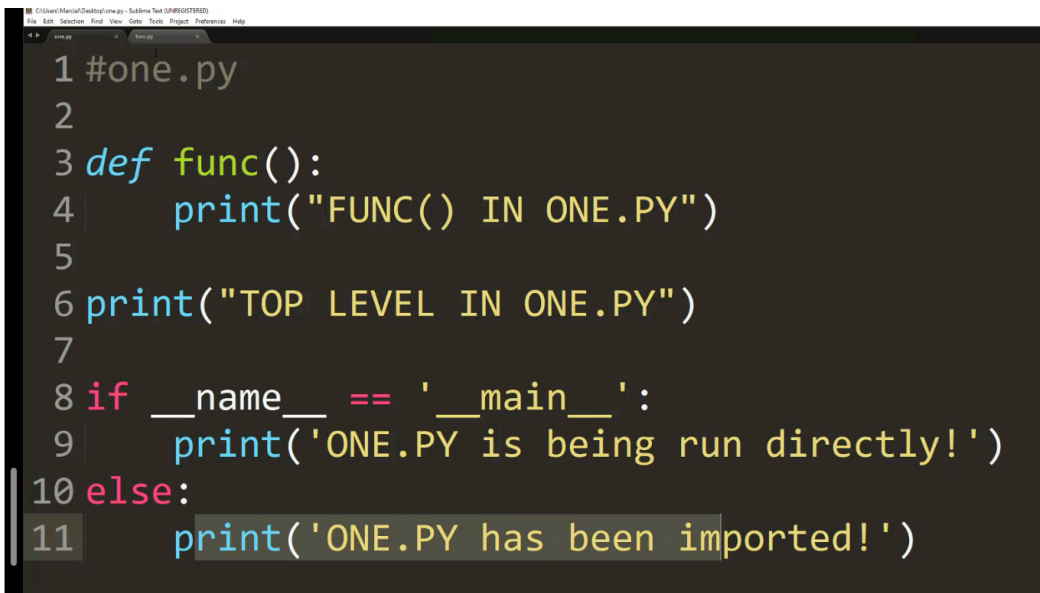


SECTION 9: MODULES & PACKAGES - 29 minutes, 3 parts

- -> `__name__` and `"__main__"`
 - -> For larger .py scripts
 - -> `if __name__ == "__main__"` <- This is at the end of those larger .py scripts
 - -> We don't know if we are using the original .py file where that function was defined, or if it were defined in another .py file
 - -> In this example, we are creating two .py files to show this
 - -> He's created two new files -> one.py and two.py
 - -> `print('hello')` <- This runs the code
 - -> On the same indentation level
- -> **At the end of the script, `__name__` <- This is at the end of the script so it stops us from overriding it by accident**
 - -> This is assigned a string
 - -> `__name__ == "__main__"` <- This is done in the background
 - -> We have an if block to check if they are equal to each other
 - -> **It tells us if the current .py file is being run directly**
 - -> If the builtin variable equals the `__main__` string
 - -> He's defined a function in one of the two .py files
- -> **Sublime does autocomplete on if blocks like this**



```
1 #one.py
2
3 def func():
4     print("FUNC() IN ONE.PY")
5
6 print("TOP LEVEL IN ONE.PY")
7
8 if __name__ == '__main__':
9     print('ONE.PY is being run directly!')
10 else:
11     print('ONE.PY has been imported!')
```

- -> If one.py is the main script or not
- -> Depending on whichever .py file we are in, run this or this piece of Python

- -> He's then written the content of the second .py file

```
1 #two.py
2 import one
3
4 print("TOP LEVEL IN TWO.PY")
5
6 one.func()
7
8 if __name__ == '__main__':
9     print("TWO.PY is being run directly!")
10 else:
11     print("TWO.PY has been imported!")
```

- -> Then in the terminal -> He runs the .py file
 - -> One of them
 - -> Then he runs the other one
 - -> This lets us see if something is being imported or run directly
 - -> We are checking if the __name__ variable is in the current .py file
- -> We are defining the functions / classes at the top of the .py file, and then the __name__ == '__main__' block at the bottom of the file
 - -> This is for organisation
 - -> Defining the functions at the top of the file and then putting all of the times they are called under the same __name__ == '__main__' block at the bottom of the code