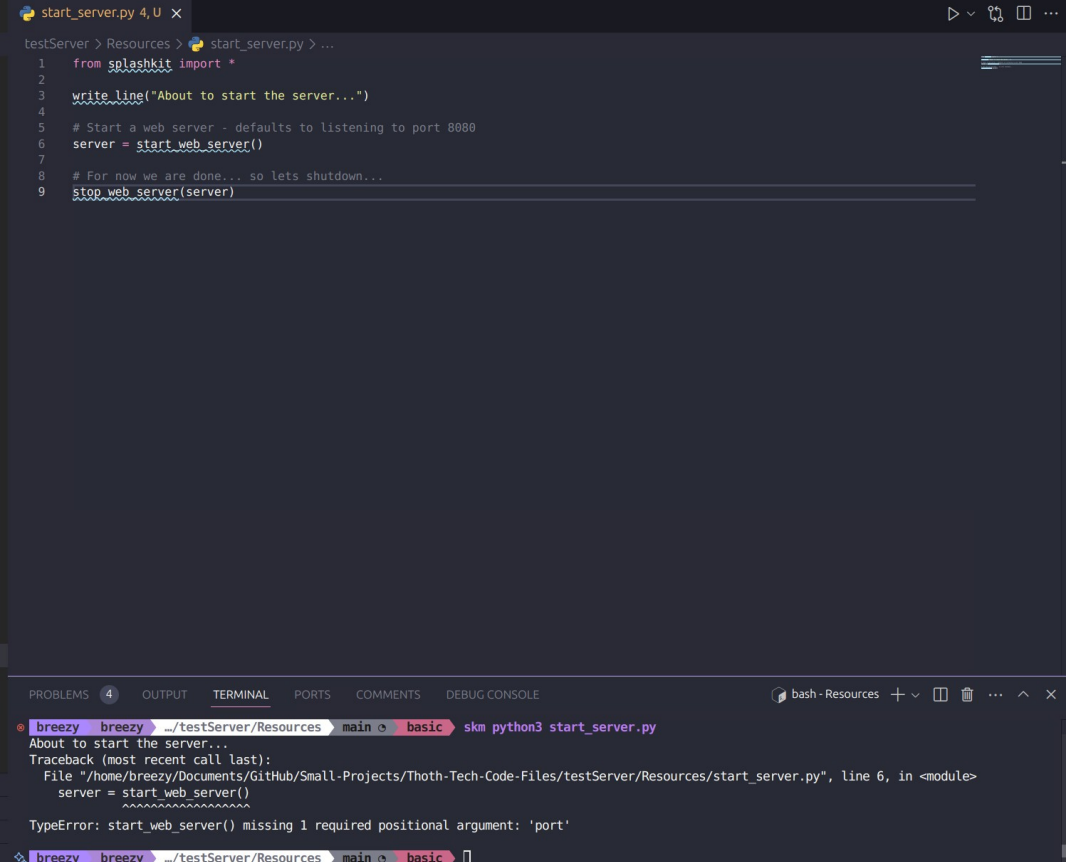


Python:

Step 2:



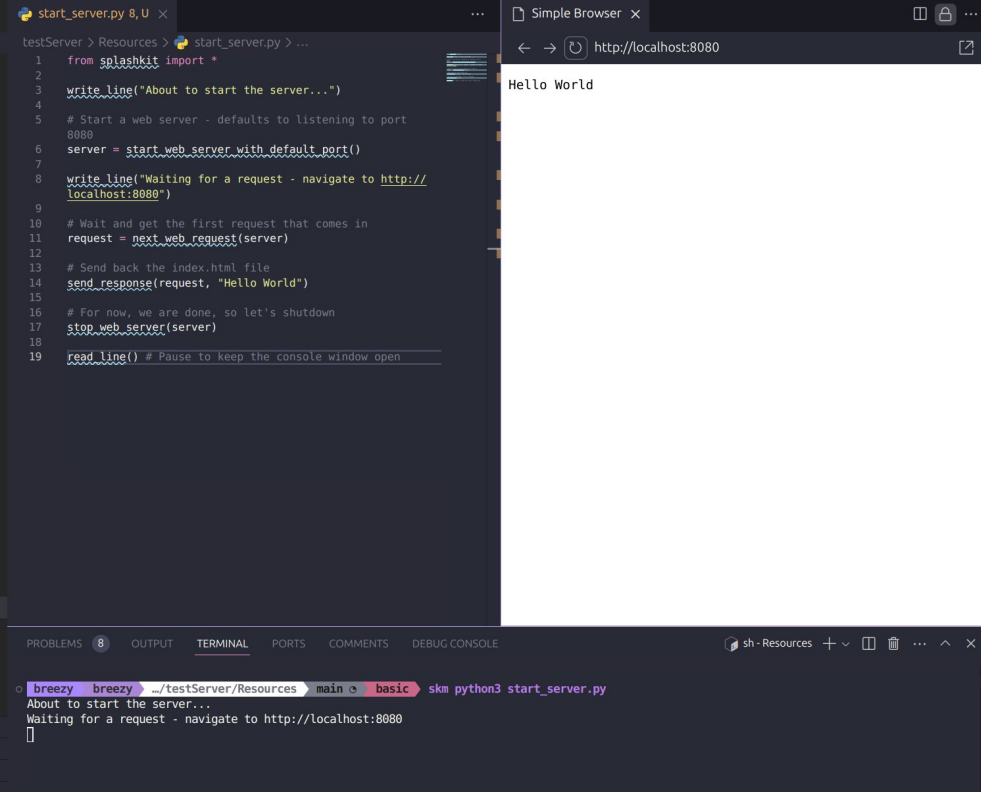
The screenshot shows a code editor with a file named `start_server.py`. The code is as follows:

```
1 from splashkit import *
2
3 write_line("About to start the server...")
4
5 # Start a web server - defaults to listening to port 8080
6 server = start_web_server()
7
8 # For now we are done... so lets shutdown...
9 stop_web_server(server)
```

The terminal window at the bottom shows the command `python3 start_server.py` being executed. The output is:

```
About to start the server...
Traceback (most recent call last):
  File "/home/breezy/Documents/GitHub/Small-Projects/Thoth-Tech-Code-Files/testServer/Resources/start_server.py", line 6, in <module>
    server = start_web_server()
             ~~~~~~^~~~~~
TypeError: start_web_server() missing 1 required positional argument: 'port'
```

Step 3:



The screenshot shows a code editor with a file named `start_server.py`. The code is as follows:

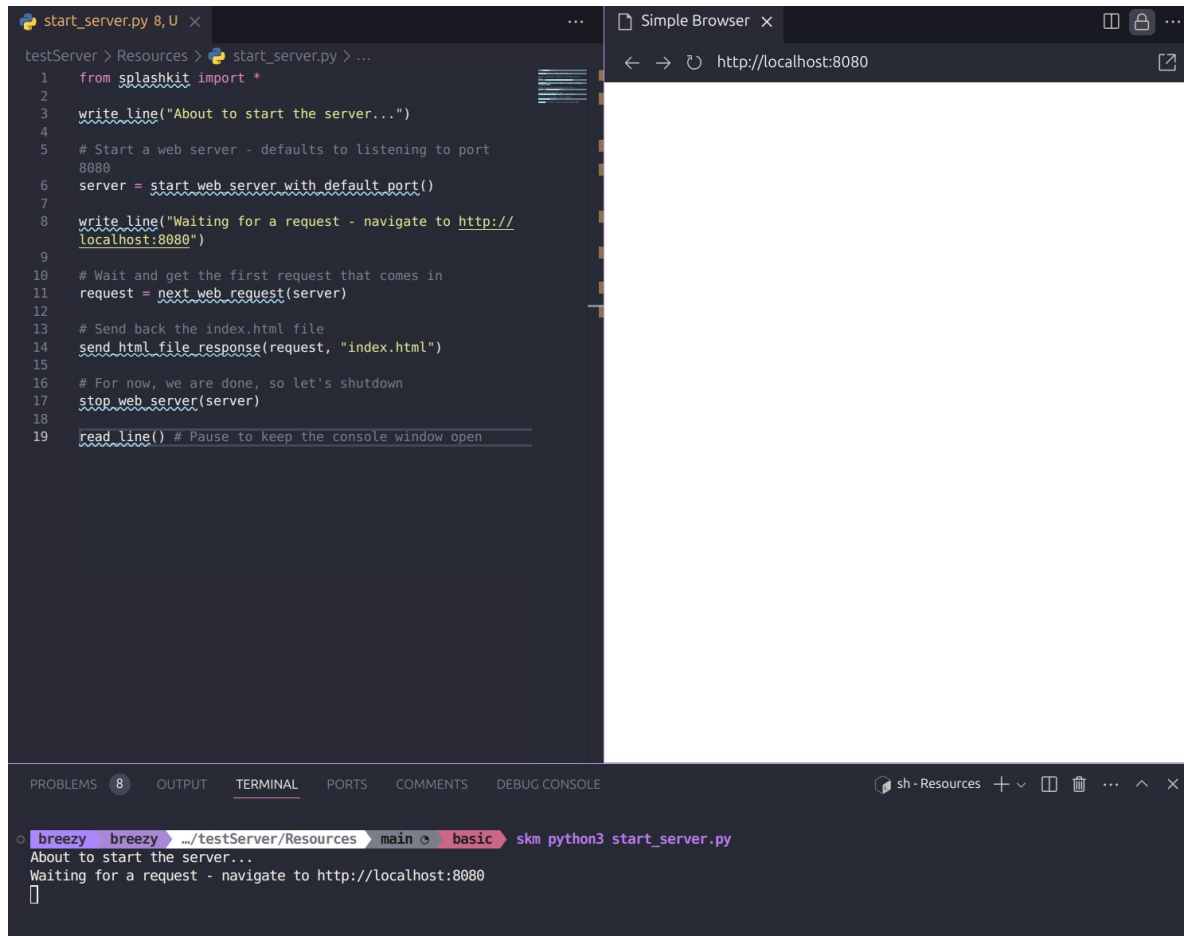
```
1 from splashkit import *
2
3 write_line("About to start the server...")
4
5 # Start a web server - defaults to listening to port
  8080
6 server = start_web_server_with_default_port()
7
8 write_line("Waiting for a request - navigate to http://
  localhost:8080")
9
10 # Wait and get the first request that comes in
11 request = next_web_request(server)
12
13 # Send back the index.html file
14 send_response(request, "Hello World")
15
16 # For now, we are done, so let's shutdown
17 stop_web_server(server)
18
19 read_line() # Pause to keep the console window open
```

The terminal window at the bottom shows the command `python3 start_server.py` being executed. The output is:

```
About to start the server...
Waiting for a request - navigate to http://localhost:8080
```

On the right, a web browser window titled "Simple Browser" is open at `http://localhost:8080`. The browser displays the text "Hello World".

Step 4:



The screenshot shows a code editor with a Python file named `start_server.py` and a Simple Browser window. The browser is displaying `http://localhost:8080`. The code in the editor is as follows:

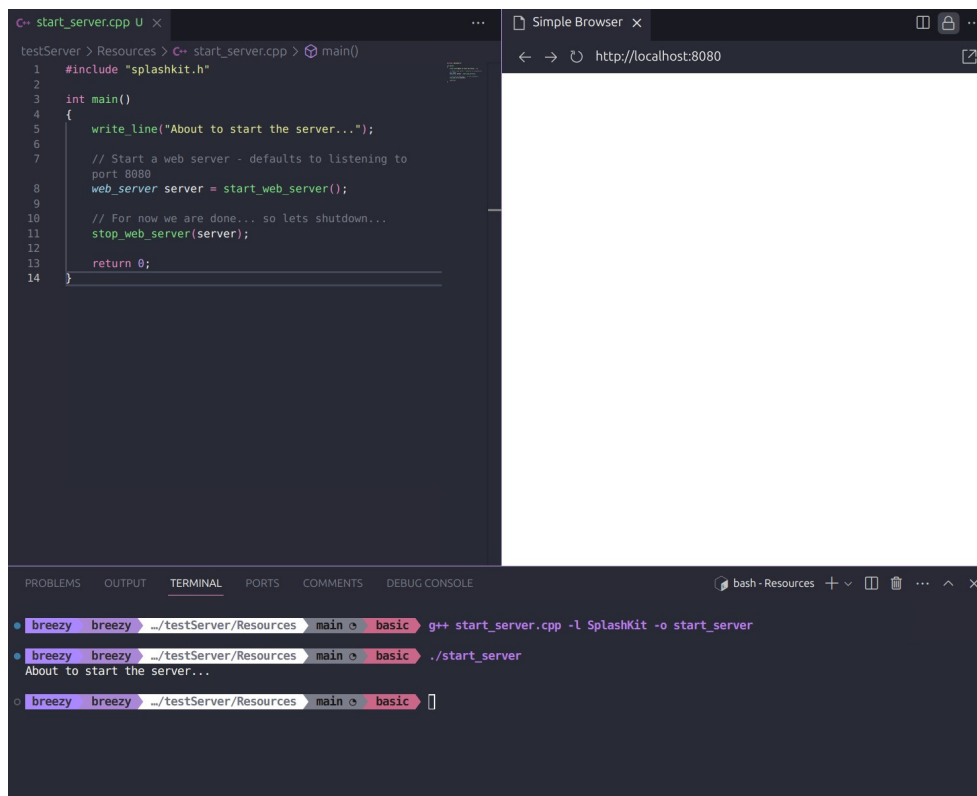
```
1 from splashkit import *
2
3 write_line("About to start the server...")
4
5 # Start a web server - defaults to listening to port
5 8080
6 server = start_web_server_with_default_port()
7
8 write_line("Waiting for a request - navigate to http://
8 localhost:8080")
9
10 # Wait and get the first request that comes in
11 request = next_web_request(server)
12
13 # Send back the index.html file
14 send_html_file_response(request, "index.html")
15
16 # For now, we are done, so let's shutdown
17 stop_web_server(server)
18
19 read_line() # Pause to keep the console window open
```

The terminal window at the bottom shows the command `python3 start_server.py` being executed, with the output:

```
About to start the server...
Waiting for a request - navigate to http://localhost:8080
[]
```

C++:

Step 2:



The screenshot shows a code editor with a C++ file named `start_server.cpp` and a terminal window. The code in the editor is as follows:

```
1 #include "splashkit.h"
2
3 int main()
4 {
5     write_line("About to start the server...");
6
7     // Start a web server - defaults to listening to
7     port 8080
8     web_server server = start_web_server();
9
10    // For now we are done... so lets shutdown...
11    stop_web_server(server);
12
13    return 0;
14 }
```

The terminal window at the bottom shows the command `g++ start_server.cpp -l SplashKit -o start_server` being executed, followed by the command `./start_server`. The output is:

```
About to start the server...
[]
```

Step 3: - Error

The screenshot shows a C++ web server implementation in a code editor. The code uses the `splashkit` library. The `main` function prints "About to start the server...", starts a web server on port 8080, prints "Waiting for a request - navigate to http://localhost:8080", waits for a request, sends back the `index.html` file, and then prints a shutdown message. The terminal shows the compilation and execution commands. The browser shows the URL `http://localhost:8080`. The terminal output shows several warning messages: "next_web_request called on an invalid server", "send_response called on an invalid request", and "stop_web_server called on an invalid server".

```
testServer > Resources > C++ start_server.cpp > main()
1 #include "splashkit.h"
2
3 int main()
4 {
5     write_line("About to start the server...");
6
7     // Start a web server - defaults to listening to
8     // port 8080
9     web_server server = web_server();
10
11    write_line("Waiting for a request - navigate to
12    http://localhost:8080");
13
14    // Wait and get the first request that comes in
15    http_request request = next_web_request(server);
16
17    // Send back the index.html file
18    send_html_file_response(request, "index.html");
19
20    // For now we are done... so lets shutdown...
21    stop_web_server(server);
22
23    return 0;
24 }
```

```
bash - Resources
• breezy breezy > ../testServer/Resources main basic g++ start_server.cpp -l SplashKit -o start_server
• breezy breezy > ../testServer/Resources main basic ./start_server
About to start the server...
Waiting for a request - navigate to http://localhost:8080
2024-08-09 08:50:51,631 WARNING [default] next_web_request called on an invalid server
2024-08-09 08:50:51,631 WARNING [default] send_response called on an invalid request
2024-08-09 08:50:51,631 WARNING [default] stop_web_server called on an invalid server
• breezy breezy > ../testServer/Resources main basic
```

Step 3: Fixed

The screenshot shows the same C++ web server implementation, but with the errors fixed. The code is identical to the previous one, but the terminal output shows that the server is now running correctly. The browser shows the URL `http://localhost:8080` and displays "Hello World".

```
testServerCPP > Resources > C++ start_server.cpp > main()
1 #include "splashkit.h"
2
3 int main()
4 {
5     write_line("About to start the server...");
6
7     // Start a web server - defaults to listening to
8     // port 8080
9     web_server server = start_web_server();
10
11    write_line("Waiting for a request - navigate to
12    http://localhost:8080");
13
14    // Wait and get the first request that comes in
15    http_request request = next_web_request(server);
16
17    // Send back the index.html file
18    send_html_file_response(request, "index.html");
19
20    // For now we are done... so lets shutdown...
21    stop_web_server(server);
22
23    return 0;
24 }
```

```
bash - Resources
• breezy breezy > ../testServerCPP/Resources main basic g++ start_server.cpp -l SplashKit -o start_server
• breezy breezy > ../testServerCPP/Resources main basic ./start_server
About to start the server...
Waiting for a request - navigate to http://localhost:8080
• breezy breezy > ../testServerCPP/Resources main basic
```

Step 4:

```
testServerCPP > Resources > C++ start_server.cpp > main()
1  #include "splashkit.h"
2
3  int main()
4  {
5      write_line("About to start the server...");
6
7      // Start a web server - defaults to listening to
8      // port 8080
9      web_server server = start_web_server();
10
11     write_line("Waiting for a request - navigate to
12     http://localhost:8080");
13
14     // Wait and get the first request that comes in
15     http_request request = next_web_request(server);
16
17     // Send back the index.html file
18     send_html_file_response(request, "index.html");
19
20     // For now we are done... so lets shutdown...
21     stop_web_server(server);
22
23     return 0;
24 }
```

Simple Browser x

http://localhost:8080

Hello World

PROBLEMS OUTPUT TERMINAL PORTS COMMENTS DEBUG CONSOLE

bash - Resources

- breezy breezy .../testServerCPP/Resources main basic g++ start_server.cpp -l SplashKit -o start_server
- breezy breezy .../testServerCPP/Resources main basic ./start_server

About to start the server...
Waiting for a request - navigate to http://localhost:8080

C#:

Step 2:

```
testServerCsharp > Resources > Server-CSharp > C# Program.cs
1  using SplashKitSDK;
2
3  Console.WriteLine("About to start the server...");
4
5  // Start a web server - defaults to listening to port
6  // 8080
7  WebServer server = SplashKit.StartWebServer();
8
9  // For now, we are done... so let's shutdown...
10 SplashKit.StopWebServer(server);
```

Simple Browser x

http://localhost:8080

PROBLEMS OUTPUT TERMINAL PORTS COMMENTS DEBUG CONSOLE

bash - Server-CSharp

- breezy breezy .../Resources/Server-CSharp main basic dotnet run build

About to start the server...

Step 3:

The screenshot shows a Visual Studio Code editor with a C# file named `Program.cs` open. The code is as follows:

```
1 using SplashKitSDK;
2
3 SplashKit.WriteLine("About to start the server...");
4
5 // Start a web server - defaults to listening to port
6 // 8080
7 WebServer server = SplashKit.StartWebServer();
8
9 SplashKit.WriteLine("Waiting for a request - navigate
10 to http://localhost:8080");
11
12 // Wait and get the first request that comes in
13 HttpRequest request = SplashKit.NextWebRequest(server);
14
15 // Send back the index.html file
16 SplashKit.SendResponse(request, "Hello World");
17
18 // For now, we are done, so let's shutdown
19 SplashKit.StopWebServer(server);
20
21 SplashKit.ReadLine(); // Pause to keep the console
22 // window open
```

Below the editor is the **TERMINAL** panel, which shows the output of the `dotnet run build` command:

```
• breezy breezy > .../Resources/Server-CSharp > main > basic dotnet run build
About to start the server...
o breezy breezy > .../Resources/Server-CSharp > main > basic dotnet run build
About to start the server...
Waiting for a request - navigate to http://localhost:8080
█
```

To the right of the editor is a **Simple Browser** window with the address `http://localhost:8080` and the content `Hello World`.

Step 4:

The screenshot shows the same Visual Studio Code editor setup as in Step 3, but with a different `Program.cs` file:

```
1 using SplashKitSDK;
2
3 SplashKit.WriteLine("About to start the server...");
4
5 // Start a web server - defaults to listening to port
6 // 8080
7 WebServer server = SplashKit.StartWebServer();
8
9 SplashKit.WriteLine("Waiting for a request - navigate
10 to http://localhost:8080");
11
12 // Wait and get the first request that comes in
13 HttpRequest request = SplashKit.NextWebRequest(server);
14
15 // Send back the index.html file
16 SplashKit.SendHtmlFileResponse(request, "index.html");
17
18 // For now, we are done, so let's shutdown
19 SplashKit.StopWebServer(server);
20
21 SplashKit.ReadLine(); // Pause to keep the console
22 // window open
```

The **TERMINAL** panel shows the same output as in Step 3, indicating the server is running and waiting for a request.

The **Simple Browser** window now displays `index.html` instead of `Hello World`.