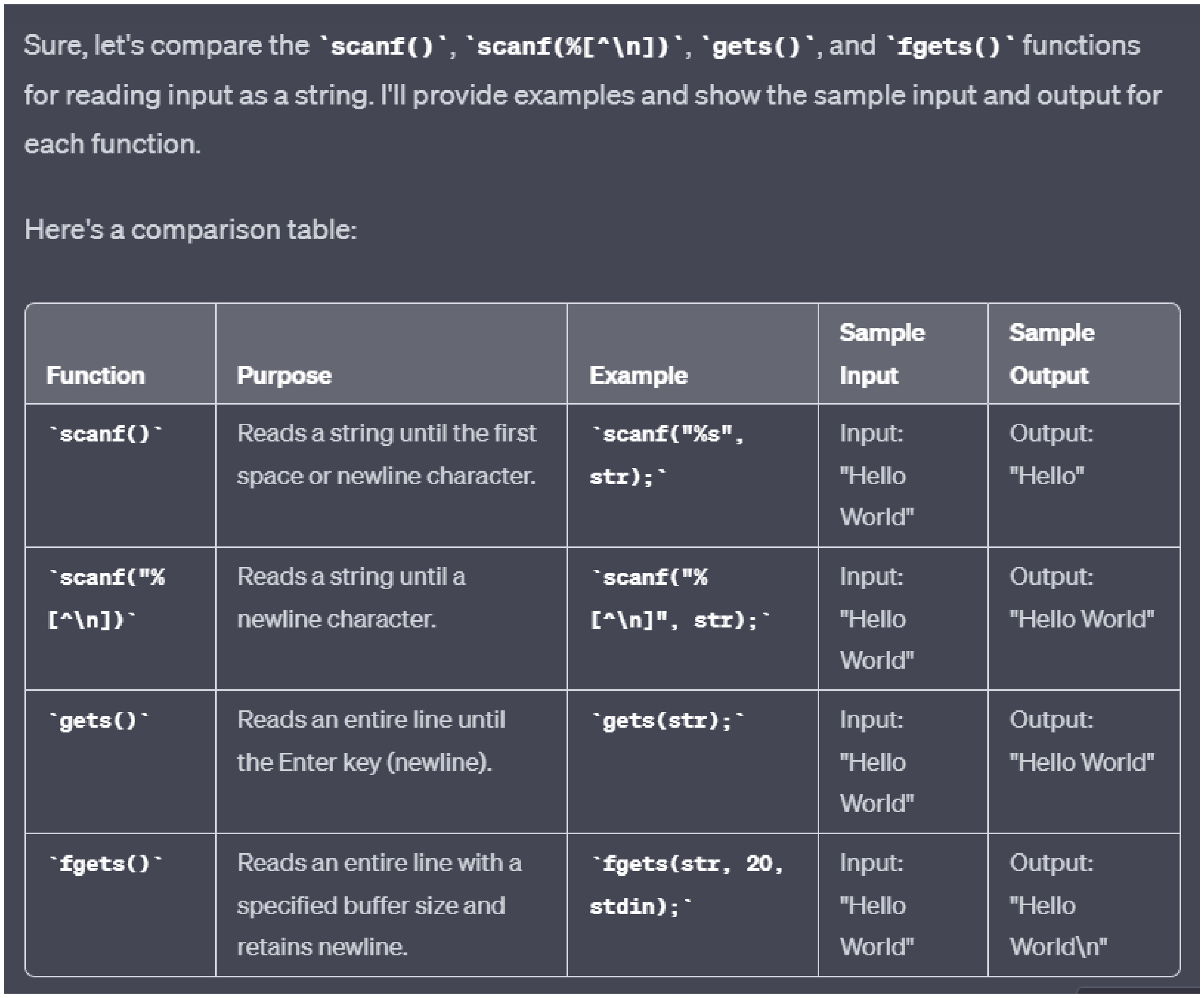


Sure, let's compare the `scanf()`, `scanf(%[^\n])`, `gets()`, and `fgets()` functions for reading input as a string. I'll provide examples and show the sample input and output for each function.

Here's a comparison table:

Function	Purpose	Example	Sample Input	Sample Output
`scanf()`	Reads a string until the first space or newline character.	`scanf("%s", str);`	Input: "Hello World"	Output: "Hello"
`scanf("%[^\n])`	Reads a string until a newline character.	`scanf("%[^\n]", str);`	Input: "Hello World"	Output: "Hello World"
`gets()`	Reads an entire line until the Enter key (newline).	`gets(str);`	Input: "Hello World"	Output: "Hello World"
`fgets()`	Reads an entire line with a specified buffer size and retains newline.	`fgets(str, 20, stdin);`	Input: "Hello World"	Output: "Hello World\n"

Now, let's provide examples for each function:

1. **`scanf()`**:

```c
#include <stdio.h>
int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);
    printf("You entered: %s\n", str);
    return 0;
}
]]>
```

```

Sample Input:
```
Hello World
```

Sample Output:
```
You entered: Hello

2. **`scanf("%[^\n])`**:

```c
#include <stdio.h>

int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%[^\n]", str);
    printf("You entered: %s\n", str);
    return 0;
}
```

Sample Input:
```
Hello World
```

Sample Output:
```
You entered: Hello World
```

3. **`gets()`**:

```c
#include <stdio.h>

int main() {
    char str[100];
    printf("Enter a string: ");
```

    gets(str);
    printf("You entered: %s\n", str);
    return 0;
}
```

Sample Input:
```
Hello World
```

Sample Output:
```
You entered: Hello World

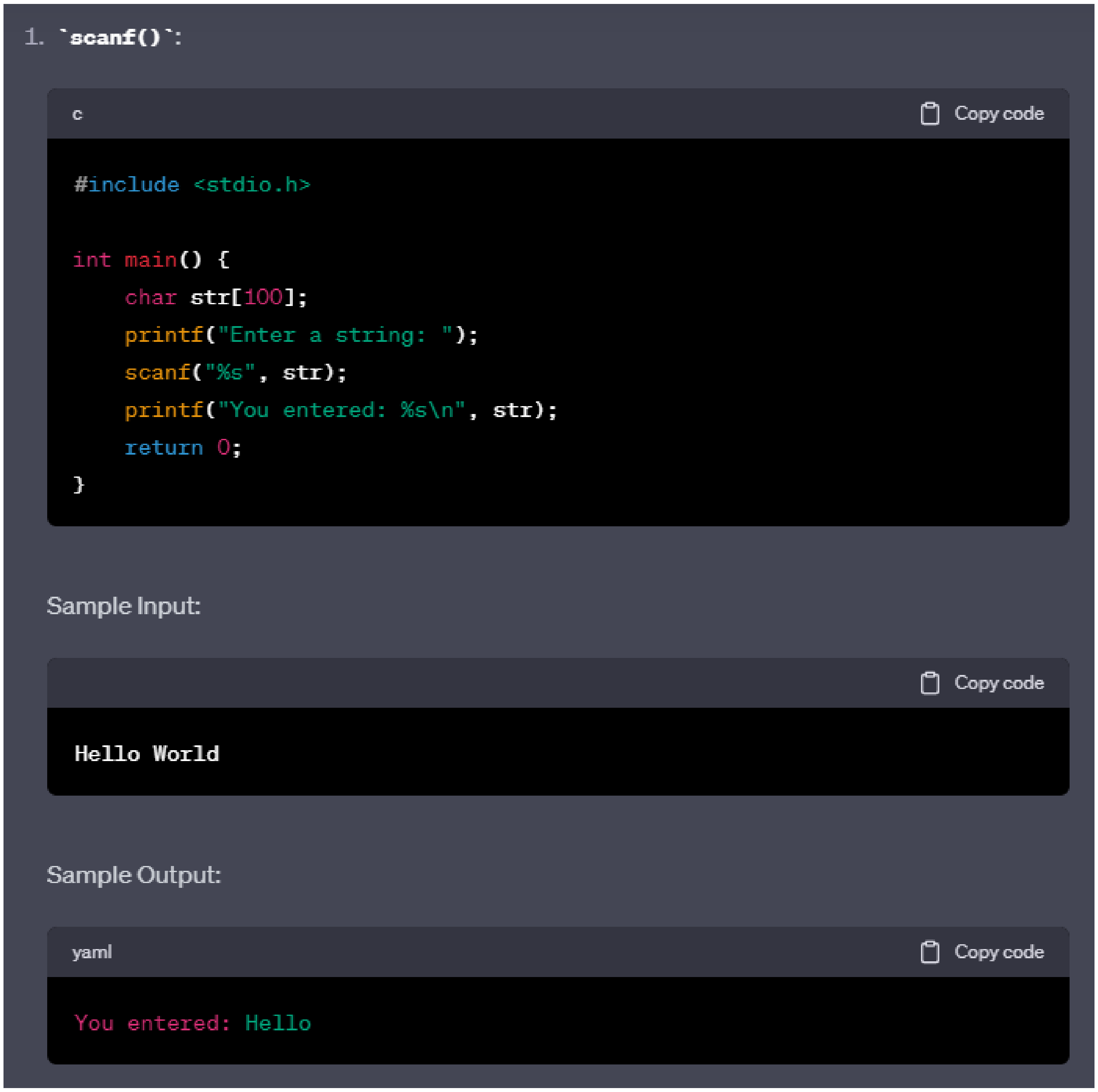
4. **`fgets()`**:

```c
#include <stdio.h>

int main() {
    char str[100];
    printf("Enter a string: ");
    fgets(str, 100, stdin);
    printf("You entered: %s", str);
    return 0;
}
```

Sample Input:
```
Hello World
```

Sample Output:
```
You entered: Hello World
```

Note that `gets()` is not recommended for use, as it doesn't limit the input size and can lead to buffer overflow vulnerabilities. Using `fgets()` is generally a safer option as it allows you to specify the maximum input size and retains newline characters when reading lines.


1. `scanf()`:

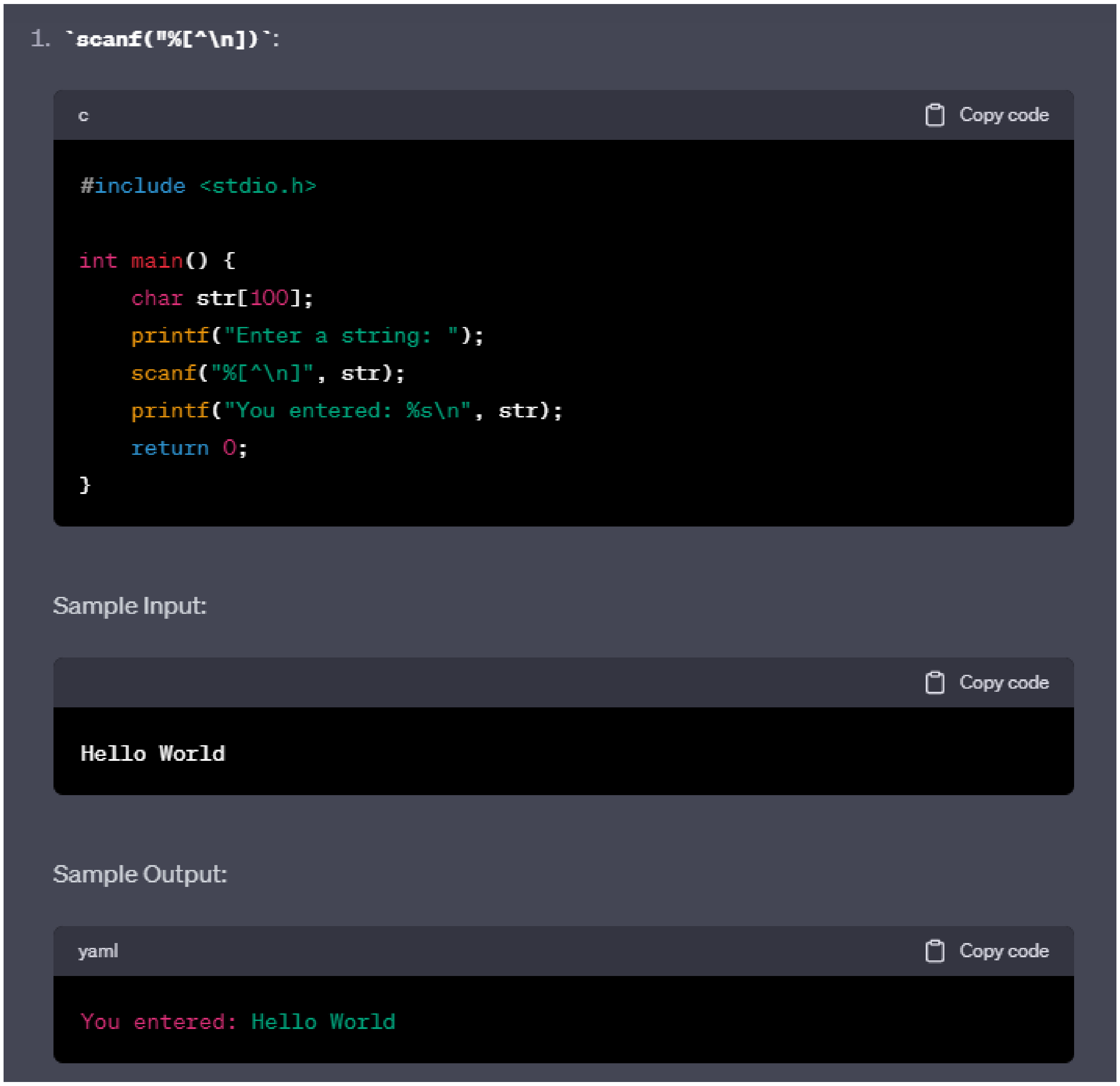
```c
#include <stdio.h>

int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);
    printf("You entered: %s\n", str);
    return 0;
}
```

Sample Input:

```
Hello World
```

Sample Output:

```yaml
You entered: Hello
```


1. `scanf("%[^\n])`:

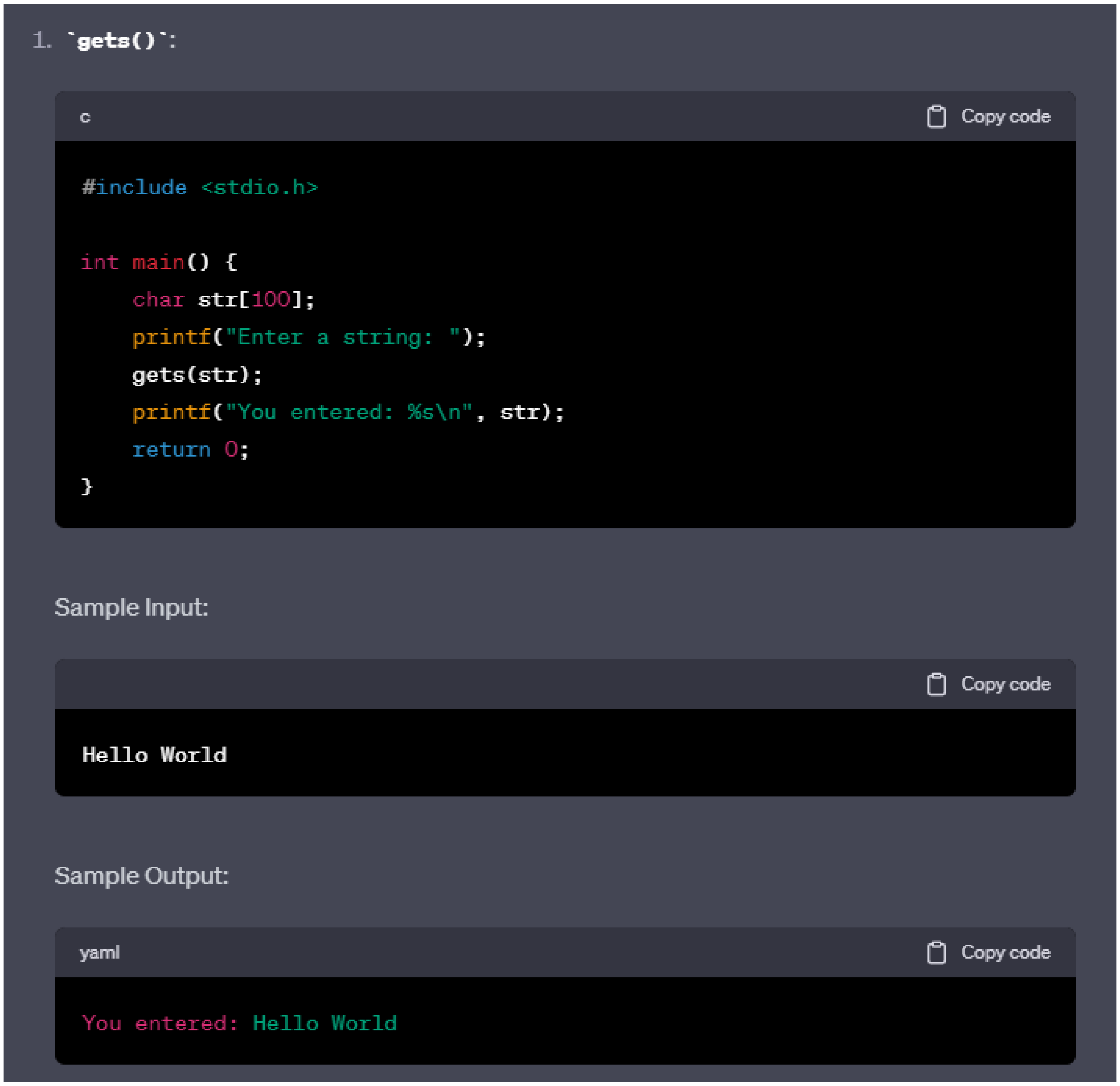
```c
#include <stdio.h>

int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%[^\n]", str);
    printf("You entered: %s\n", str);
    return 0;
}
```

Sample Input:

```
Hello World
```

Sample Output:

```yaml
You entered: Hello World
```


1. `gets()`:

c Copy code

```c
#include <stdio.h>

int main() {
    char str[100];
    printf("Enter a string: ");
    gets(str);
    printf("You entered: %s\n", str);
    return 0;
}
```

Sample Input:

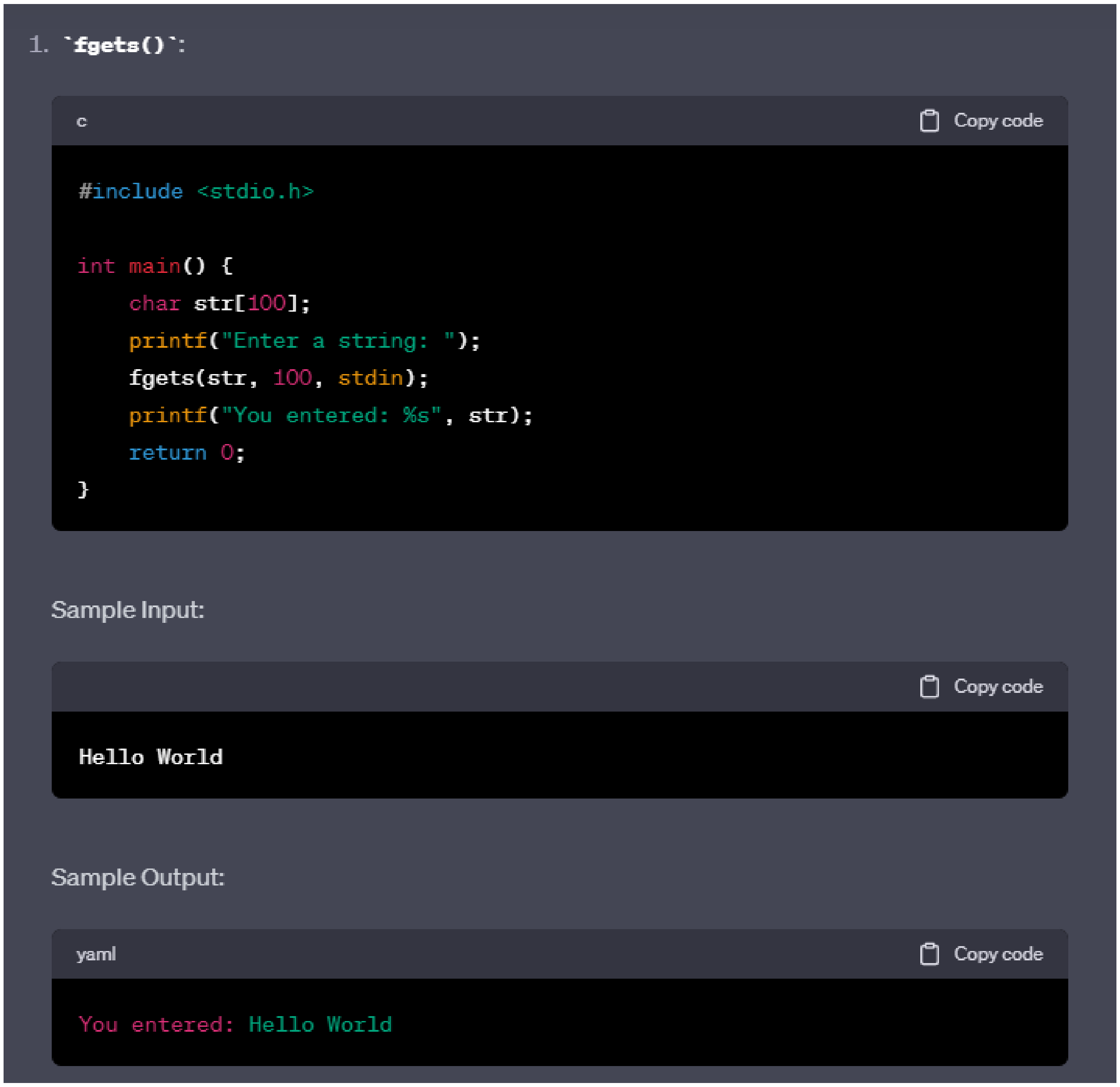
Copy code

```
Hello World
```

Sample Output:

yaml Copy code

```yaml
You entered: Hello World
```



1. `fgets()`:

c

```c
#include <stdio.h>

int main() {
    char str[100];
    printf("Enter a string: ");
    fgets(str, 100, stdin);
    printf("You entered: %s", str);
    return 0;
}
```

Sample Input:

```
Hello World
```

Sample Output:

yaml

```yaml
You entered: Hello World
```