

Creating First file in Python

What is the significance of keeping your code organised? Assume you're working on a large project with hundreds or thousands of lines of code. Having all of that code in a single file can quickly become overwhelming. This is where breaking down your code into multiple files comes in handy.

You can easily locate specific sections of your code when making changes or fixing bugs by dividing your code into logical units and placing each unit in its own file. This increases maintainability and saves you time and frustration in the future.

So, as you begin your coding adventure, keep in mind the importance of organising your code in separate files. It will facilitate your work as a developer.

Creating a New File

In Python, comments are like notes that help us explain our code for better understanding. They are not executed by the computer but provide valuable information for developers. There are two ways to add comments in Python:

- a) Single-line comments
- b) Multi-line comments

So for a single line comment we use the `#` symbol to add a single-line comment. Everything after the `#` symbol on that line will be treated as a comment.

If you need to write more than one line of comments, it's like writing a longer note or explanation. In Python, you can use three single quotes (`'''`) or three double quotes (`"""`) to start and end a multi-line comment.

Let's understand both with an example:

```
# This is a single-line comment
```

```
''' This is a multi-line comment using single triple quotes. It spans  
across multiple lines. '''
```

""" Another way to create a multi-line comment. This is using triple double-quotes. """

```
Commenting in python

2 ways to add comments

Single line comments

and multiline comment

1. Single line comment

# this is a single line comment

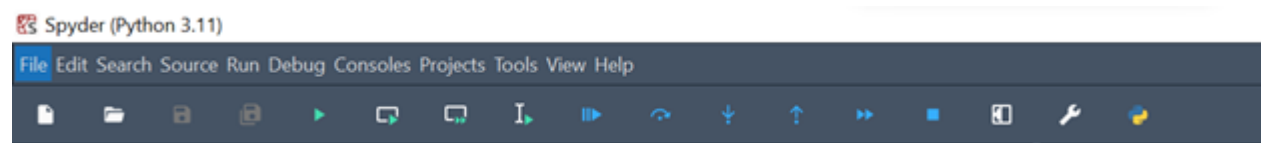
2. Multi line comment

''' This is multi line comment using single quotes.
It spans across multiple lines.'''

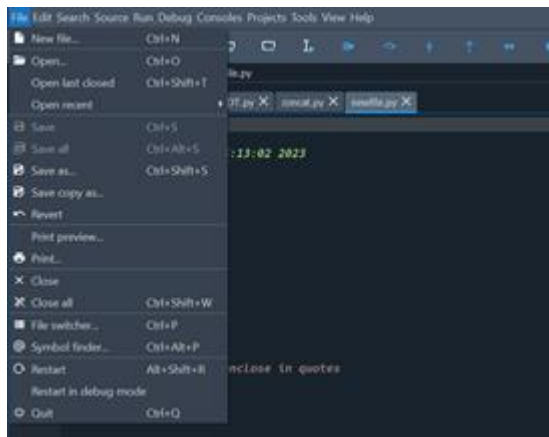
"""another way to create a multiline comment
Using triple double quotes"""
```

To create a new Python file:

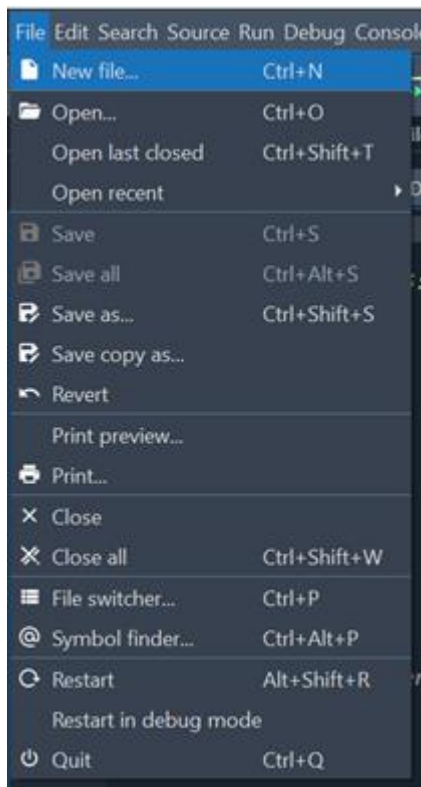
Step 1: Look at the menu bar at the top of the Spyder IDE window. Locate the "File" menu.



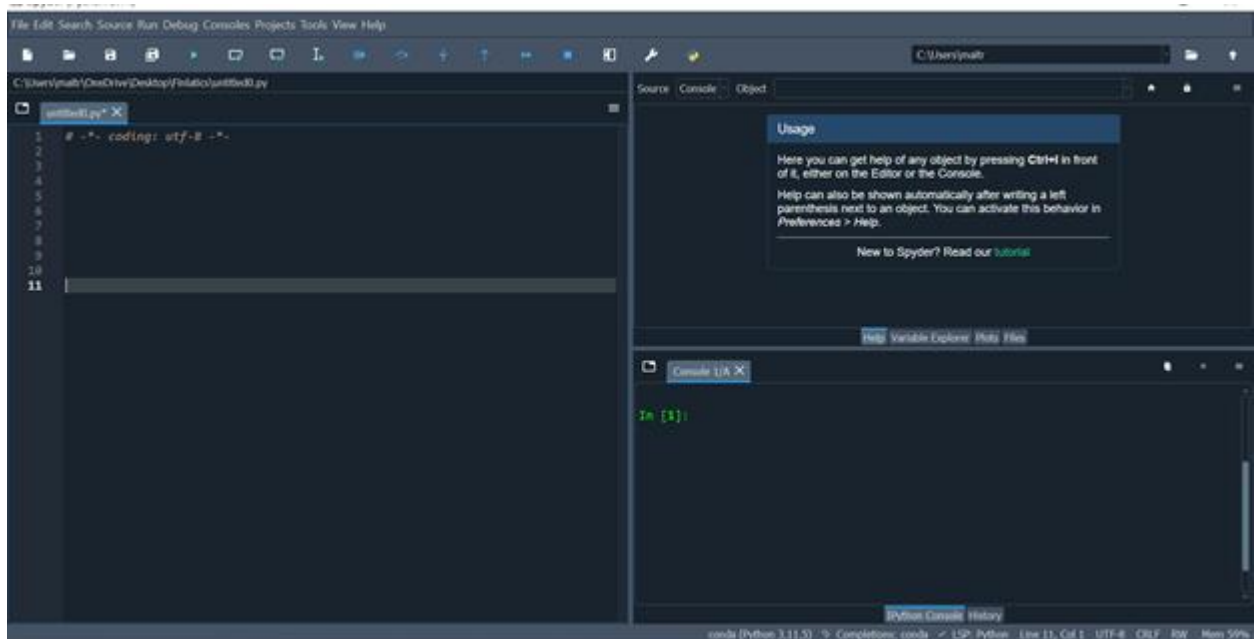
Step 2: Click on the "File" menu to reveal a dropdown list of options.



Step 3: From the dropdown list, click on "New File" or use the appropriate keyboard shortcut, which is usually "Ctrl+N" or "Cmd+N" on Mac.

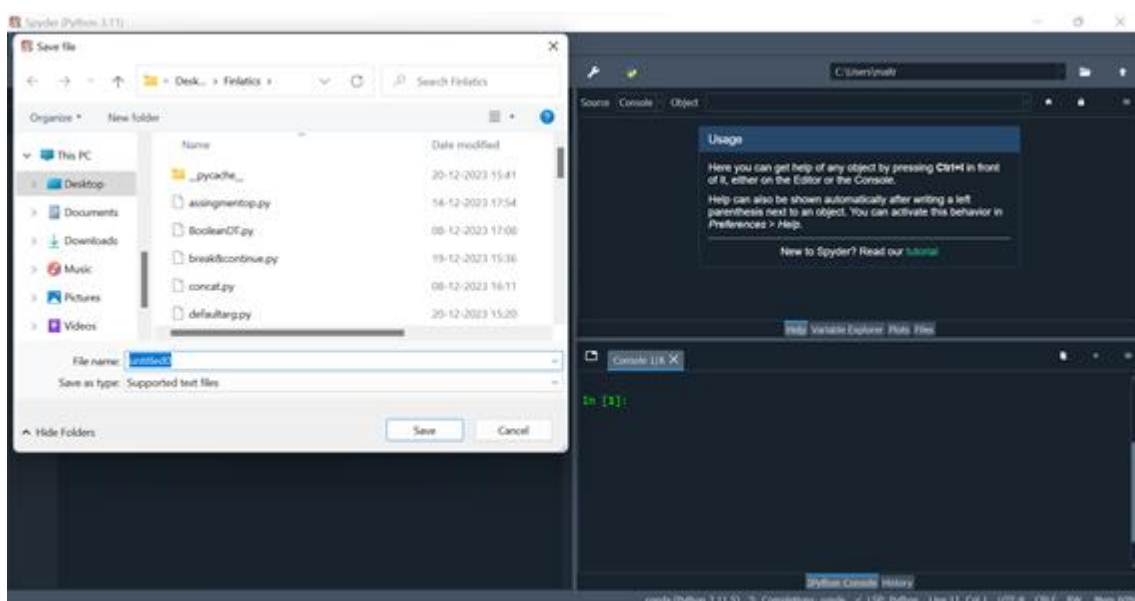


Step 4: Congratulations! You've just created a new Python file in the Spyder IDE. A blank tab should now appear in the editor area.

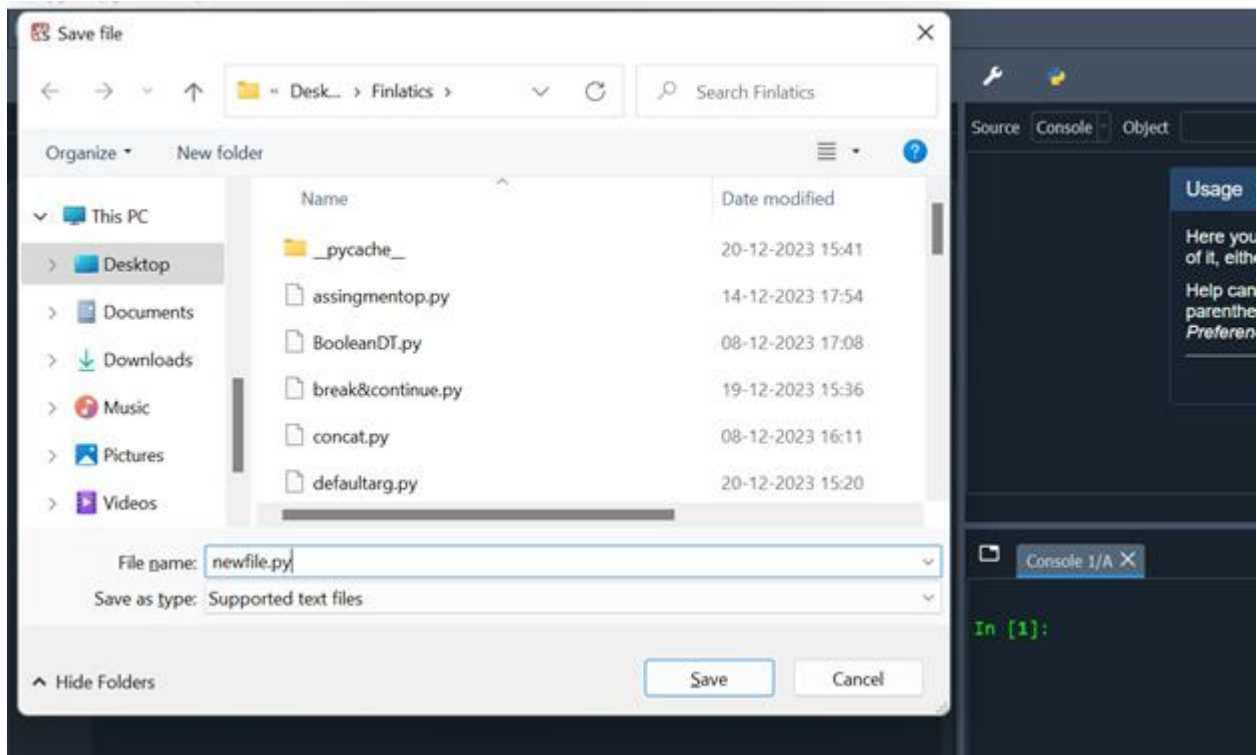


Now that you've created a new Python file in Spyder IDE, it's crucial to save your work. Saving your file not only ensures that you don't lose any progress but also allows you to easily locate and access your code later on. Let's go ahead and save the file with a meaningful name and the appropriate file extension. Step 1: Look at the menu bar at the top of the Spyder IDE window. Locate the "File" menu. Step 2: Click on the "File" menu to reveal a dropdown list of options. Step 3: From the dropdown list, click on "Save" or use the keyboard shortcut "Ctrl+S" or "Cmd+S" on Mac.

Step 4: A "Save File" dialog box will appear, allowing you to choose the location where you want to save your file.



Step 5: Browse to the desired folder location, provide a meaningful name for your file, and make sure to save it with the `.py` extension, which indicates that it's a Python file.



Now by saving your file with a meaningful name, you'll be able to easily identify and understand its purpose, even when you come back to it after a long time.

Now that you've saved your file, you can continue working on your code, knowing that your progress is safely stored.

Now, let's dive into writing your very first Python code! We'll start with a simple yet powerful function called `print()`. It's like magic that allows you to communicate with the computer and make it display messages or information.

It may sound a bit overwhelming, but don't worry! We'll talk about functions in more detail later in the course. For now, think of functions as these cool shortcuts that make your life as a programmer easier.

The `print()` function is used to show or output information on the screen. It's like telling the computer to talk or share something with you.

Let me show you how it works. Open the Python file you just created and get ready to write your first line of code.

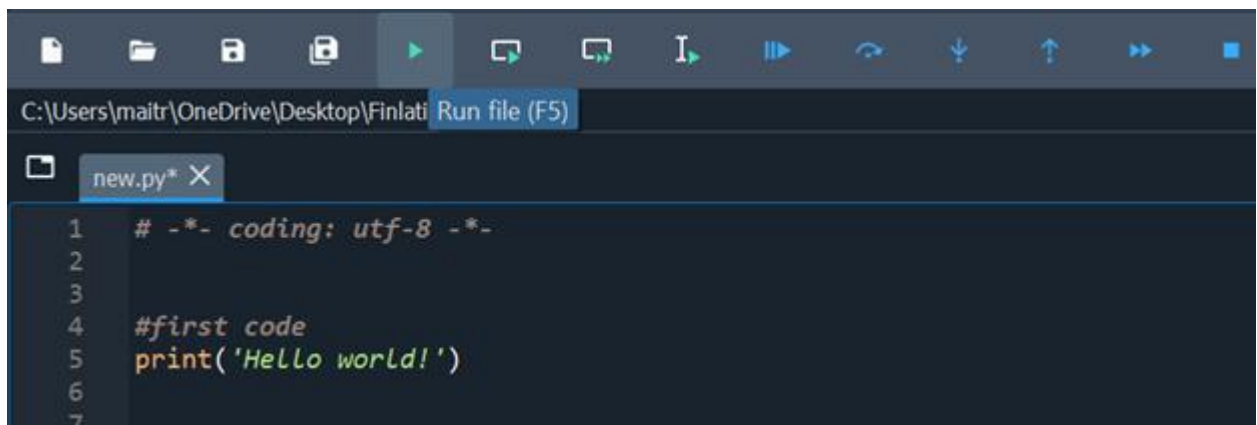
Step 1: In the file, type the following code: `print('Hello World')`.

Step 2: Great! You've just written your first Python code. This line of code tells the computer to display the text "Hello World" on the screen.

```
#first code
print('Hello world!')
```

Now that we have written our first Python code, let's see how we can run it and see the output. Running your code means telling the computer to execute the instructions you've written.

To run the code we just wrote, follow these steps: Step 1: Look at the toolbar at the top of the Spyder IDE window. Locate the "Run" button, usually represented by a green arrow.



Step 2: Once you've located the "Run" button, simply click on it.

Step 3: Congratulations! You've just run your Python code. The output of the code, which is "Hello World," should appear in the console or output window. When you run this code, the computer will execute it and show you the result. In this case, the result will be the message "Hello World" printed on the screen.

Running your code allows you to see the results of your programming efforts and verify that everything is working as expected and also helps you understand what's happening in your program.

Feel free to change the text inside the parentheses to whatever you like. The `print()` function can display not only words but also numbers, calculations, and much more!

Now, let's explore some examples using the `print()` function. Let's dive into our first example! We'll explore the code `print('Good Day')`, which displays the message "Good Day" on the screen.

Step 1: In your Python file, write the code `print('Good Day')`.

Step 2: Great! You've just written a code line that uses the `print()` function to display the text "Good Day".

```
print('Good Day')
```

When you run this code, the computer will execute it and display the output message "Good Day" on the screen.

```
Good Day
```

The text 'Good Day' is enclosed in single quotes, which tells the computer that it's a string—a sequence of characters that represents text. Strings can contain letters, numbers, and special characters. They allow us to work with text-based information in our programs.

It's important to note that strings can also be enclosed in double quotes like `print("Good Day")`..

```
#Within single quotes  
print('Good Day')  
  
#Within double quotes  
print("Good Day!")
```

We will discuss strings in-depth later in the course. We'll explore various operations and manipulations you can perform with strings to make your programs more powerful and versatile.

Let's move on to our next example, where we'll print a number using the `print()` function.

Step 1: In your Python file, type the code `print(42)`.

Step 2: Excellent! In this line of code, we're using the `print()` function to display the number 42 as output.

```
print(42)
```

Unlike strings, which are enclosed in quotes, numbers don't require any quotes when using the `print()` function.

When we don't enclose a number in quotes, Python understands it as a numerical value, not a piece of text but if a number is enclosed in quotes python will treat it as a string.

Output:

```
42
```

Numbers can be integers (whole numbers) like 42 or floating-point numbers (decimal numbers) like 3.14.

When you print a number without quotes, Python recognizes it as a numerical value and performs mathematical operations or displays it as is, depending on how you use it in your code.

So, remember, numbers are not enclosed in quotes when using the `print()` function because they are treated as numerical values by Python.

Now Let's explore another example that showcases how to concatenate or combine text using the `print()` function. We'll use the code `print('Good' + 'morning')` to create the concatenated string 'Goodmorning'.

Step 1: In your Python file, type the code `print('Good' + 'morning')`.

```
#Concatenation method  
# we use + operator for string concatenation  
print('Good' + 'Morning')
```

Step 2: Great! In this line of code, we're using the `print()` function to combine two strings, 'Good' and 'morning', using the '+' operator.

The '+' operator allows us to join or concatenate strings together, creating a new string that combines the text from both strings.

When you run the code ``print('Good' + 'morning')``, the computer will concatenate the two strings and display the result, which is 'Goodmorning', on the screen.

Output:

```
GoodMorning
```

This example demonstrates how you can use the ``print()`` function along with the `'+'` operator to create more complex and meaningful messages by combining different strings.

String concatenation can be useful in various scenarios, such as building sentences, displaying dynamic information, or creating customized messages in your programs.

So, in Python, when you use the `'+'` operator with strings, it concatenates them together to create a new string.

Let's explore another example that demonstrates how to concatenate or combine text and numbers using the ``print()`` function. We'll use the code ``print('The answer is: ', 56)`` to create a message that combines text and a number.

Step 1: In your Python file, type the code ``print('The answer is: ', 56)``.

Step 2: In this line of code, we're using the ``print()`` function to display the text 'The answer is: ' followed by the number 56.

```
#Concatenate String and Number  
#We use comma to concatenate string and a number  
print('The answer is ',56)
```

To concatenate text and a number, we can simply separate them by using a comma within the ``print()`` function.

When you run the code ``print('The answer is: ', 56)``, the computer will automatically concatenate the text and the number and display the output as 'The answer is: 56'.

Output:

```
The answer is 56
```

This example demonstrates that when using the `print()` function, you can directly pass multiple arguments separated by commas to display both text and numbers on the same line.

Python treats each argument separately and concatenates them for you, without requiring explicit conversion to a string.