TUPLES IN TYPESCRIPT

Student of GIAIC : Hareem Jaweid (Roll No:00423585)

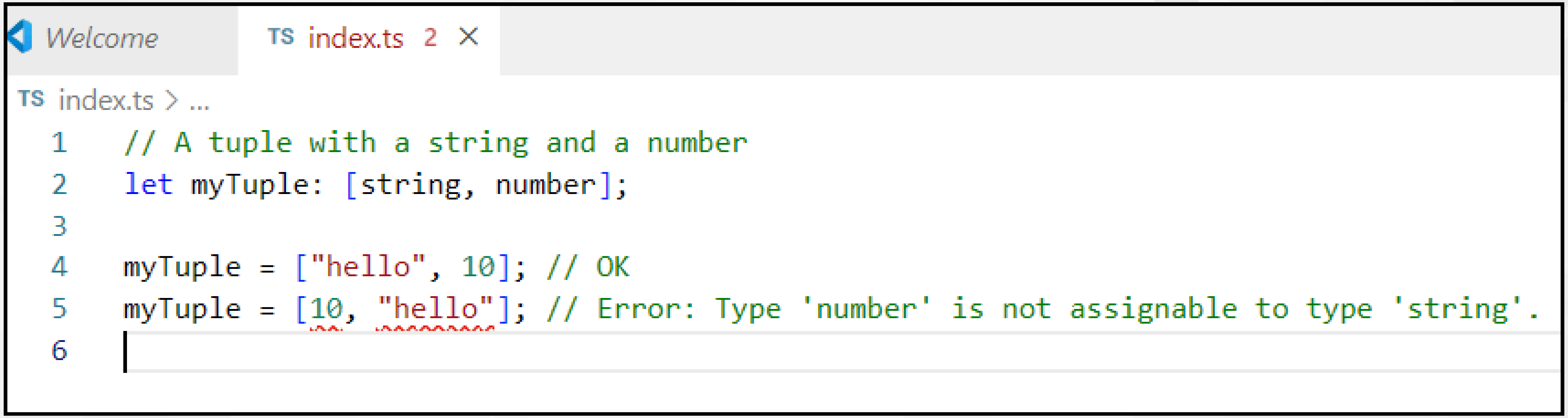
Onsite Student : Wednesday (7 -10) pm

IN TYPESCRIPT (TS), TUPLES ARE A TYPE OF ARRAY WITH A FIXED NUMBER OF ELEMENTS, WHERE EACH ELEMENT CAN HAVE A DIFFERENT TYPE. THIS IS PARTICULARLY USEFUL FOR DEFINING DATA STRUCTURES WITH A SPECIFIC SHAPE AND TYPE CONSTRAINTS. HERE'S A QUICK GUIDE ON HOW TO USE TUPLES IN TYPESCRIPT:

Student of GIAIC : Hareem Jaweid (Roll No:00423585)
Onsite Student : Wednesday (7 -10) pm

DEFINING A TUPLE

To define a tuple, you use the array syntax but specify the types of each element in the array.

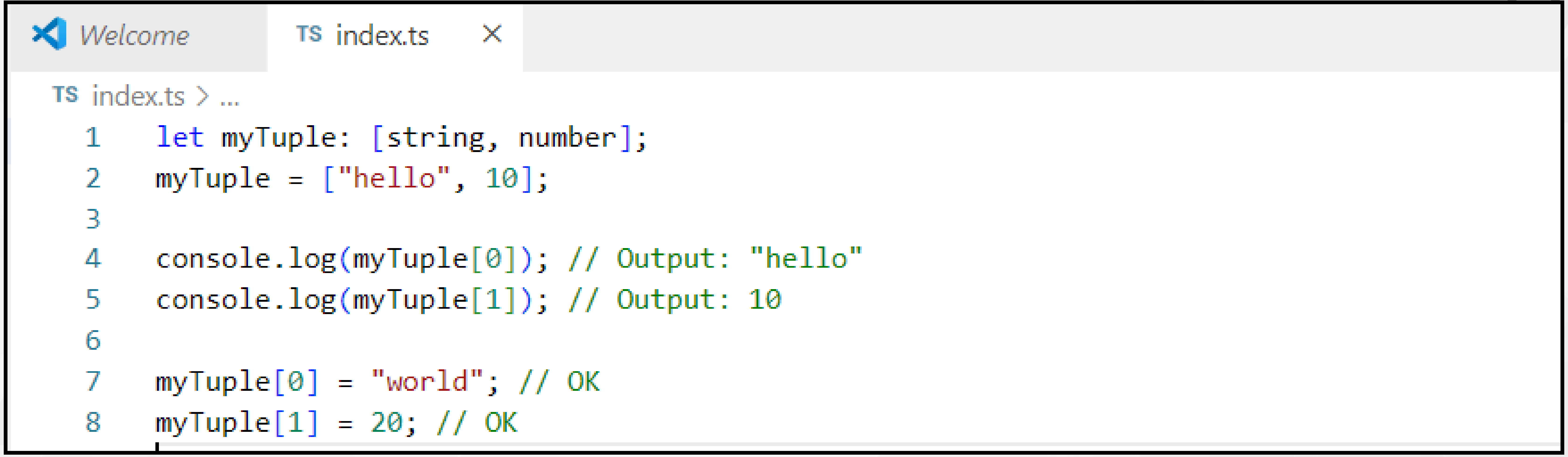


```typescript
// A tuple with a string and a number
let myTuple: [string, number];

myTuple = ["hello", 10]; // OK
myTuple = [10, "hello"]; // Error: Type 'number' is not assignable to type 'string'.

```

Student of GIAIC : Hareem Jaweid (Roll No:00423585)
Onsite Student : Wednesday (7 -10) pm

ACCESSING AND MODIFYING TUPLE ELEMENTS

You can access and modify elements in a tuple just like you would with an array.

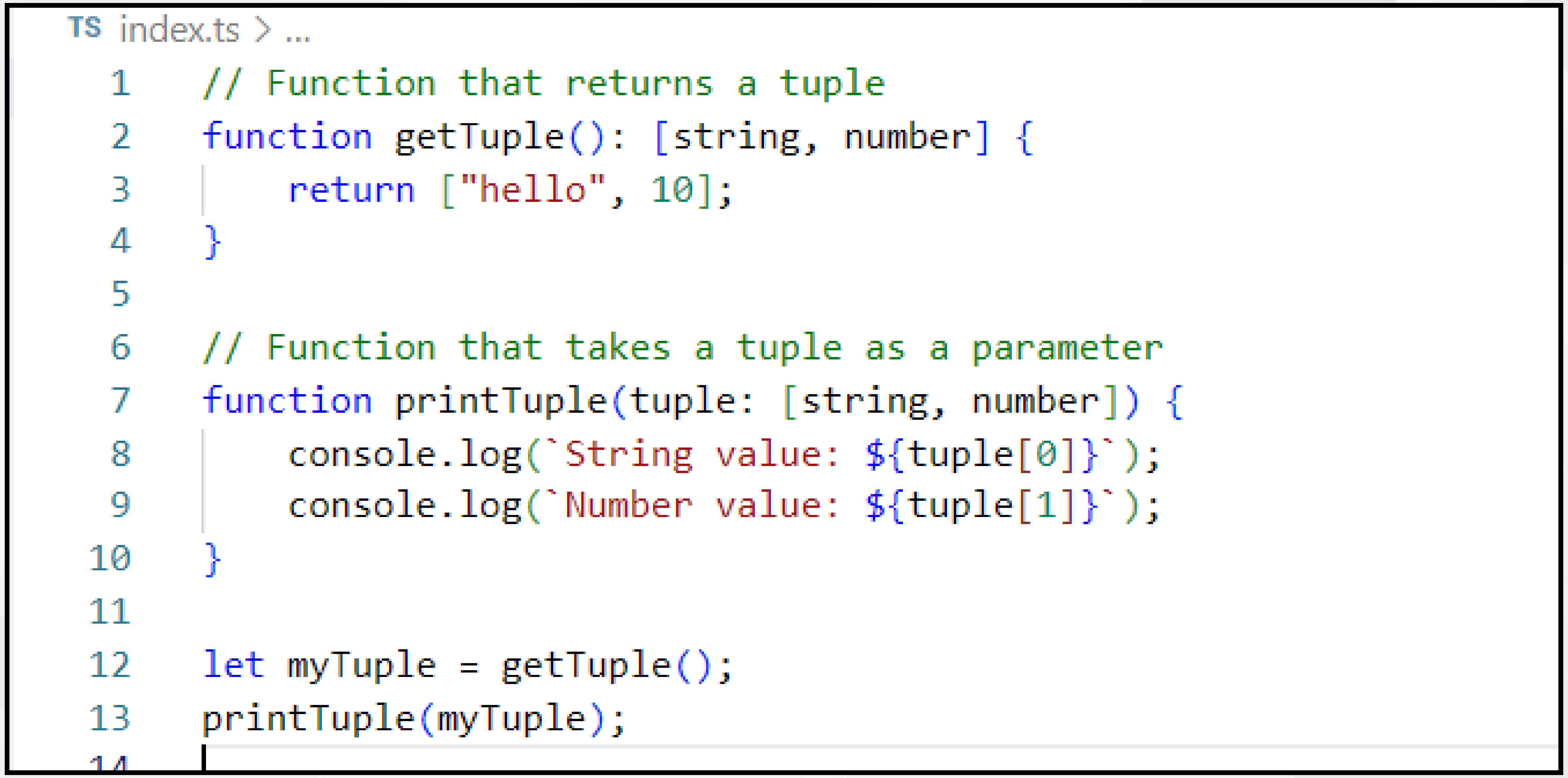


```
let myTuple: [string, number];
myTuple = ["hello", 10];

console.log(myTuple[0]); // Output: "hello"
console.log(myTuple[1]); // Output: 10

myTuple[0] = "world"; // OK
myTuple[1] = 20; // OK
```

Student of GIAIC : Hareem Jaweid (Roll No:00423585)
Onsite Student : Wednesday (7 -10) pm

USING TUPLE TYPES IN FUNCTIONS

Tuples can be used as parameter types and return types for functions.



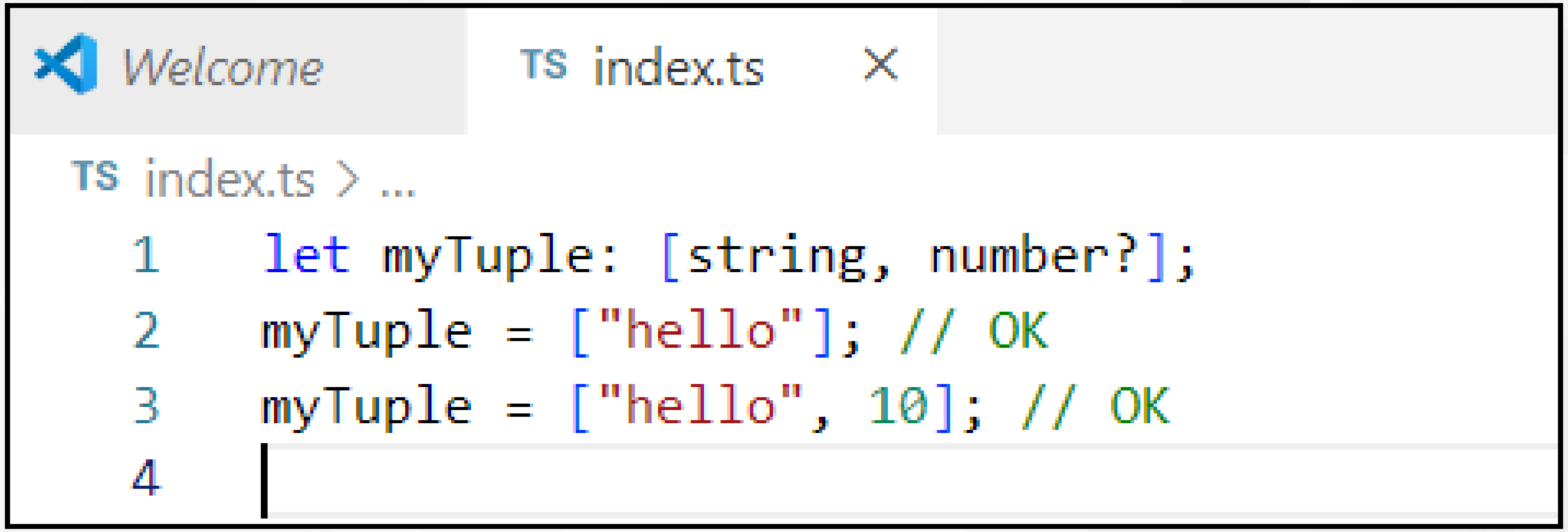
TS index.ts > ...

```typescript
// Function that returns a tuple
function getTuple(): [string, number] {
    return ["hello", 10];
}

// Function that takes a tuple as a parameter
function printTuple(tuple: [string, number]) {
    console.log(`String value: ${tuple[0]}`);
    console.log(`Number value: ${tuple[1]}`);
}

let myTuple = getTuple();
printTuple(myTuple);
```

Student of GIAIC : Hareem Jaweid (Roll No:00423585)
Onsite Student : Wednesday (7 -10) pm

OPTIONAL ELEMENTS

You can define optional elements in a tuple using the question mark ? syntax.

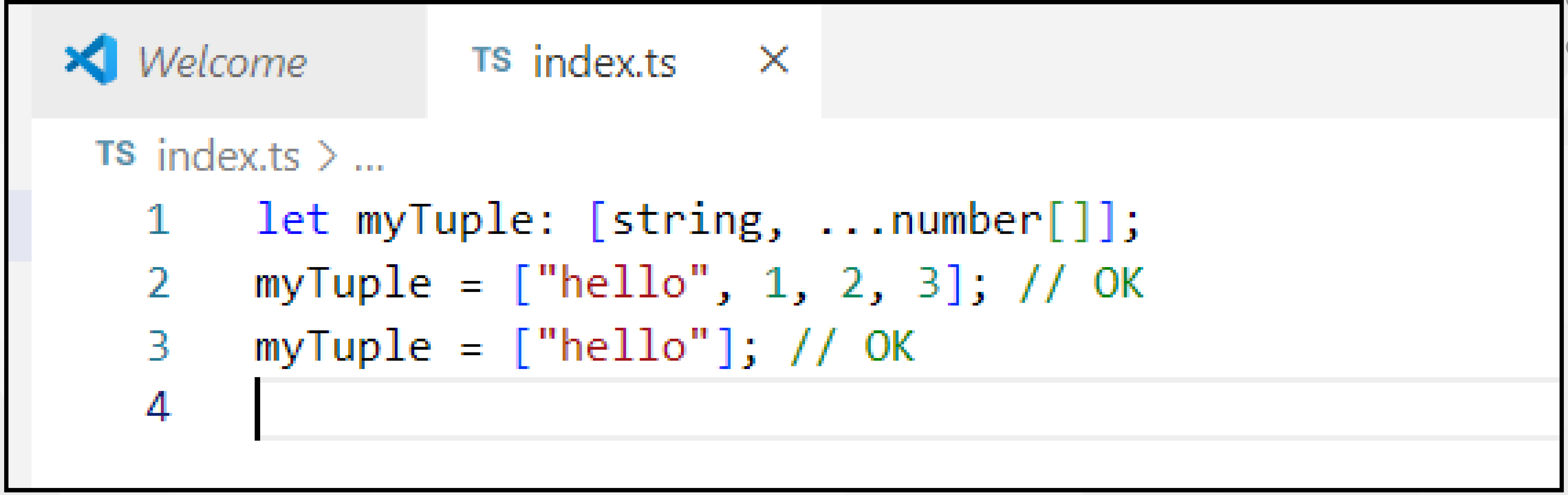


```typescript
let myTuple: [string, number?];
myTuple = ["hello"]; // OK
myTuple = ["hello", 10]; // OK

```

Student of GIAIC : Hareem Jaweid (Roll No:00423585)
Onsite Student : Wednesday (7 -10) pm

REST ELEMENTS

You can use the rest syntax to define tuples with a variable number of elements of a specific type.

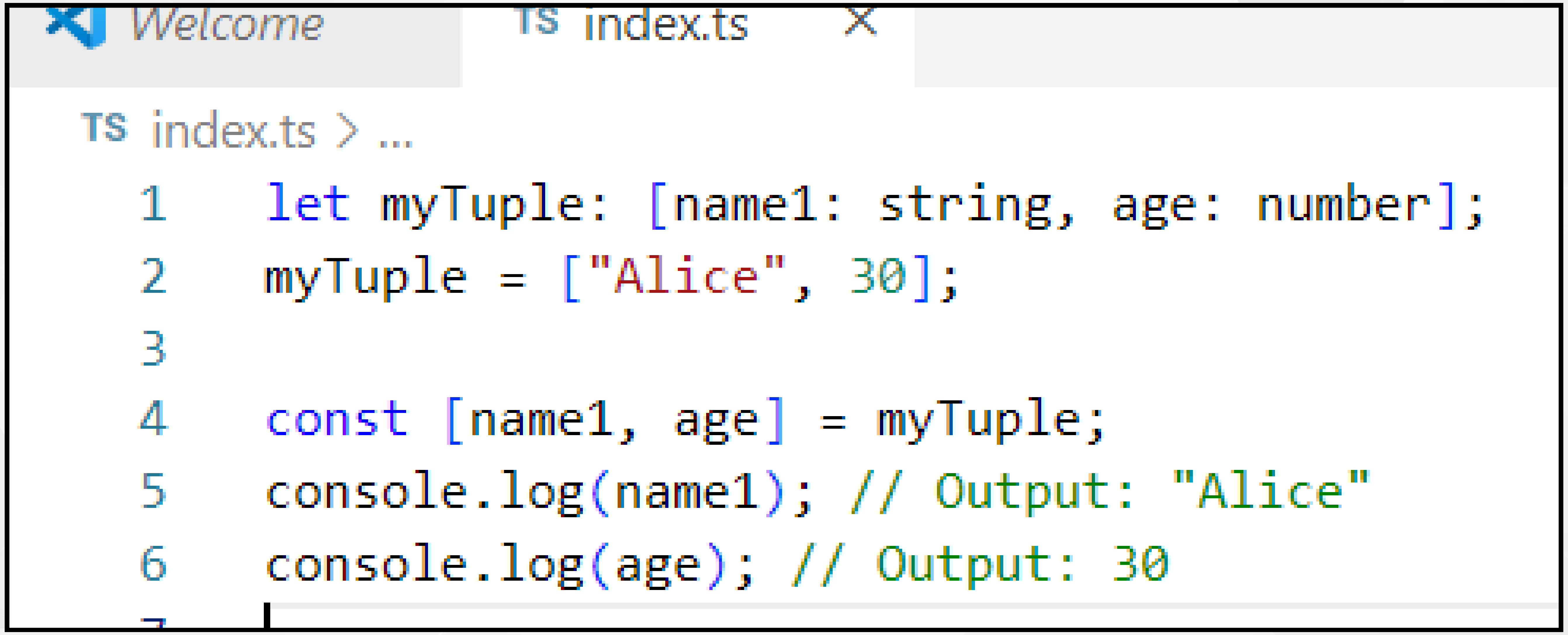


```
let myTuple: [string, ...number[]];
myTuple = ["hello", 1, 2, 3]; // OK
myTuple = ["hello"]; // OK
```

Student of GIAIC : Hareem Jaweid (Roll No:00423585)
Onsite Student : Wednesday (7 -10) pm

NAMED TUPLES (WITH LABELLED TUPLE ELEMENTS)

As of TypeScript 4.0, you can add labels to tuple elements to make the code more readable.

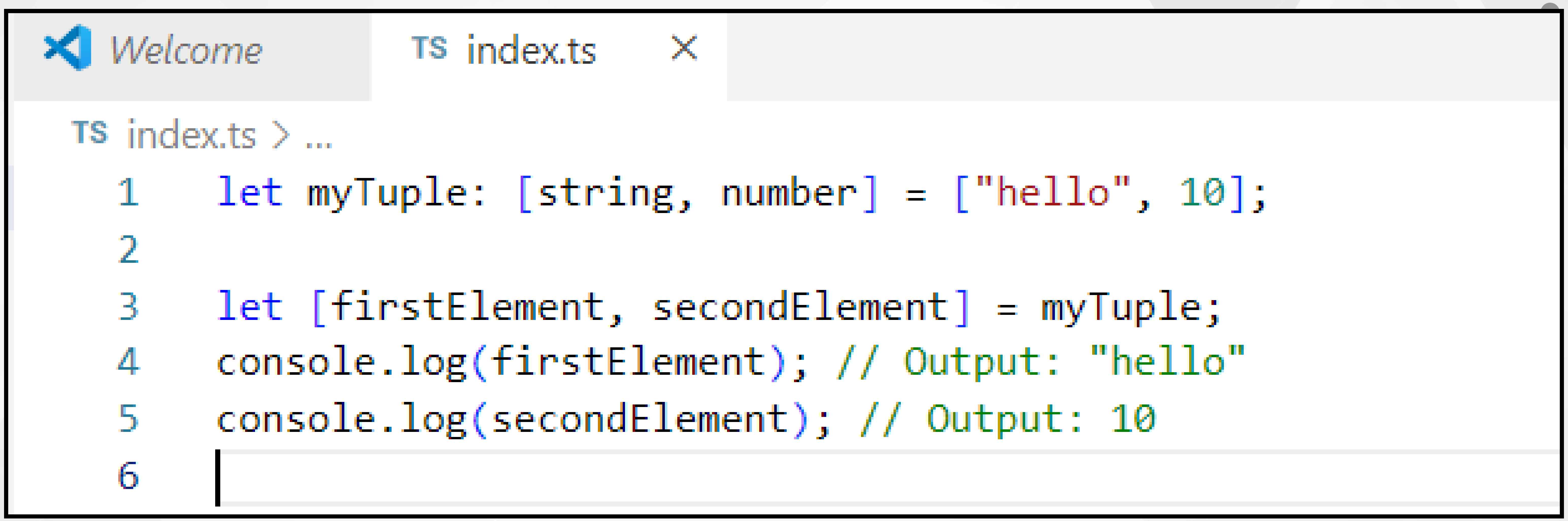


```
let myTuple: [name1: string, age: number];
myTuple = ["Alice", 30];

const [name1, age] = myTuple;
console.log(name1); // Output: "Alice"
console.log(age); // Output: 30
```

Student of GIAIC : Hareem Jaweid (Roll No:00423585)
Onsite Student : Wednesday (7 -10) pm

DESTRUCTURING TUPLES

Tuples can be destructured just like arrays.



```typescript
let myTuple: [string, number] = ["hello", 10];

let [firstElement, secondElement] = myTuple;
console.log(firstElement); // Output: "hello"
console.log(secondElement); // Output: 10

```

Student of GIAIC : Hareem Jaweid (Roll No:00423585)
Onsite Student : Wednesday (7 -10) pm

USING TUPLES EFFECTIVELY IN TYPESCRIPT HELPS IN SCENARIOS WHERE THE DATA STRUCTURE IS FIXED AND TYPES OF ELEMENTS ARE KNOWN IN ADVANCE. THIS ENHANCES TYPE SAFETY AND CLARITY IN YOUR CODE.

Student of GIAIC : Hareem Jaweid (Roll No:00423585)
Onsite Student : Wednesday (7 -10) pm

THANK YOU

Student of GIAIC : Hareem Jaweid (Roll No:00423585)
Onsite Student : Wednesday (7 -10) pm