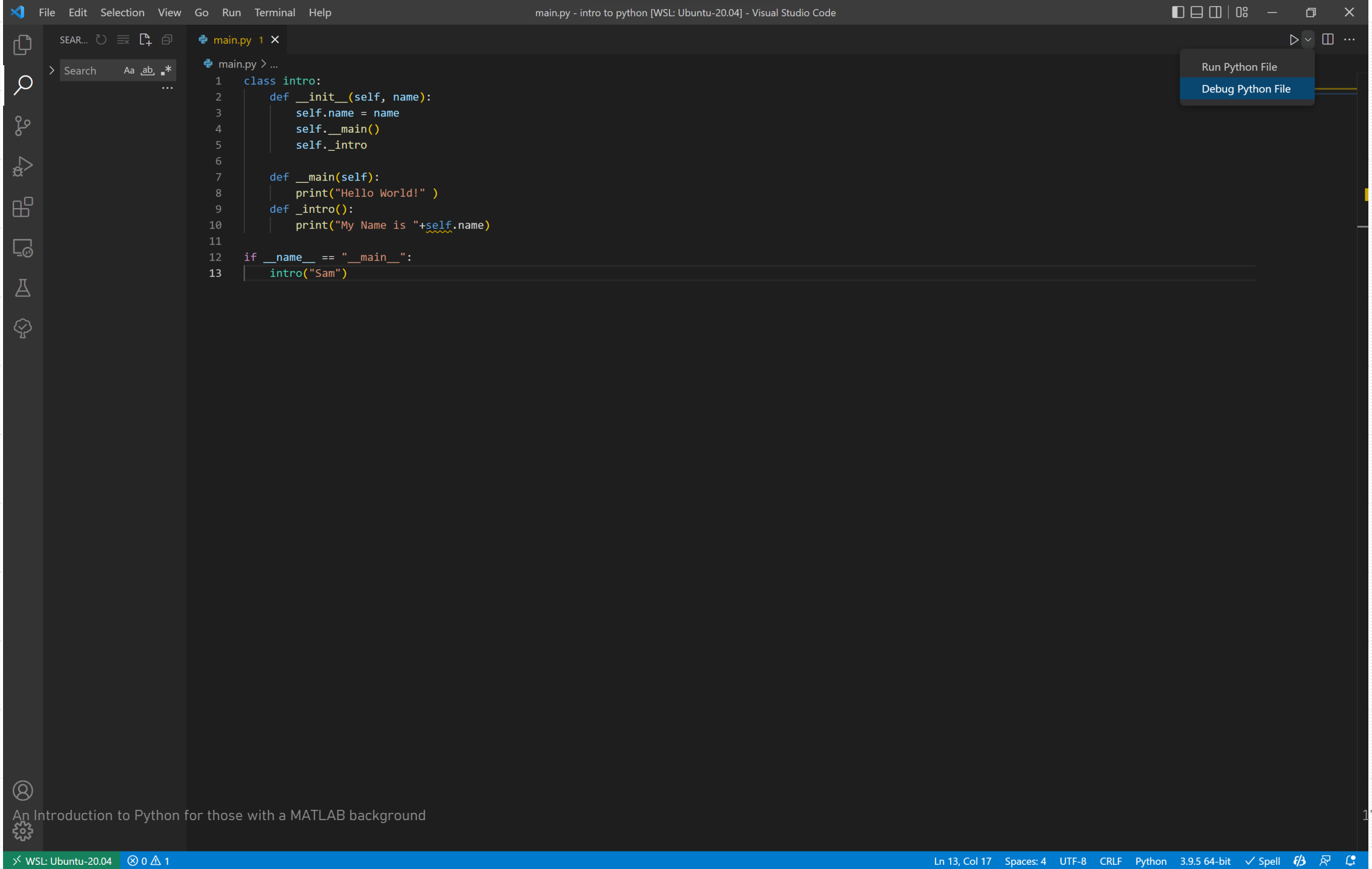
Helpful Link VS Code

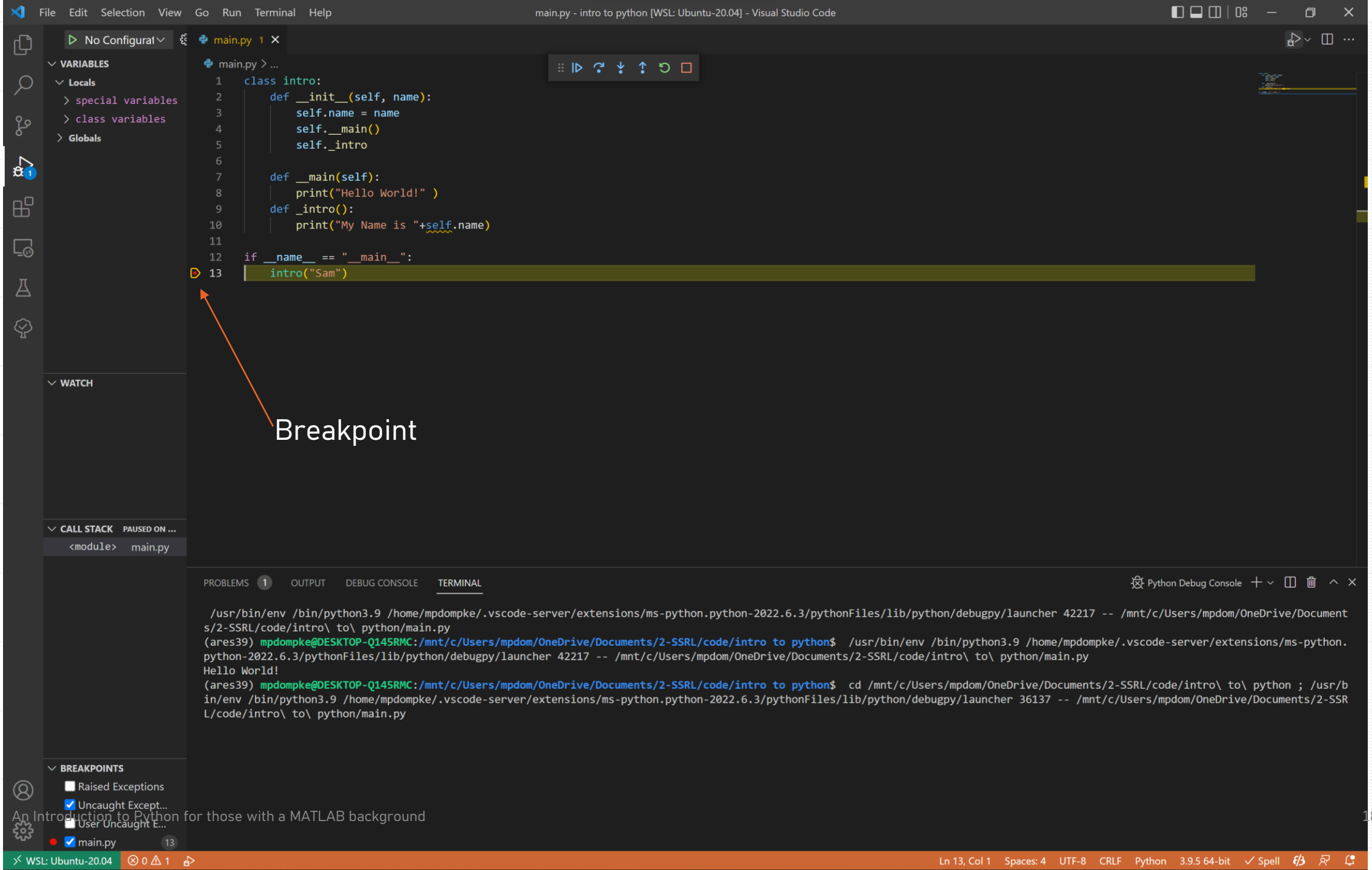
- Basic Editing in VS Code: https://code.visualstudio.com/docs/editor/codebasics
- Get started with VS Code and Python: https://code.visualstudio.com/docs/python/python-tutorial
- Managing Extensions: https://code.visualstudio.com/docs/editor/extension-marketplace
- Python Debugging VS Code: https://code.visualstudio.com/docs/python/debugging
- Working with VS Code and WSL: https://code.visualstudio.com/docs/remote/wsl-tutorial
- Remote Debugging using SSH (useful for debugging Raspberry Pi's): https://code.visualstudio.com/docs/remote/ssh

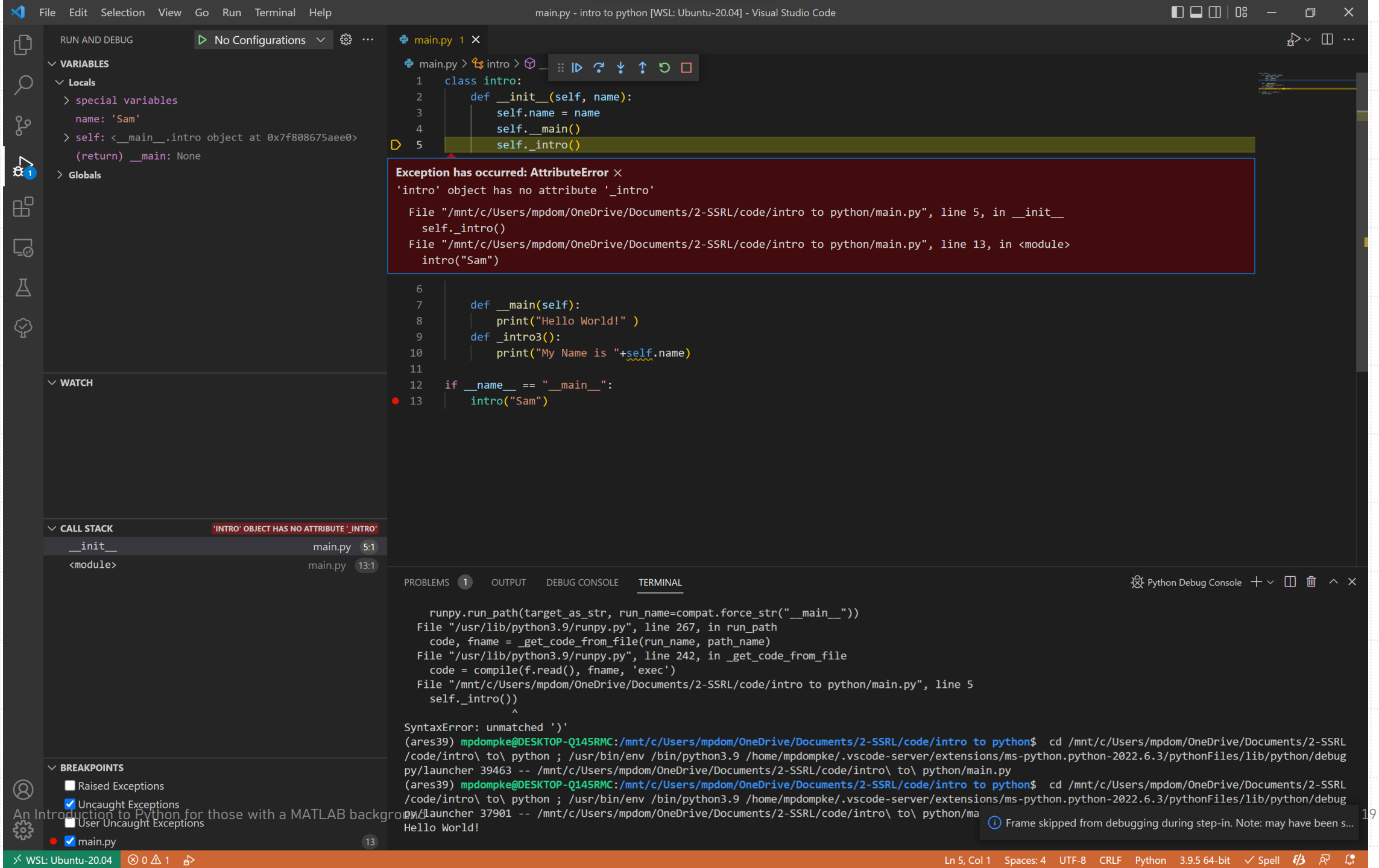
Debugging in VS code

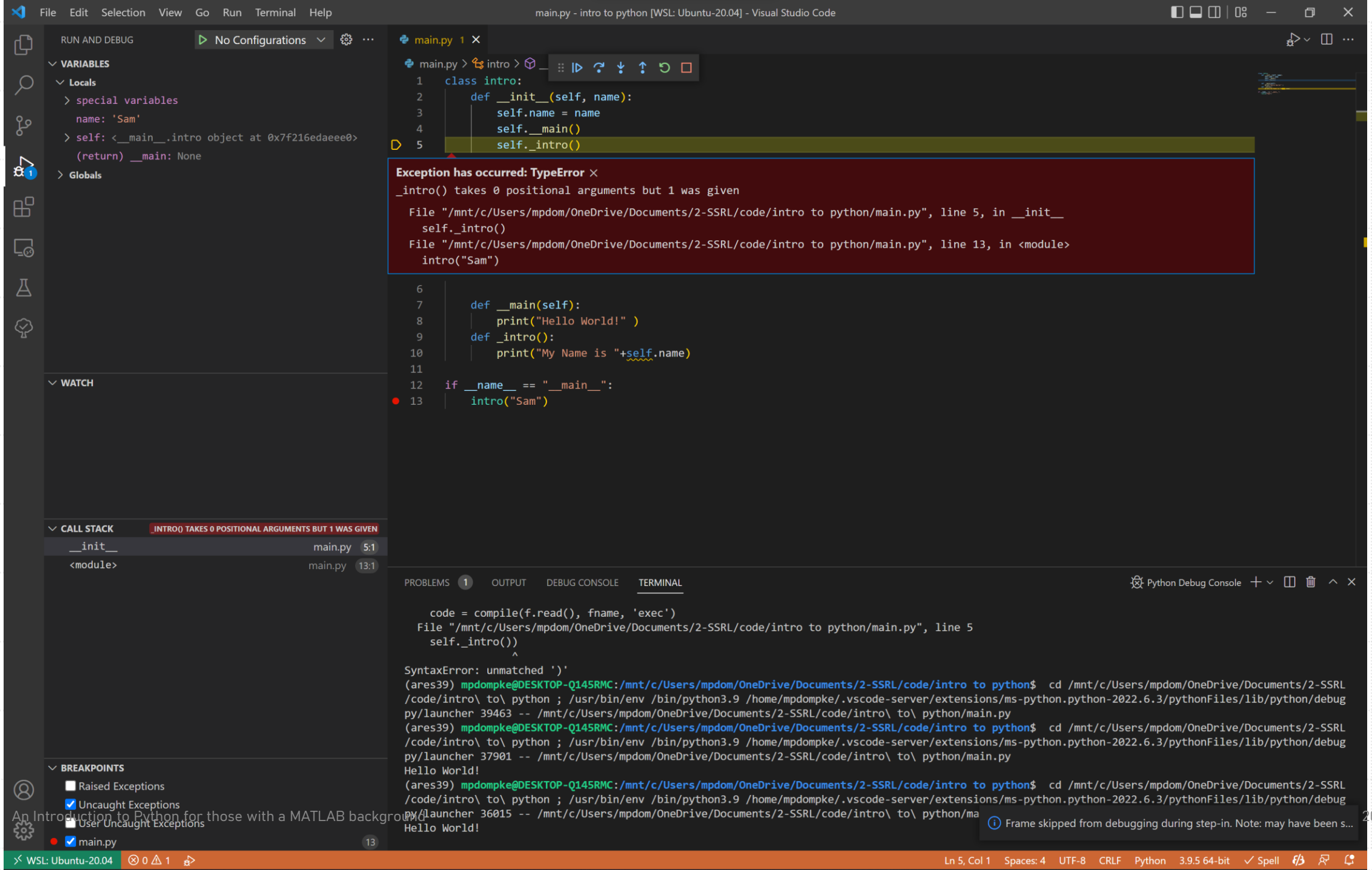

File Edit Selection View Go Run Terminal Help
main.py - intro to python [WSL: Ubuntu-20.04] - Visual Studio Code
SEAR...
Search
main.py 1
main.py > ...
1 class intro:
2 def __init__(self, name):
3 self.name = name
4 self.__main()
5 self._intro
6
7 def __main(self):
8 print("Hello World!")
9 def _intro():
10 print("My Name is "+self.name)
11
12 if __name__ == "__main__":
13 intro("Sam")
WSL: Ubuntu-20.04
0 1
Ln 13, Col 17
Spaces: 4
UTF-8
CRLF
Python
3.9.5 64-bit
Spell

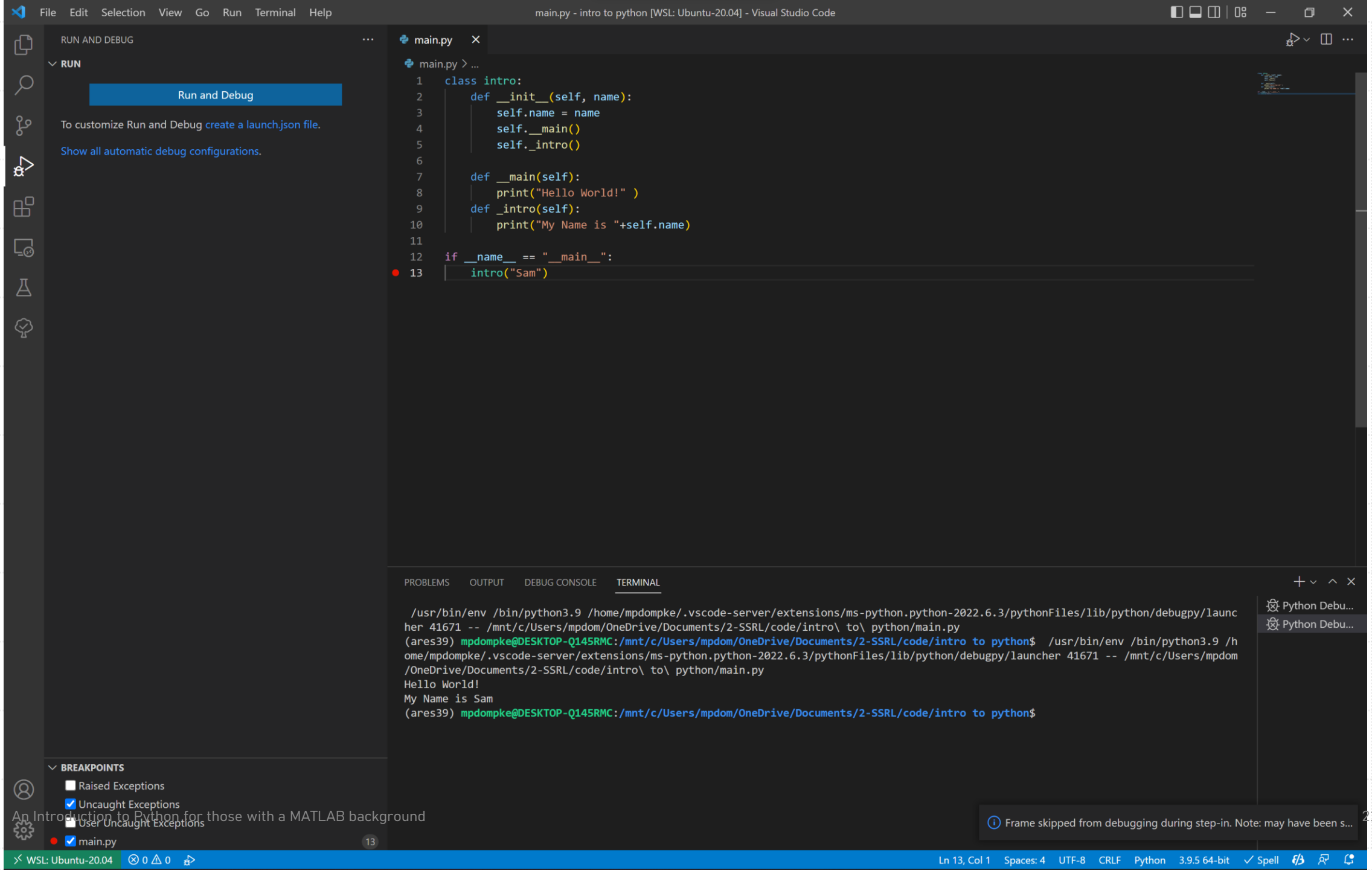

File Edit Selection View Go Run Terminal Help
main.py - intro to python [WSL: Ubuntu-20.04] - Visual Studio Code
SEAR...
Search
main.py 1
main.py > ...
1 class intro:
2 def __init__(self, name):
3 self.name = name
4 self.__main()
5 self._intro
6
7 def __main(self):
8 print("Hello World!")
9 def _intro():
10 print("My Name is "+self.name)
11
12 if __name__ == "__main__":
13 intro("Sam")
WSL: Ubuntu-20.04
0 1
Ln 13, Col 17
Spaces: 4
UTF-8
CRLF
Python
3.9.5 64-bit
Spell


File Edit Selection View Go Run Terminal Help
main.py - intro to python [WSL: Ubuntu-20.04] - Visual Studio Code
SEAR...
Search
main.py 1
main.py > ...
Run Python File
Debug Python File
1 class intro:
2 def __init__(self, name):
3 self.name = name
4 self.__main()
5 self._intro
6
7 def __main(self):
8 print("Hello World!")
9 def _intro():
10 print("My Name is "+self.name)
11
12 if __name__ == "__main__":
13 intro("Sam")
WSL: Ubuntu-20.04
0 1
Ln 13, Col 17 Spaces: 4 UTF-8 CRLF Python 3.9.5 64-bit Spell


File Edit Selection View Go Run Terminal Help
main.py - intro to python [WSL: Ubuntu-20.04] - Visual Studio Code
No Configurat
main.py 1
VARIABLES
Locals
special variables
class variables
Globals
WATCH
CALL STACK PAUSED ON ...
<module> main.py
BREAKPOINTS
Raised Exceptions
Uncaught Except...
User Uncaught E...
main.py 13
main.py > ...
1 class intro:
2 def __init__(self, name):
3 self.name = name
4 self.__main()
5 self._intro
6
7 def __main(self):
8 print("Hello World!")
9 def _intro():
10 print("My Name is "+self.name)
11
12 if __name__ == "__main__":
13 intro("Sam")
Breakpoint
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL
Python Debug Console
/usr/bin/env /bin/python3.9 /home/mpdompke/.vscode-server/extensions/ms-python.python-2022.6.3/pythonFiles/lib/python/debugpy/launcher 42217 -- /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro\ to\ python/main.py
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$ /usr/bin/env /bin/python3.9 /home/mpdompke/.vscode-server/extensions/ms-python.python-2022.6.3/pythonFiles/lib/python/debugpy/launcher 42217 -- /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro\ to\ python/main.py
Hello World!
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$ cd /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro\ to\ python ; /usr/bin/env /bin/python3.9 /home/mpdompke/.vscode-server/extensions/ms-python.python-2022.6.3/pythonFiles/lib/python/debugpy/launcher 36137 -- /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro\ to\ python/main.py
WSL: Ubuntu-20.04 0 1
Ln 13, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.5 64-bit Spell

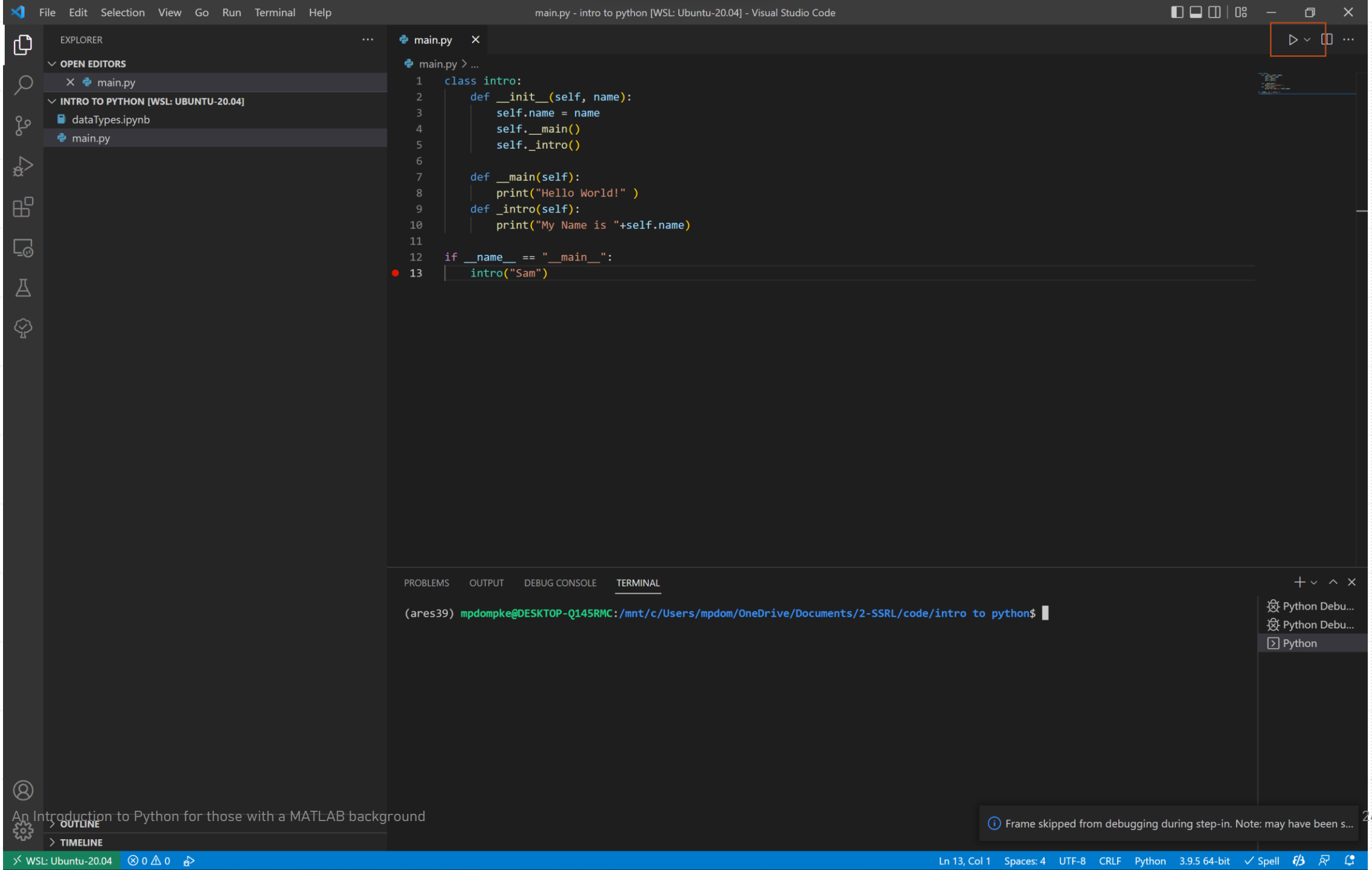

File Edit Selection View Go Run Terminal Help
main.py - intro to python [WSL: Ubuntu-20.04] - Visual Studio Code
RUN AND DEBUG
No Configurations
VARIABLES
Locals
special variables
name: 'Sam'
self: <__main__.intro object at 0x7f808675aee0>
(return) __main: None
Globals
WATCH
CALL STACK
'INTRO' OBJECT HAS NO ATTRIBUTE '_INTRO'
__init__ main.py 5:1
<module> main.py 13:1
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
main.py 13
main.py 1
main.py > intro > __
1 class intro:
2 def __init__(self, name):
3 self.name = name
4 self.__main()
5 self._intro()
Exception has occurred: AttributeError
'intro' object has no attribute '_intro'
File "/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python/main.py", line 5, in __init__
self._intro()
File "/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python/main.py", line 13, in <module>
intro("Sam")
6
7 def __main(self):
8 print("Hello World!")
9 def _intro3():
10 print("My Name is "+self.name)
11
12 if __name__ == "__main__":
13 intro("Sam")
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
Python Debug Console
runpy.run_path(target_as_str, run_name=compat.force_str("__main__"))
File "/usr/lib/python3.9/runpy.py", line 267, in run_path
code, fname = _get_code_from_file(run_name, path_name)
File "/usr/lib/python3.9/runpy.py", line 242, in _get_code_from_file
code = compile(f.read(), fname, 'exec')
File "/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python/main.py", line 5
self._intro())
^
SyntaxError: unmatched ')'
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$ cd /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL
/code/intro\ to\ python ; /usr/bin/env /bin/python3.9 /home/mpdompke/.vscode-server/extensions/ms-python.python-2022.6.3/pythonFiles/lib/python/debug
py/launcher 39463 -- /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro\ to\ python/main.py
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$ cd /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL
/code/intro\ to\ python ; /usr/bin/env /bin/python3.9 /home/mpdompke/.vscode-server/extensions/ms-python.python-2022.6.3/pythonFiles/lib/python/debug
py/launcher 37901 -- /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro\ to\ python/ma
Hello World!
Frame skipped from debugging during step-in. Note: may have been s...
WSL: Ubuntu-20.04
Ln 5, Col 1
Spaces: 4
UTF-8
CRLF
Python
3.9.5 64-bit
Spell

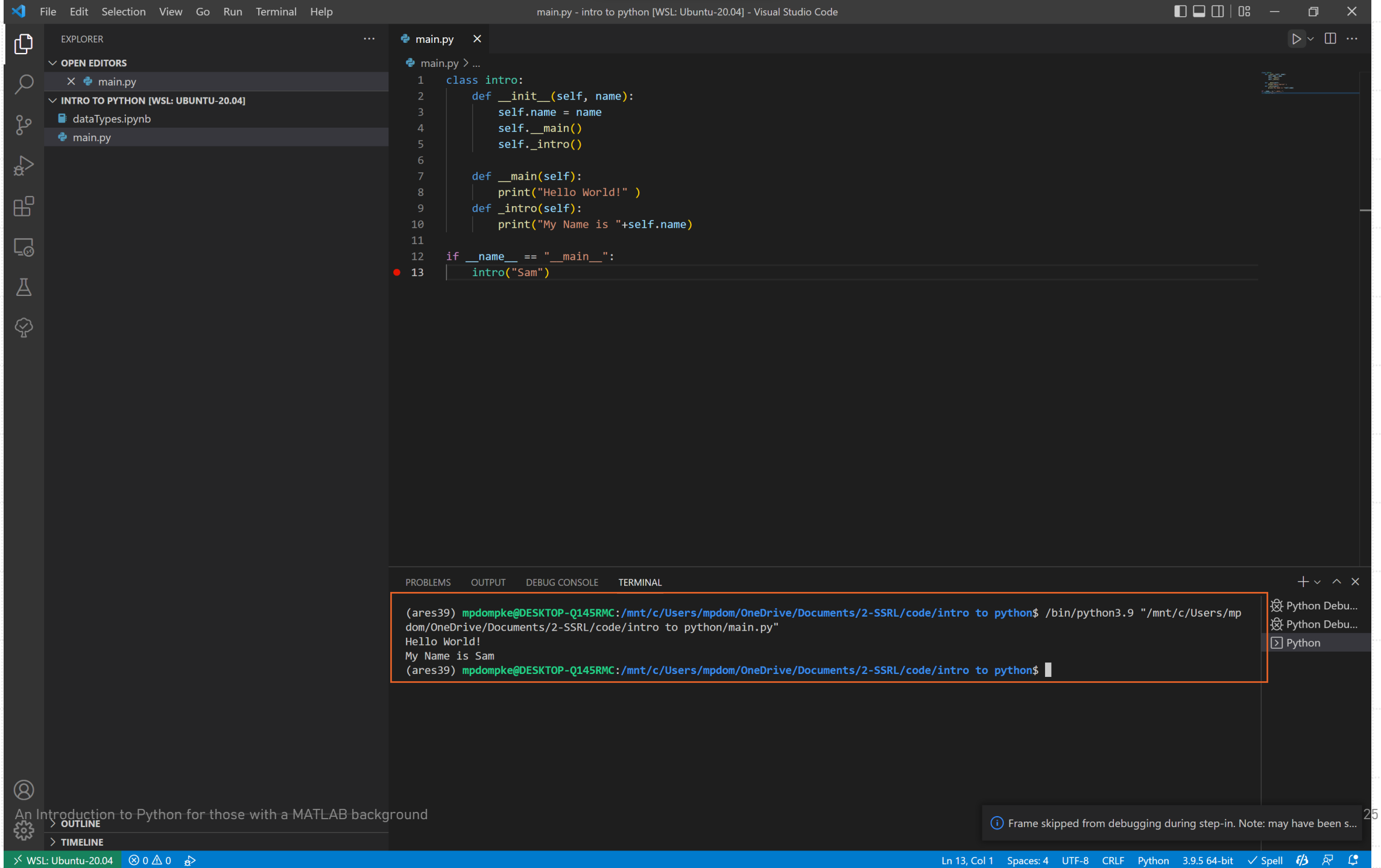

File Edit Selection View Go Run Terminal Help
main.py - intro to python [WSL: Ubuntu-20.04] - Visual Studio Code
RUN AND DEBUG
No Configurations
VARIABLES
Locals
special variables
name: 'Sam'
self: <__main__.intro object at 0x7f216edaeee0>
(return) __main: None
Globals
WATCH
CALL STACK
_INTRO() TAKES 0 POSITIONAL ARGUMENTS BUT 1 WAS GIVEN
__init__ main.py 5:1
<module> main.py 13:1
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
main.py 13
main.py 1
main.py > intro > __
1 class intro:
2 def __init__(self, name):
3 self.name = name
4 self.__main()
5 self._intro()
Exception has occurred: TypeError
_intro() takes 0 positional arguments but 1 was given
File "/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python/main.py", line 5, in __init__
self._intro()
File "/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python/main.py", line 13, in <module>
intro("Sam")
6
7 def __main(self):
8 print("Hello World!")
9 def _intro():
10 print("My Name is "+self.name)
11
12 if __name__ == "__main__":
13 intro("Sam")
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
Python Debug Console
code = compile(f.read(), fname, 'exec')
File "/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python/main.py", line 5
self._intro())
^
SyntaxError: unmatched ')'
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$ cd /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro\ to\ python ; /usr/bin/env /bin/python3.9 /home/mpdompke/.vscode-server/extensions/ms-python.python-2022.6.3/pythonFiles/lib/python/debugpy/launcher 39463 -- /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro\ to\ python/main.py
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$ cd /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro\ to\ python ; /usr/bin/env /bin/python3.9 /home/mpdompke/.vscode-server/extensions/ms-python.python-2022.6.3/pythonFiles/lib/python/debugpy/launcher 37901 -- /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro\ to\ python/main.py
Hello World!
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$ cd /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro\ to\ python ; /usr/bin/env /bin/python3.9 /home/mpdompke/.vscode-server/extensions/ms-python.python-2022.6.3/pythonFiles/lib/python/debugpy/launcher 36015 -- /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro\ to\ python/ma
Hello World!
Frame skipped from debugging during step-in. Note: may have been s...
WSL: Ubuntu-20.04
Ln 5, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.5 64-bit Spell


File Edit Selection View Go Run Terminal Help
main.py - intro to python [WSL: Ubuntu-20.04] - Visual Studio Code
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show all automatic debug configurations.
main.py
main.py > ...
1 class intro:
2 def __init__(self, name):
3 self.name = name
4 self.__main()
5 self._intro()
6
7 def __main(self):
8 print("Hello World!")
9 def _intro(self):
10 print("My Name is "+self.name)
11
12 if __name__ == "__main__":
13 intro("Sam")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
/usr/bin/env /bin/python3.9 /home/mpdompke/.vscode-server/extensions/ms-python.python-2022.6.3/pythonFiles/lib/python/debugpy/launcher 41671 -- /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro\ to\ python/main.py
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$ /usr/bin/env /bin/python3.9 /home/mpdompke/.vscode-server/extensions/ms-python.python-2022.6.3/pythonFiles/lib/python/debugpy/launcher 41671 -- /mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro\ to\ python/main.py
Hello World!
My Name is Sam
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$
Python Debu...
Python Debu...
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
main.py 13
Frame skipped from debugging during step-in. Note: may have been s...
WSL: Ubuntu-20.04
Ln 13, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.5 64-bit Spell

Now that it is debugged we can run it

Running from VS code


File Edit Selection View Go Run Terminal Help
main.py - intro to python [WSL: Ubuntu-20.04] - Visual Studio Code
EXPLORER
OPEN EDITORS
main.py
INTRO TO PYTHON [WSL: UBUNTU-20.04]
dataTypes.ipynb
main.py
main.py
main.py > ...
1 class intro:
2 def __init__(self, name):
3 self.name = name
4 self.__main()
5 self._intro()
6
7 def __main(self):
8 print("Hello World!")
9 def _intro(self):
10 print("My Name is "+self.name)
11
12 if __name__ == "__main__":
13 intro("Sam")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$
Python Debu...
Python Debu...
Python
OUTLINE
TIMELINE
Frame skipped from debugging during step-in. Note: may have been s...
WSL: Ubuntu-20.04
Ln 13, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.5 64-bit Spell


File Edit Selection View Go Run Terminal Help
main.py - intro to python [WSL: Ubuntu-20.04] - Visual Studio Code
EXPLORER
OPEN EDITORS
main.py
INTRO TO PYTHON [WSL: UBUNTU-20.04]
dataTypes.ipynb
main.py
main.py
main.py > ...
1 class intro:
2 def __init__(self, name):
3 self.name = name
4 self.__main()
5 self._intro()
6
7 def __main(self):
8 print("Hello World!")
9 def _intro(self):
10 print("My Name is "+self.name)
11
12 if __name__ == "__main__":
13 intro("Sam")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$ /bin/python3.9 "/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python/main.py"
Hello World!
My Name is Sam
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$
Python Debu...
Python Debu...
Python
OUTLINE
TIMELINE
Frame skipped from debugging during step-in. Note: may have been s...
WSL: Ubuntu-20.04
Ln 13, Col 1
Spaces: 4
UTF-8
CRLF
Python
3.9.5 64-bit
Spell

Running from Terminal*

```
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$ ls
dataTypes.ipynb  main.py
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$ python main.py
Hello World!
My Name is Sam
```


* Ran on Linux however should be the same for Windows and MacOS

Editing in terminal (for Linux)

```
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$ ls
dataTypes.ipynb  main.py
(ares39) mpdompke@DESKTOP-Q145RMC:/mnt/c/Users/mpdom/OneDrive/Documents/2-SSRL/code/intro to python$ nano main.py
```

```
  GNU nano 4.8                                  main.py
class intro:
    def __init__(self, name):
        self.name = name
        self.__main()
        self._intro()

    def __main(self):
        print("Hello World!" )
    def _intro(self):
        print("My Name is "+self.name)

if __name__ == "__main__":
    intro("Sam")
```



```
[ Read 13 lines (Converted from DOS format) ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  M-U Undo  M-A Mark Text  M-] To Bracket  M-Q Previous
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo  M-6 Copy Text  ^Q Where Was  M-W Next
```

Comments

Comments

	MATLAB	Python
Single Line	% This is my comment	# This is my comment
Multi Line	%{ This is my comment Written in More than just one line }%	""" This is my comment Written in More than just one line """

Comments cont.

MATLAB

```
% This is my comment
disp(“Hello, World!)
%{
This is my
multi line comment
}%
```

PYTHON

```
# This is my comment
print(“Hello, World!)
“””
This is my
multi line comment
“””
```

MATLAB vs. Python Operators

MATLAB vs Python Arithmetic Operators

MATLAB Operator	Python Operator	Name	Symbolic Example	Numerical Example
+	+	Addition	x + y	8 + 3 = 11
-	-	Subtraction	x - y	8 - 3 = 7
*	*	Multiplication	x * y	8 * 3 = 24
/	/	Division	x / y	8 / 3 = 2.667
mod(a,m)	%	Modulus	x % y	8 % 3 = 2
^	**	Exponentiation	x ** y	8 * 3 = 512
floor(a/b)	//	Floor Division	x // y	8 // 3 = 2

MATLAB vs Python Assignment Operators

MATLAB Operator	Python Operator	Example	Same As	Numerical Example (x = 5)
=	=	x = 5	x= 5	x = 5
N/A	+=	x += 3	x= x + 3	x += 3 -> 8
N/A	-=	x -= 3	x= x - 3	x -= 3 -> 2
N/A	*=	x *= 3	x= x * 3	x *= 3 -> 15
N/A	/=	x /= 3	x= x / 3	x /= 3 -> 1.667
N/A	%=	x %= 3	x= x % 3	x %= 3 -> 2
N/A	//=	x //= 3	x= x // 3	x //= 3 -> 1
N/A	**=	x **= 3	x= x ** 3	x **= 3 -> 125

MATLAB vs Python Comparison Operators

MATLAB Operator	Python Operator	Name	Symbolic Example	Example 1
==	==	Equal	x == y	5 == 3 -> False
~=	!=	Not Equal	x != y	5 != 3 -> True
>	>	Greater Than	x > y	5 > 3 -> True
<	<	Less than	x < y	5 < 3 -> False
>=	>=	Greater than or equal to	x >= y	5 >= 5 -> True
<=	<=	Less than or equal to	x <= y	5 <= 3 -> False

MATLAB vs Python Logical Operators

MATLAB Operator	Python Operator	Description	Example (x = 23)
&	and	Returns True if both statements are True	x< 5 and x > 10 -> False
|	or	Returns True if one of the statements are true	x> 5 or x < 10 -> True
~	not	Reverse the result, returns False if the result is True	Not(x> 5 or x < 10) -> False

MATLAB vs Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location.

MATLAB Operator	Python Operator	Description	Example
N/A	is	Returns True if both variables are the same object	x is y
N/A	is not	Returns True if both variables are not the same object	x is not y

```
x = ["apple", "banana"]
y = ["apple", "banana"]
z = x

print(x is z)

# returns True because z is the same object as x

print(x is y)

# returns False because x is not the same object as y, even if they have the
same content

print(x == y)

# to demonstrate the difference between "is" and "==": this comparison
returns True because x is equal to y
```

```
x = ["apple", "banana"]
y = ["apple", "banana"]
z = x

print(x is not z)

# returns False because z is the same object as x

print(x is not y)

# returns True because x is not the same object as y, even if they have the
same content

print(x != y)

# to demonstrate the difference between "is not" and "!=": this comparison
returns False because x is equal to y
```

MATLAB vs Python Identity Operators cont.

- Be careful when checking for Identity.
- Python treats some data types different than others

Ex:

```
>>> a = 'test'
>>> b = 'test'
>>> a == b
True
>>> a is b
True
```

MATLAB vs Python Membership Operators

Membership operators are used to test if a sequence is presented in an object

MATLAB Operator	Python Operator	Description	Example
N/A	in	Returns True if a sequence with the specified value is present in the object	x in y
N/A	not in	Returns True if a sequence with the specified value is not present in the object	x not in y

```
x = ["apple", "banana"]

print("banana" in x)

# returns True because a sequence with the value "banana" is in the list

print("pineapple" not in x)

# returns True because a sequence with the value "pineapple" is not in the list
```

Python Syntax

Executing Python Syntax

- In command line
  - Python syntax can be executed directly in the command line

```
(ares39) mpdompke@DESKTOP-Q145RMC:~$ python
Python 3.9.12 (main, Apr  5 2022, 06:56:58)
[GCC 7.5.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World!")
Hello World!
>>>
```

Executing Python File in Terminal

```
  GNU nano 4.8                                                  main.py
from datetime import datetime as dt
print('This is my Program')
print('Running: '+str(__file__))
print('Currnet Time(UTC): ' + str(dt.utcnow()))
```

```
(ares39) mpdompke@DESKTOP-Q145RMC:~$ python main.py
This is my Program
Running: /home/mpdompke/main.py
Currnet Time(UTC): 2022-06-16 03:08:00.723841
```

Python Indentation

- Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.
- Python uses indentation to indicate a block of code.
- Ex1

```
if 5 > 2:
  print("Five is greater than two!") # Five is greater than two!
```

- Ex2

```
if 5 > 2:
print("Five is greater than two!") # Error Raised: Syntax Error
```

Python Variables

- In Python, variables are created when you assign a value to it:

Ex:

```
>>> x = 5
>>> y = "hello world!"
>>> print(x)
5
>>> print(y)
hello world!
```

Python Variables cont.

- In Python variables are **CASE-SENSITIVE**

Ex:

```
>>> q = 5
>>> Q = 'Hello world'
>>> print(str(q)+'\n'+str(Q))
5
Hello world
```

Data Types

Data Types in Python

Category	Data Types
Text Type	str
Numeric Types	int, float, complex
Sequence Types	list, tuple
Mapping Types	dict
Set Types	set, frozenset
Boolean Types	bool
Binary Types	bytes, bytearray, memoryview
None Types	NoneType

Setting the Data Type

Example	Data Type
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview
x = None	NoneType

Lists, Tuples, Sets and Dictionaries

List

- List are used to store multiple items in a single variable
- Items within a list can be a changed

Ex

myList = ["apple", "banana", "cherry"]

myList[1] = “cherry”

print(myList)

OUTPUT: ["apple", "cherry“, "cherry"]

Tuple

- Similar to a list a tuple stores a collection of items
- However there are a few changes,
  - A Tuple can not be changed once declared
  - Items can not be added or removed

Ex:

```
mytuple = ("apple", "banana", "cherry")
print(mytuple[1])
```

OUTPUT: “banana”

```
mytuple[1] = “grape”
```

ERROR RAISED:

TypeError: 'tuple' object does not support item assignment

Indexing a list and tuple

	MATLAB	MATLAB Example A = [2032,2074,1234]	Python	Python Example A = [2032,2074,1234]
Starts with	1	A(1) -> 2032	0	A[0] -> 2032
Last element of a list	end	A(end) -> 1234	-1	A[-1] -> 1234
n from end of list	end – (n-1)	A(end – (2 - 1)) -> 2074 Or A(end – 1) -> 2074	-n	A[-2] -> 2074

^ wired notation but necessary to be in the same place as Python for the same n value

Sets

- Similar to a tuple and list a set is used to store multiple items
- A set is a collection which is *unordered*, *unchangeable*, and *unindexed*. However a set can have items added or removed
- Duplicates are not allowed in a set

```
thisset = {"apple", "banana", "cherry"}

thisset.add("orange")
print(thisset)
```

OUTPUT: {"apple", "banana", "cherry“, "orange"}

```
thisset.remove("banana")
print(thisset)
```

OUTPUT: {"apple", "cherry“, "orange"}

Dictionaries

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered, changeable and do not allow duplicates.

EX:

```
thisdict =    {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict['brand'])
```

OUTPUT:

Ford

Dictionaries cont.

- Adding to a dictionary

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["color"] = "red"
print(thisdict)
```

OUTPUT:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

Summary

Name	Bracket Type	Ordered	Changeable	Indexed	Duplicates	Example
List	[ ]	X	X	X	X	A = [1,2,3]
Tuple	( )	X		X	X	B = (1,2,3)
Set	{ }		* Can add or remove			C = {“a”,”b”,”c”}

Dictionary	{ }	X* with Python 3.7 and later	X			D = {
						“a”:1,
						“b”: “two”
						}

Strings

What is a string?

- In python a string is a data type, that is represented by either single or double quotation marks
- ‘hello’ is the same as “hello”
- A string can be displayed with the print() function.

Concatenate Strings

- In python the concatenating operator is +

```
>>> a = "Hello"
>>> b = "World"
```

```
>>> c = a + b
>>> print(c)
HelloWorld
```

```
>>> c = a + " " + b
>>> print(c)
Hello World
```

A " " can be added to the concatenation to add a space

Strings are Arrays

```
>>> a = "Hello, World"
>>> print(a[2:9])
llo, Wo
```

Length of a String

- The length of a string can be determined with the len() function.
- This function can also be used to determine the number of items in a list, tuple, dict, set,

```
>>> a = "Hello, World"
>>> print(len(a))
12
```

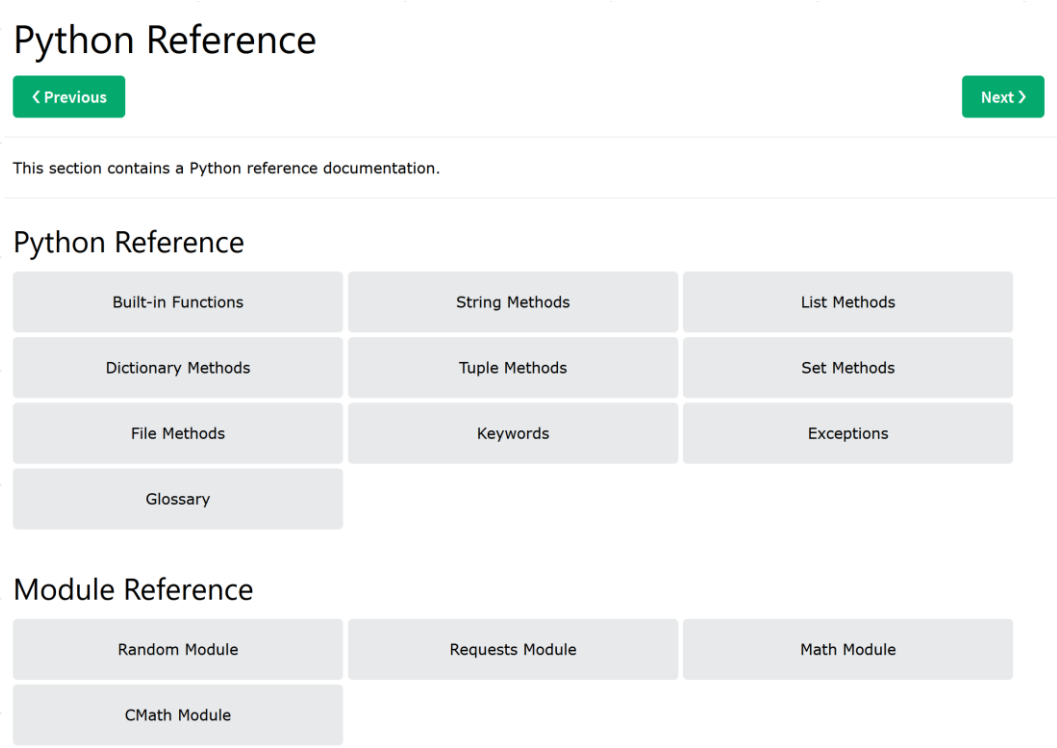
Modifying Strings

Built in string methods	Description	Example
.upper()	Returns a strings in upper case	
.lower()	Returns a string in lower case	
.replace(a,b)	Replaced a string with another string	
.split()	Returns a list where the text between a specific operator becomes the list items	

```
>>> a = "Hello, World"
>>> print(a.upper())
HELLO, WORLD
>>> print(a.lower())
hello, world
>>> print(a.replace('W','A'))
Hello, Aorld
>>> print(a.split(','))
['Hello', ' World']
```

Complete list of string methods: https://www.w3schools.com/python/python_ref_string.asp

Python Reference



w3schools Python Reference

Conditional Statements

If … Else

MATLAB

```
x = 5;
if x < 5
        disp('x is less than 5')
elseif x == 5
        disp('x is 5')
else
        disp('x is greater than 5')
End
x
```

Python

```
x = 5
if x < 5:
        print('x is less than 5')
elif x == 5:
        print('x is 5')
else:
        print('x is greater than 5')
print(x)
```

Both display the same output

Loops

For Loops

MATLAB

```
s = 7;
for i = 1:s
        i^2
end
```

Python

```
s = 7
for i in range(s):
        print((i+1)**2)
```

Both display the following

1
4
9
16
25
36
49

For Loops with list

MATLAB

```
l = [1, 3, 5, 7];
for i = l
        i^2
end
```

Python

```
l = [1, 3, 5, 7]
for i in l:
        print(i**2)
```

Both display the following
1
9
25
49

While Loops

MATLAB

```
n = 10;
f = n;
while n > 1
        n = n – 1;
        f = f*n
End
disp(['n! = ' num2str(f)])
```

Python

```
n = 10
f = n
while n > 1:
        n –= 1
        f *= n
print('n! = ' + str(f))
```

Error Handling

Error Handling

MATLAB

```
try
    a = notaFunction(5,6);
catch
    warning('Problem using function. Assigning a value of 0.');
    a = 0;
End
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
OUTPUT:
Warning: Problem using function.  Assigning a value of 0.
```

Python

```
try:
                x =1/0
except Exception as e:
                print('Error: '+str(e))
##############################
OUTPUT:
Error: division by zero
```

Error Types in Python

Exception	Description
IndexError	When the wrong index of a list is retrieved.
AssertionError	It occurs when the assert statement fails
AttributeError	It occurs when an attribute assignment is failed.
ImportError	It occurs when an imported module is not found.
KeyError	It occurs when the key of the dictionary is not found.
NameError	It occurs when the variable is not defined.
MemoryError	It occurs when a program runs out of memory.
TypeError	It occurs when a function and operation are applied in an incorrect type.

For complete list see: https://docs.python.org/3/tutorial/errors.html

Functions

Functions

MATLAB

```
function ave = average(x)
        % function to find the average of a list
        ave = sum(x(:))/numel(x);
end

m = [5,3,2];
average(x)
```

Python

```
def average(x):
        # function to find the average of a list
        ave = sum(x) / len(x)
        return ave

m = [5,3,2]
print(average(x))
```

Both Display
5

Recursion

- https://www.google.com/search?client=firefox-b-1-d&sxsrf=ALiCzsbUPTBDEAUY835ELoxB-EYGBWDaqQ:1659463125939&q=recursion&spell=1&sa=X&ved=2ahUKEwi_uY7x3aj5AhUVFFkFHT7EBXcQBSgAegQIAhAz&biw=1620&bih=919&dpr=2

Classes and Methods

Creating a Simple Classes

```
class MyClass:
        x = 5
```

Creating an Object

p = MyClass()

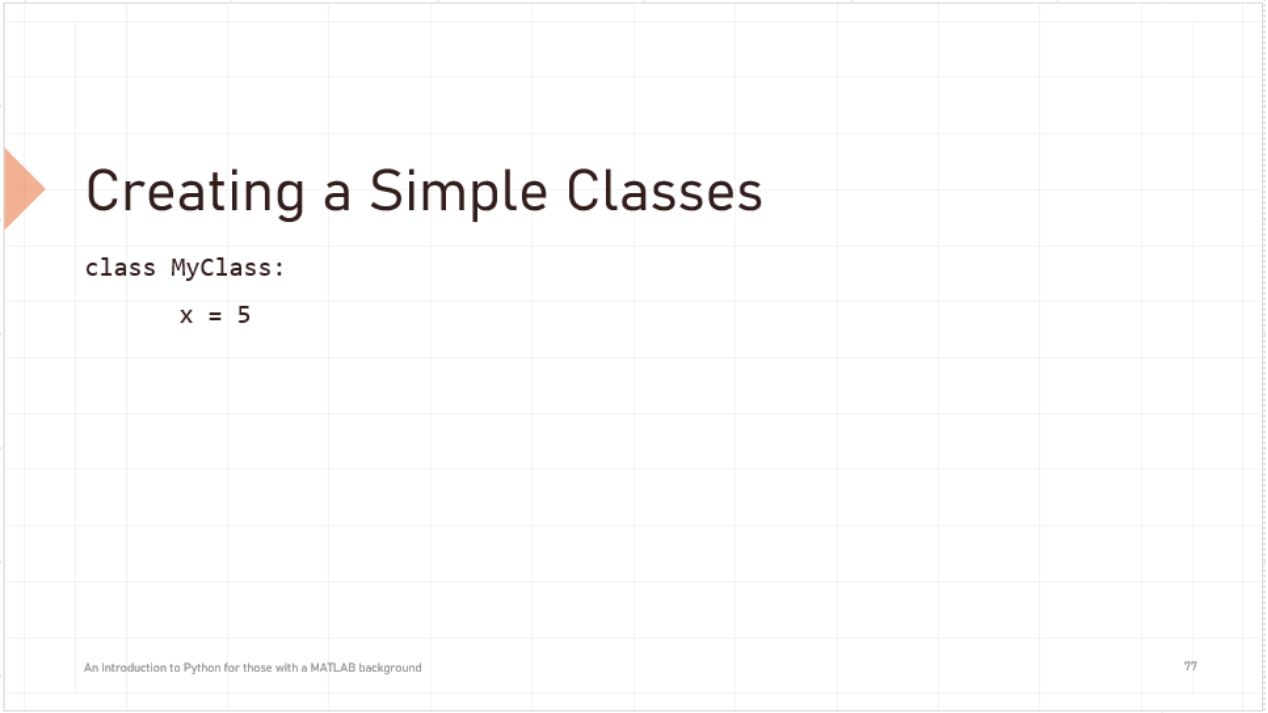
print(p.x)

OUTPUT: 5

p.x = 6

print(p.x)

OUTPUT: 6



The __init__() Function

```
class person:
        def __init__(self, name: str, age: int):
                        ^parameters
            self.name :str = name
            self. age : int = age
```

The __init__() Function cont.

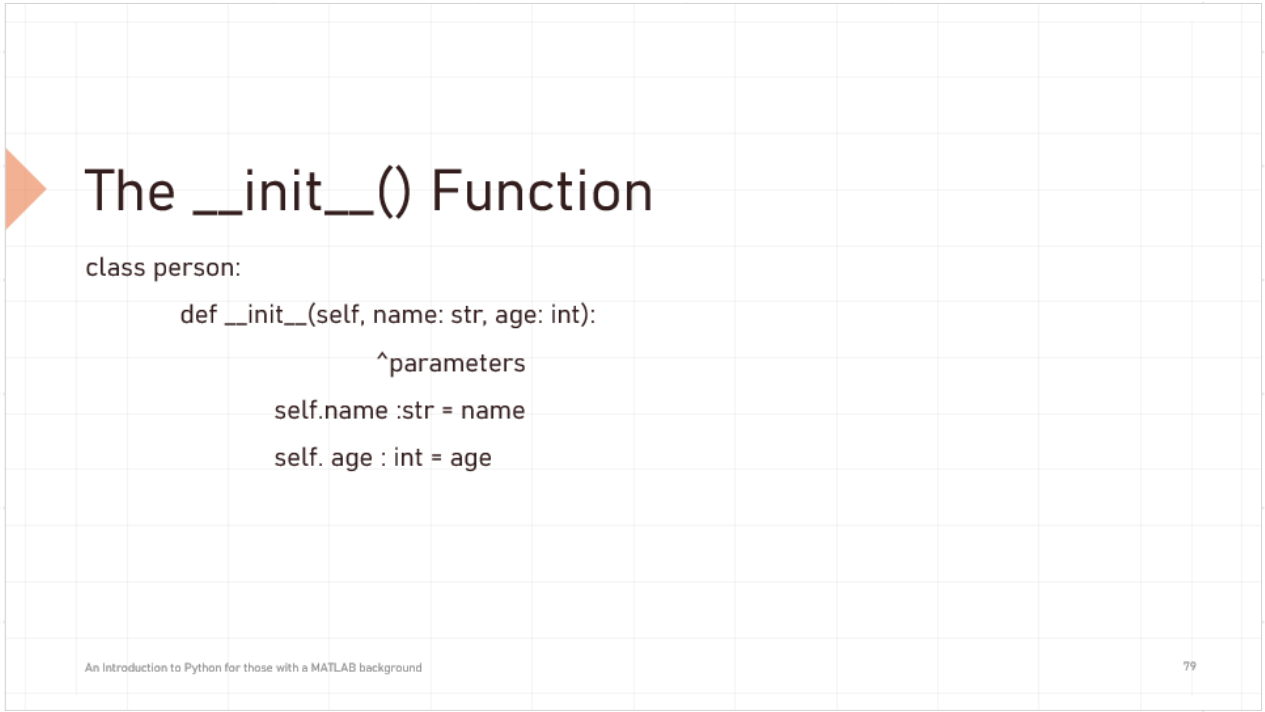
p1 = person('sam', 42)

print(p1.name)

OUTPUT: sam

print(p1.age)

OUTPUT: 42



The self Parameter

The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

Object Methods

```
class person:
        def __init__(self, name: str, age: int):
                self.name :str = name
                self. age : int = age
        def nameTag(self):
                return str("hello my name is " + self.name)
```

Object Methods cont.

p1 = person('sam',42)

print(p1.nameTag())

OUTPUT: hello my name is sam

Object Methods

```
class person:
        def __init__(self, name: str, age: int):
                self.name :str = name
                self. age : int = age
        def nameTag(self):
                return str("hello my name is " + self.name)
```

Object Methods cont.

```
class person:
        def __init__(self, name: str, age: int):
                self.name :str = name
                self. age : int = age
        def nameTag(self):
                return str("hello my name is " + self.name)
        def displayNameTag(self):
                print(self.nameTag())
```

Object Methods cont.

p1 = person('sam',42)

p1.displayNameTag()

OUTPUT: hello my name is sam

Object Methods cont.

```
class person:
        def __init__(self, name: str, age: int):
                self.name :str = name
                self. age : int = age
        def nameTag(self):
                return str("hello my name is " + self.name)
        def displayNameTag(self):
                print(self.nameTag())
```

Modifying Object Properties

p1 = person('sam',42)

p1.displayNameTag()

OUTPUT: hello my name is sam

p1.name = 'joe'

p1.displayNameTag()

OUTPUT: hello my name is joe

Object Methods cont.

```
class person:
        def __init__(self, name: str, age: int):
                self.name :str = name
                self. age : int = age
        def nameTag(self):
                return str("hello my name is " + self.name)
        def displayNameTag(self):
                print(self.nameTag())
```


An Introduction to Python for those with a MATLAB background

The __str__() Method

```
class person:
        def __init__(self, name: str, age: int):
                self.name :str = name
                self. age : int = age
        def nameTag(self):
                return str("hello my name is " + self.name)
        def displayNameTag(self):
                print(self.nametag())
        def __str__(self):
                string = 'Name: ' + str(name)+'\nAge: ' + str(self.age)
                return string
```

The __str__() Method

```
class person:
        def __init__(self, name: str, age: int):
                self.name :str = name
                self. age : int = age
        def nameTag(self):
                return str("hello my name is " + self.name)
        def displayNameTag(self):
                print(self.nametag())
        def __str__(self):
                string = 'Name: ' + str(name)+'\nAge: ' + str(self.age)
                return string
```

There are two errors, what are they?

The __str__() Method

```
class person:
        def __init__(self, name: str, age: int):
                self.name :str = name
                self. age : int = age
        def nameTag(self):
                return str("hello my name is " + self.name)
        def displayNameTag(self):
                print(self.nametag())
        def __str__(self):
                string = 'Name: ' + str(name)+'\nAge: ' + str(self.age)
                return string
```

The __str__() Method

```
class person:
        def __init__(self, name: str, age: int):
                self.name :str = name
                self. age : int = age
        def nameTag(self):
                return str("hello my name is " + self.name)
        def displayNameTag(self):
                print(self.nameTag()) << error 1 Python is case senative
        def __str__(self):
                string = 'Name: ' + str(self.name)+'\nAge: ' + str(self.age) << error 2 missing self
                return string
```

The __str__() Method

```
class person:
    def __init__(self, name: str, age: int):
        self.name :str = name
        self. age : int = age
    def nameTag(self):
        # return name tag
        return str('hello my name is ' + self.name)
    def displayNameTag(self):
        # display name tag
        print(self.nameTag())
    def __str__(self):
        string = 'Name: ' + str(self.name)+'\nAge: ' + str(self.age)
        return string
```

The __str__() Method cont.

p1 = person('sam',42)

print(p1)

OUTPUT:

"

Name: sam

Age: 42

"

The __str__() Method

```
class person:
    def __init__(self, name: str, age: int):
        self.name :str = name
        self. age : int = age
    def nameTag(self):
        # return name tag
        return str('hello my name is ' + self.name)
    def displayNameTag(self):
        # display name tag
        print(self.nameTag())
    def __str__(self):
        string = 'Name: ' + str(self.name)+'\nAge: ' + str(self.age)
        return string
```

Classes Summary

```
class person:

    def __init__(self, name: str, age: int):

        self.name :str = name

        self. age : int = age

    def nameTag(self):

        # return name tag

        return str('hello my name is ' + self.name)

    def displayNameTag(self):

        # display name tag

        print(self.nameTag())

    def __str__(self):

        string = 'Name: ' + str(self.name)+'\nAge: ' + str(self.age)

        return string
```

```
p1 = person('John', 20)
p1.displayNameTag()
print(p1)
"""
OUTPUT:
hello my name is John
Name: John
Age: 20
"""
p2 = person('Jane', 21)
p2.displayNameTag()
print(p2)
"""
OUTPUT:
hello my name is Jane
Name: Jane
Age: 21
"""
p1.name = 'Sam'
p1.displayNameTag()
p2.displayNameTag()
"""
OUTPUT:
hello my name is Sam
hello my name is Jane
"""
```

Object Orientated Programming

Object Orientated Program

- Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior. [3]

OOP cont.

- **Classes** are user-defined data types that act as the blueprint for individual objects, attributes and methods.
- **Objects** are instances of a class created with specifically defined data. Objects can correspond to real-world objects or an abstract entity. When class is defined initially, the description is the only object that is defined.
- **Methods** are functions that are defined inside a class that describe the behaviors of an object. Each method contained in class definitions starts with a reference to an instance object. Additionally, the subroutines contained in an object are called instance methods. Programmers use methods for reusability or keeping functionality encapsulated inside one object at a time.
- **Attributes** are defined in the class template and represent the state of an object. Objects will have data stored in the attributes field. Class attributes belong to the class itself.

OOP cont.

- **Inheritance.** Classes can reuse code from other classes. Relationships and subclasses between objects can be assigned, enabling developers to reuse common logic while still maintaining a unique hierarchy. This property of OOP forces a more thorough data analysis, reduces development time and ensures a higher level of accuracy.

OOP cont.

- Common OOP languages
  - Java
  - Python
  - C++

Object Orientated Design

```
class myPerson(person):
    # parent class is person
    def __init__(self, name: str, age: int, hair_color: str):
        super().__init__(name, age) # call parent class constructor
        self.hair_color = hair_color
    def __str__(self): # override parent class __str__ method
        string = 'Name: ' + str(self.name)+'\nAge: ' + str(self.age) + '\nHair Color: ' + str(self.hair_color)
        return string
```

Object Orientated Design cont.

```
p3 = myPerson('John', 20, ‘red')
print(p3.name)
print(p3.age)
print(p3.hair_color+'\n')
p3.displayNameTag()
print('\n'+str(p3))
"""
OPUTPUT:
John
20
brown

hello my name is John

Name: John
Age: 20
Hair Color: red
"""
```

Object Orientated Design

```
class myPerson(person):

    # parent class is person

    def __init__(self, name: str, age: int, hair_color: str):

        super().__init__(name, age) # call parent class constructor

        self.hair_color = hair_color

    def __str__(self): # override parent class __str__ method

        string = 'Name: ' + str(self.name)+'\nAge: ' + str(self.age) + '\nHair Color: ' + str(self.hair_color)

        return string
```

Python Modules

Import

- ‘import’ can be used to include a set of classes, variables or functions from another file
- This is useful so that scrips can be separate.

Import cont.

```
import math

print(math.cos(5)) # 0.28366218546322625
```

Creating a Module

File Name: myMath.py

File Content:

```
def cos(angle:float) -> float:
        """
        Returns the cosine of an angle in degrees
        param: angle:float (degrees)
        return: float
        """
        return math.cos(math.degrees(angle))
```

Creating a Module cont.

```
import myMath

print(myMath.cos(5)) #0.9961946980917455
```

Creating a Module

File Name: myMath.py

File Content:

```
def cos(angle:float) -> float:
        """
        Returns the cosine of an angle in degrees
        param: angle:float (degrees)
        return: float
        """
        return math.cos(math.degrees(angle))
```

Python for Data Analysis

An introduction to NumPy and Matplotlib

Data Analysis

- Numpy can be use for matrix operations similar to MATLAB
- https://numpy.org/doc/stable/user/absolute_beginners.html
- https://www.w3schools.com/python/numpy/default.asp

Data Presentation

- Matlibplot is similar to how plotting is done in matlab
- https://matplotlib.org/
- https://www.w3schools.com/python/matplotlib_intro.asp

Python Resources

- python.org Official Python Website
- w3schools Python Tutorial
- w3schools Python Reference
- Python in VS code
- Python in PyCharm
- Download Python
- PyPi Python Package Installer

References

- [1] – https://www.mathworks.com/discovery/what-is-matlab.html
- [2] – https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python
- [3] – https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP

Release Log

- Initial Release (8/2/2022 – Michael Dompke)