Make a Python Package Using Class by Reference

by

Christopher Andrew Topalian

Dedicated
to
God the Father

How to Make a Python Package (Library)

Let's make a library of functions for us to use in all of our projects, which we will install system wide on our computer. Later, we can even choose to upload our package to the internet and share it with the world.

By making a package of functions we can update our package anytime that we want and all of our projects will utilize the same updated library!

This makes programming much easier, because we only have to make the library one time, instead of over and over again.

This allows us to be much more productive, because the package can easily be imported after it has been installed system wide using pip install .

Let's make a package named cos.
On the next pages we walk through creating the package that we name cos.

cos stands for College of Scripting :-)

Class by Reference means that we are building the class from functions in other files.

This way is good for people who want to make a namespace and want to make instances, but don't require state variables.

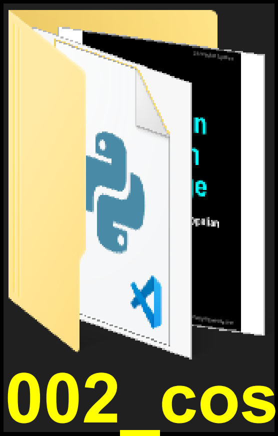
The advantage of class by reference is that we are building a library of functions in separate files, which many people like to do, for easy organization, and we achieve a namespace by using our class by reference, that can also create instances. But, remember that no state variables are available to the instances of the class that we create by using class by reference.

Thus, if we instead do need a class to allow for both instances and for state variables in the class, then we have to INSTEAD put the methods directly in the one class file, instead of class by reference using separate files.

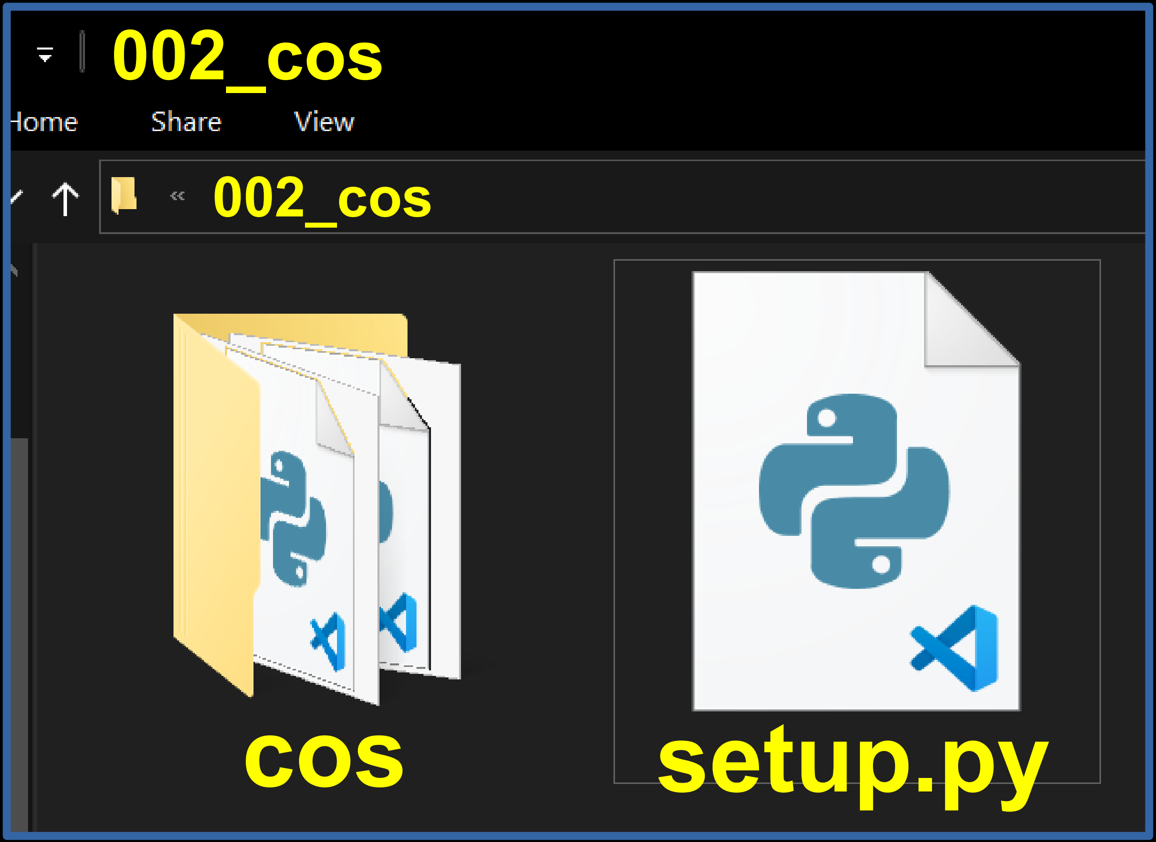
This book teaches class by reference, so remember that we are creating a namespace and we can create instances, but no state variables.

How to Create a Python Package

* We create a folder, named
002_cos



* In 002_cos folder we make another folder named cos and make a file named setup.py



setup.py is used to Install our Package

We create a setup.py file, which defines our package and its metadata. This file is essential for distributing and installing our library across projects.

First, we make a new text file in VSCode Editor and type the script that we see on the next page and save it as setup.py

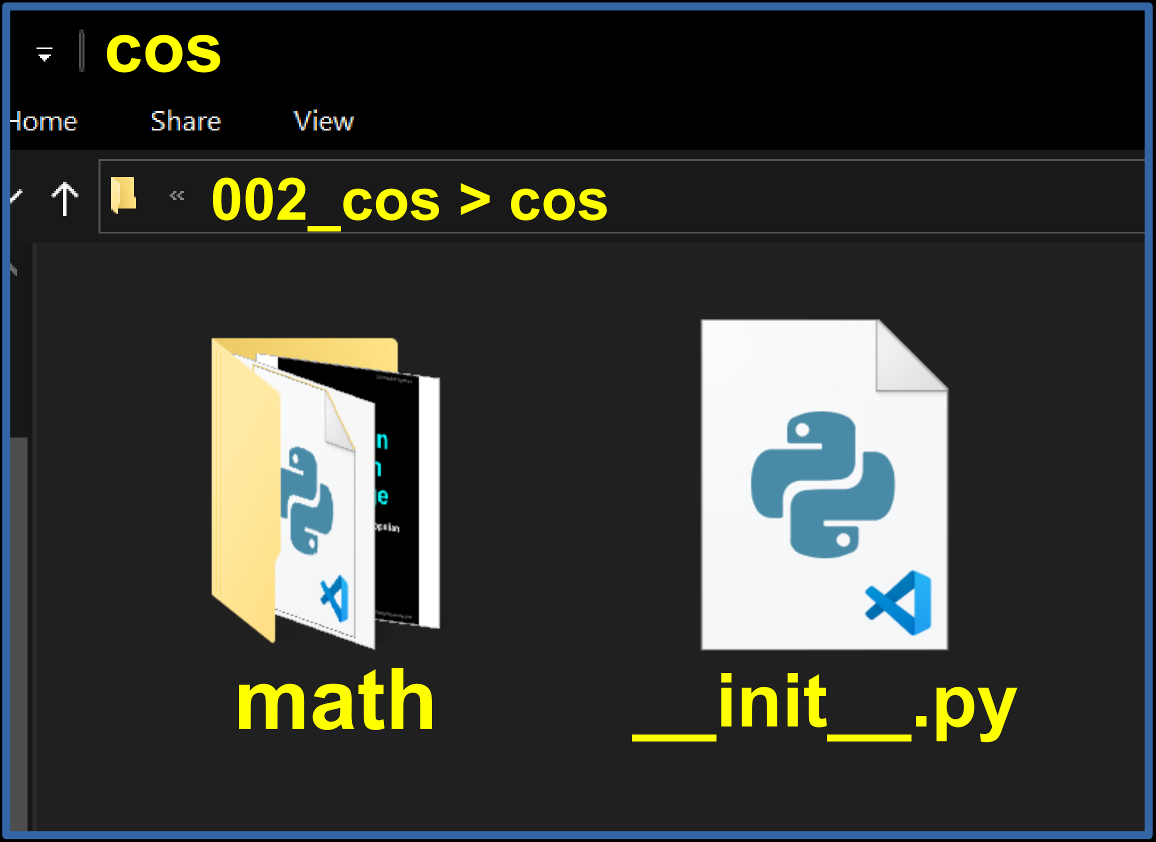
NEXT PAGE SHOWS setup.py

```
# setup.py

from setuptools import setup, find_packages

setup(
    name = 'cos',
    version = '0.1',
    author = 'Christopher Andrew Topalian',
    packages = find_packages(),
)

####

# Dedicated to God the Father
# All Rights Reserved Christopher Andrew Topalian
Copyright 2000-2024
# https://github.com/ChristopherTopalian
# https://github.com/ChristopherAndrewTopalian
# https://sites.google.com/view/CollegeOfScripting
```

* In the cos folder, we have 1 file and 1 folder:

__init__.py

math



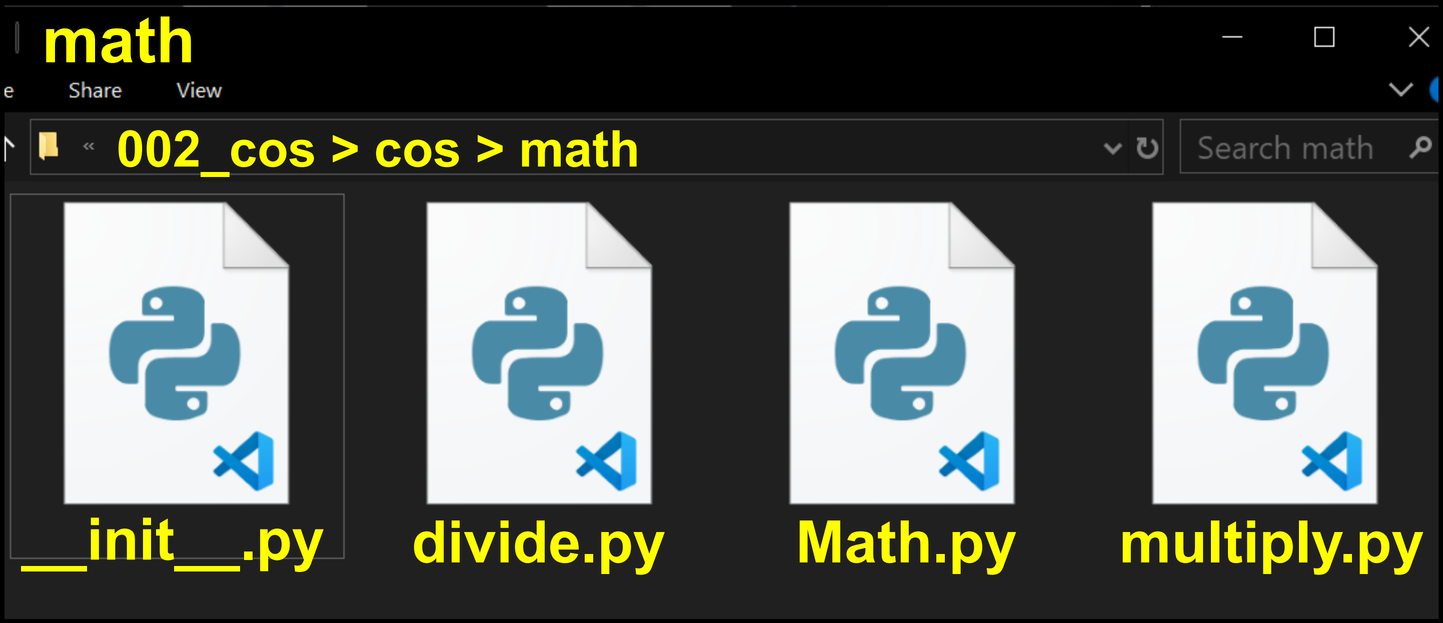
The file is named __init__.py

It has two underscores before and after the word init

We save our file as __init__.py

```
# __init__.py

from .math.Math import Math

####

# Dedicated to God the Father
# All Rights Reserved Christopher Andrew Topalian
Copyright 2000-2024
# https://github.com/ChristopherTopalian
# https://github.com/ChristopherAndrewTopalian
# https://sites.google.com/view/CollegeOfScripting
```

Python files that we add to our math folder:



We place multiply.py and divide.py inside of the math folder.

Thus, the two functions we are adding to our library are: multiply.py and divide.py

We also add a class file named Math.py

This class file will reference the functions named divide and multiply.

We also add a blank __init__.py file.
This __init__.py is left intentionally blank.

This blank __init__.py file must be located in the math folder to package things correctly.

On the next page, we see the function script that we save as multiply.py

NEXT PAGE SHOWS multiply.py

```
# multiply.py

def multiply(a, b):
  return a * b

##

if __name__ == "__main__":
  print(multiply(4, 4))
  input('')

#####

# Dedicated to God the Father
# All Rights Reserved Christopher Andrew
Topalian Copyright 2000-2024
# https://github.com/ChristopherTopalian
# https://github.com/ChristopherAndrewTopalian
# https://sites.google.com/view/CollegeOfScripting
```

On the next page, we see the function script that we save as divide.py

NEXT PAGE SHOWS divide.py

```
# divide.py

def divide(a, b):
    return a / b

##

if __name__ == "__main__":
    print(divide(10, 2))
    input('')

####

# Dedicated to God the Father
# All Rights Reserved Christopher Andrew
Topalian Copyright 2000-2024
# https://github.com/ChristopherTopalian
# https://github.com/ChristopherAndrewTopalian
# https://sites.google.com/view/CollegeOfScripting
```

On the next page, we see our class file that we named Math.py

This Math.py file is assigns functions from separate files, as static methods of our class.

NEXT PAGE SHOWS Math.py

```
# Math.py


from .divide import divide
from .multiply import multiply

class Math:
    # assigning functions as static methods
    divide = staticmethod(divide)
    multiply = staticmethod(multiply)

##

# Dedicated to God the Father
# All Rights Reserved Christopher Andrew Topalian
Copyright 2000-2024
# https://github.com/ChristopherTopalian
# https://github.com/ChristopherAndrewTopalian
# https://sites.google.com/view/CollegeOfScripting
```

On the next page, we see the file that we save as __init__.py

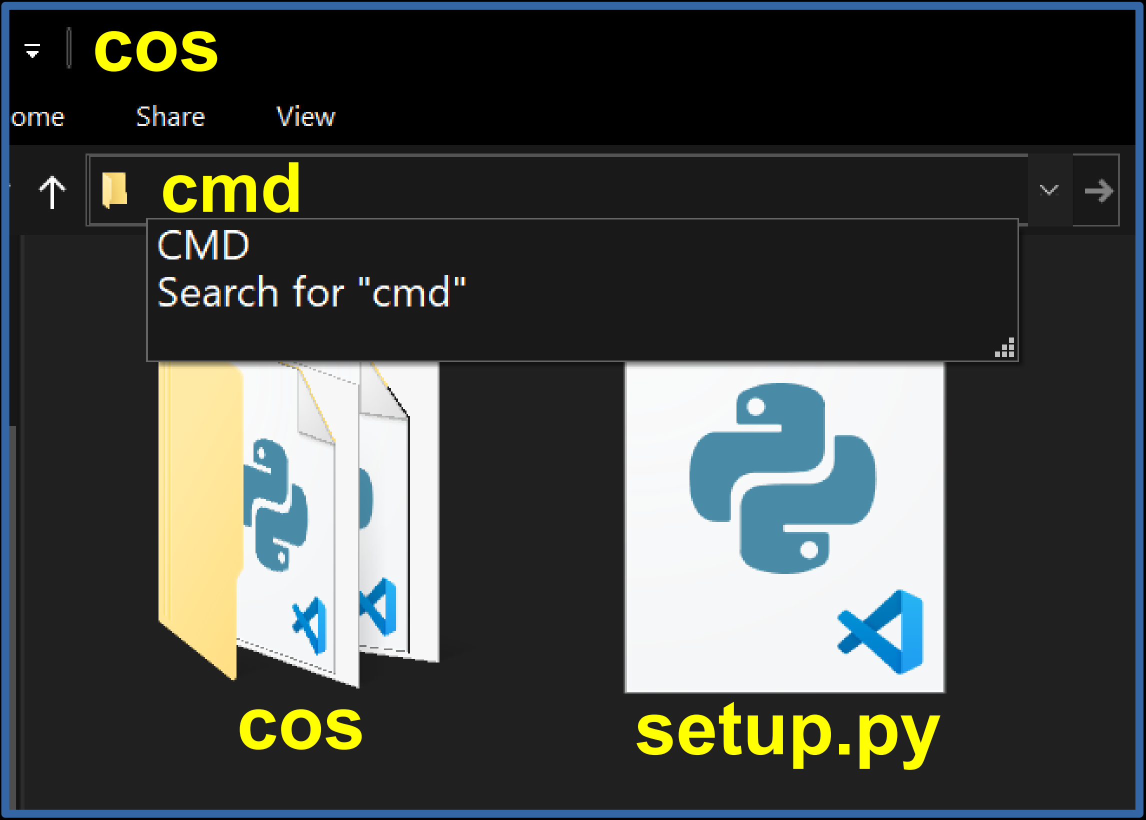
This __init__.py file is left blank intentionally and located inside of the math folder.

NEXT PAGE SHOWS __init__.py

```python
# __init__.py



# Dedicated to God the Father
# All Rights Reserved Christopher Andrew Topalian Copyright 2000-2024
# https://github.com/ChristopherTopalian
# https://github.com/ChristopherAndrewTopalian
# https://sites.google.com/view/CollegeOfScripting
```

How to Install Our Package Computer Wide

We type cmd into the cos folder address bar and press Enter



We type in the Command prompt:

```
pip install .
```

press Enter

This installs our Package system wide!

Now, we can easily import our package into any of our Python projects. This makes creating and updating big applications very easy :-)

Now, let's make a new script and import and use the functions from our package that we have created and have installed worldwide on our computer system.

We make a new script in VSCode and save it as, usesOurPackage.py

NEXT PAGE SHOWS usesOurPackage.py

```
# usesOurPackage.py

from cos import Math

# using as class attributes
print(Math.divide(10, 2))
print(Math.multiply(3, 4))

# using an instance
math_instance = Math()
print(math_instance.divide(10, 2))
print(math_instance.multiply(3, 4))

##

# Dedicated to God the Father
# All Rights Reserved Christopher Andrew Topalian
Copyright 2000-2024
# https://github.com/ChristopherTopalian
# https://github.com/ChristopherAndrewTopalian
# https://sites.google.com/view/CollegeOfScripting
```

Updating Our Package

When we make changes and want to update our package with new functions or changes to our existing functions:

We Don't Update the package using:

```
pip install -e .
```

press Enter

Instead, it is easier to uninstall the package and then install it again to avoid conflicts.

We open the system wide command prompt and type:

```
pip uninstall cos
```

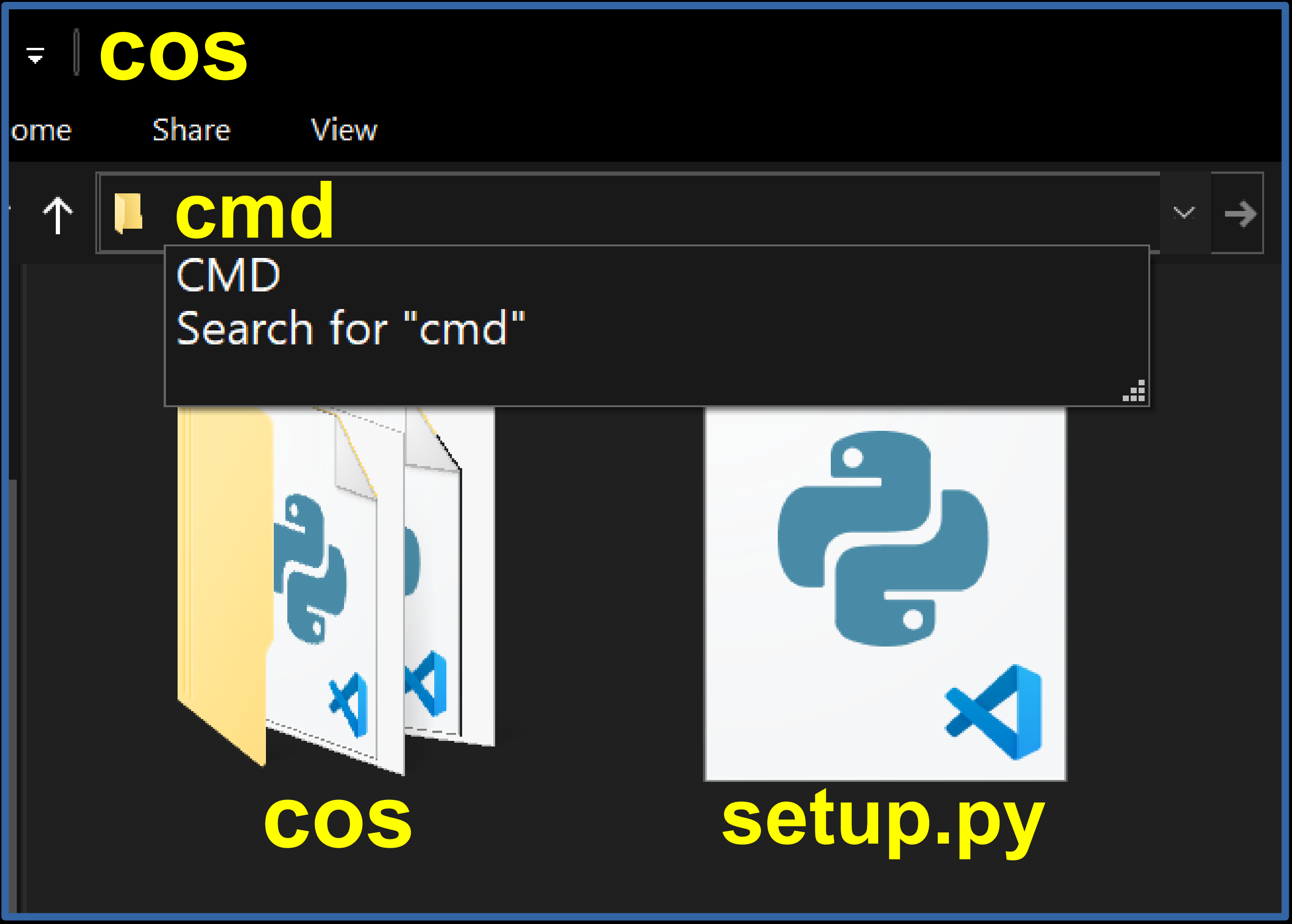
press Enter

This uninstalls the cos package from the system.

NEXT PAGE SHOWS:
Installing our cos package again

We Repeat the Install Process like before:

We type cmd into the cos folder address bar and press Enter



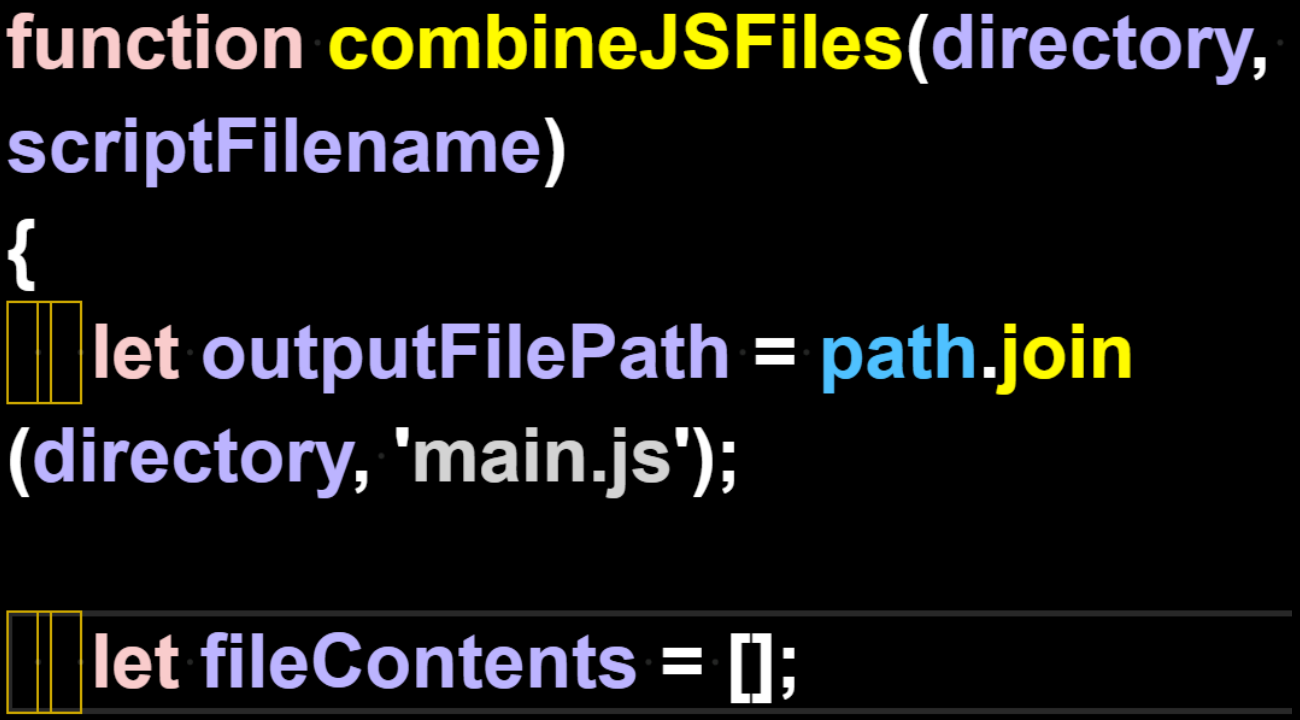
This again installs our package system wide. Now, any changes that we have made, such as any additional functions included, are now available in our library of functions.

Now, we can import our package from anywhere in our Python environment on our computer and use the functions from our package in any script that we are making :-)

How to Paste Code from a PDF that has Junk Characters.

How to Paste Code from a PDF that has Junk Characters.

When we paste from a pdf into VSCode, it might look like this:

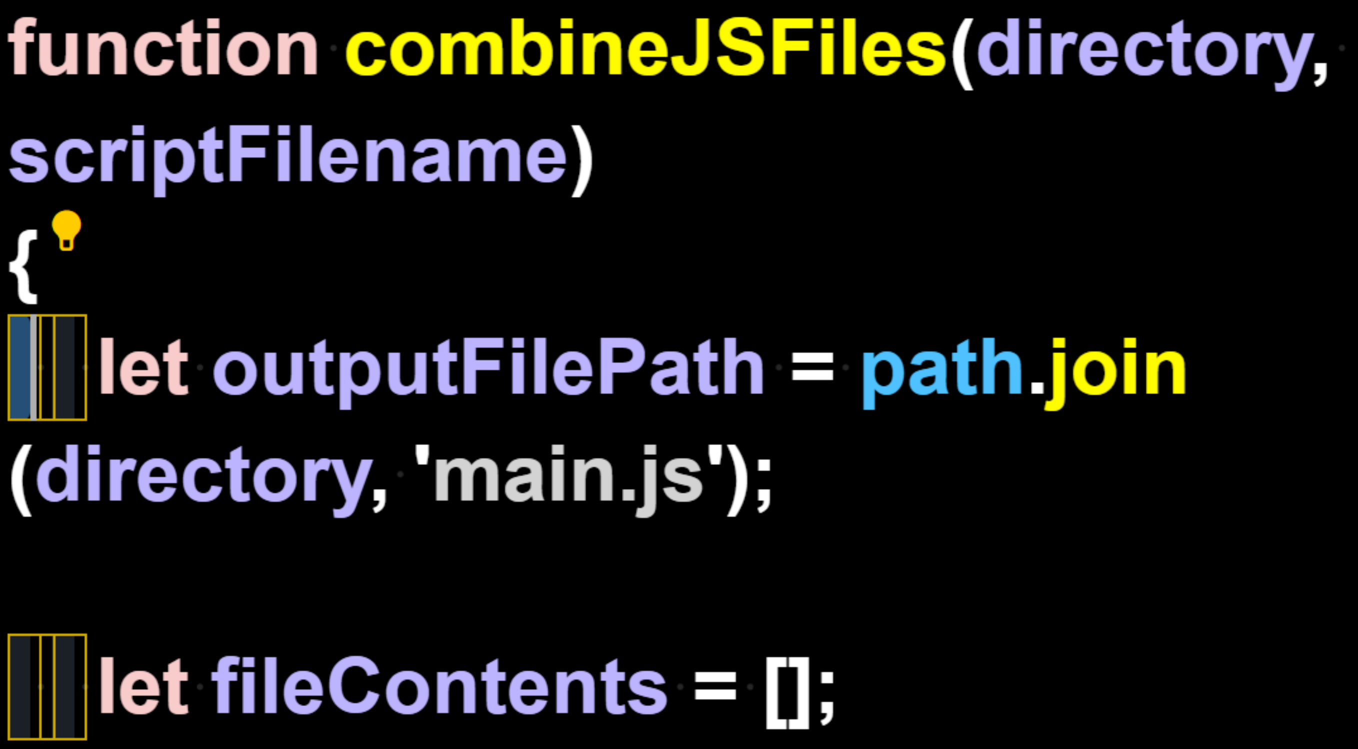


```
function combineJSFiles(directory,
scriptFilename)
{
  let outputFilePath = path.join
(directory, 'main.js');

  let fileContents = [];
```

We can't leave those junk characters in the code, so we remove them with find/replace.

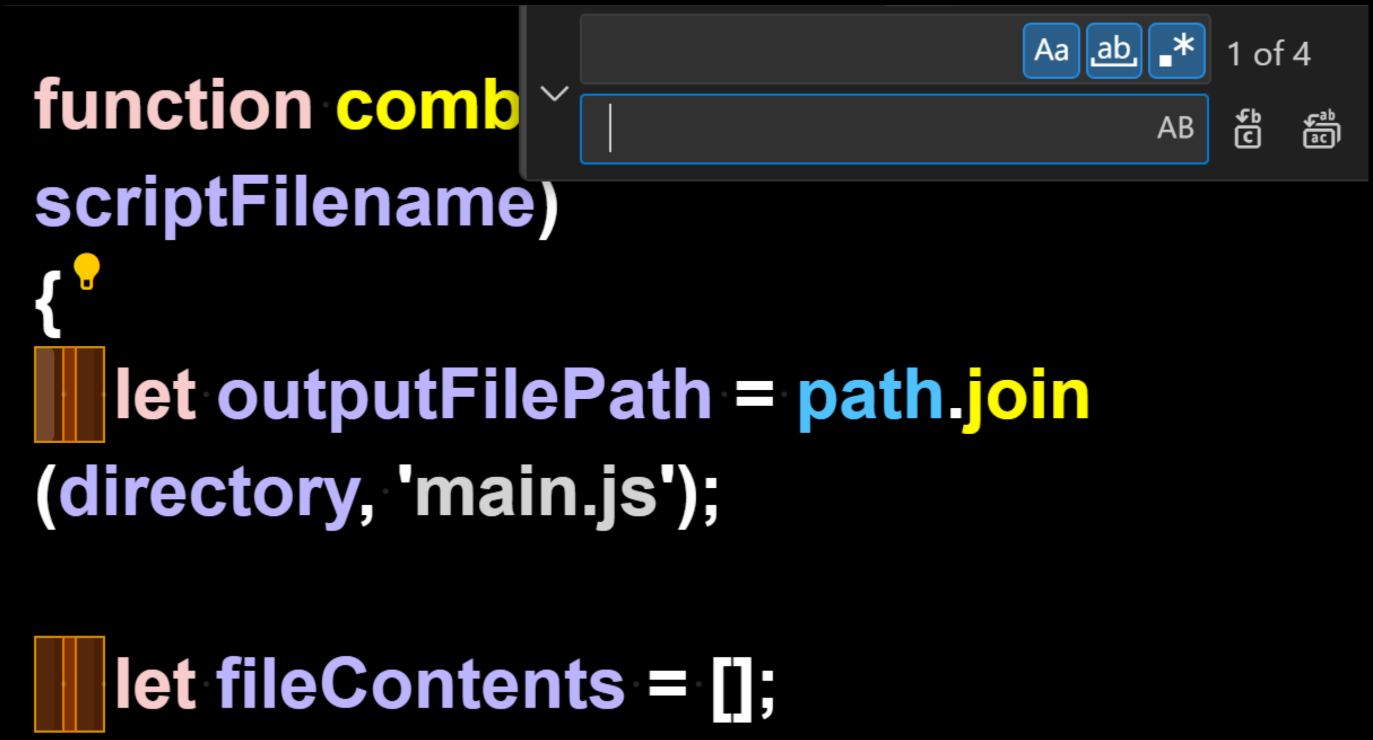
We Find 1 of the spaces.
We Replace All with the 1 space that we typed.
This gets rid of the junk characters in the code.

We highlight 1 space with our mouse arrow:



```
function combineJSFiles(directory, scriptFilename)
{
    let outputFilePath = path.join(directory, 'main.js');

    let fileContents = [];
```

We press **Control + H** to open the Find/Replace feature and **Replace All** with our own **Space**



```
function combineJSFiles(directory, scriptFilename)
{
    let outputFilePath = path.join(directory, 'main.js');

    let fileContents = [];
```

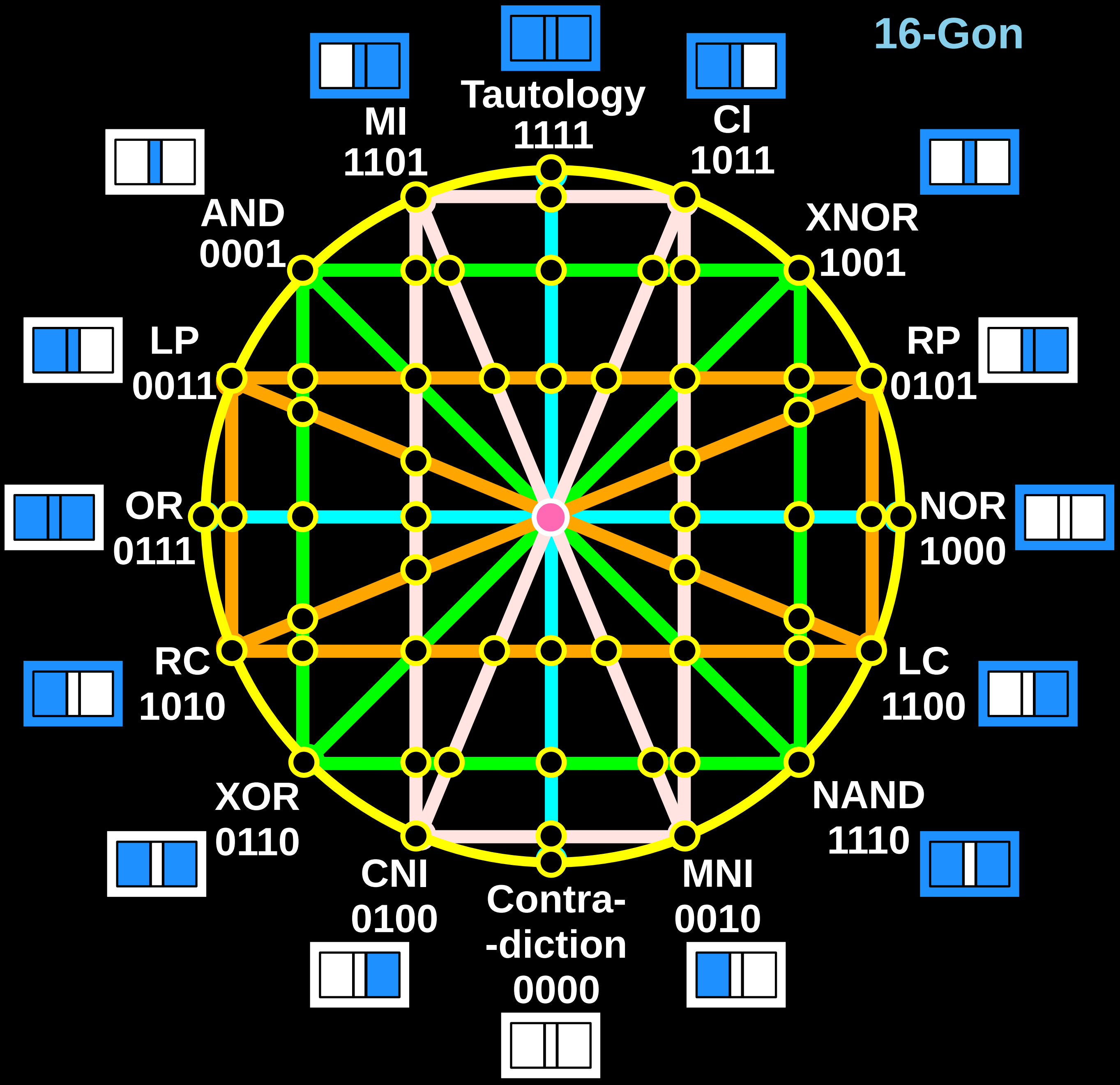
Here we see that the Find/Replace All has replaced the junk characters with our working spaces instead:

```
function combineJSFiles(directory, 
scriptFilename)
{
    let outputFilePath = path.join
    (directory, 'main.js');

    let fileContents = [];
```

Now that the code
has no junk characters,
it can run.

Christopher Topalian

True Artificial Intelligence System



For More Tutorials:

GitHub.com/ChristopherTopalian

GitHub.com/ChristopherAndrewTopalian

Sites.google.com/view/CollegeOfScripting

CollegeOfScripting.weebly.com

CollegeOfScripting.wordpress.com

Youtube.com/ScriptingCollege

Twitter.com/CollegeOfScript

Rumble.com/user/CollegeOfScripting

Dedicated to God the Father

This book is created by the
College of Scripting Music & Science.
Always remember, that each time you write a script with a pencil and paper, it becomes imprinted so deeply in memory that the material and methods are learned extremely well.

When you Type the scripts, the same is true.
The more you type and write out the scripts by keyboard or pencil and paper, the more you will learn programming!

Write and Type every example that you find.
Keep all of your scripts organized.
Every script that you create increases your programming abilities.
SEEING CODE, is one thing,
but WRITING CODE is another.
Write it, Type it, Speak it, See it, Dream it.

CollegeOfScripting.weebly.com