PHP - Return Type Declarations

PHP version 7 extends the scalar type declaration feature to the return value of a function also. As per this new provision, the return type declaration specifies the type of value that a function should return. We can declare the following types for return types –

- int
- float
- bool
- string
- interfaces
- array
- callable

To implement the return type declaration, a function is defined as –

```
function myfunction(type $par1, type $param2): type {
   # function body
   return $val;
}
```

PHP parser is coercive typing by default. You need to declare "strict_types=1" to enforce stricter verification of the type of variable to be returned with the type used in the definition.

Example

In the following example, the division() function is defined with a return type as int.

</> Open Compiler

```
<?php
   function division(int $x, int $y): int {
      $z = $x/$y;
      return $z;
   }
```

```
    $x=20.5;
    $y=10;

    echo "First number: " . $x;
    echo "\nSecond number: " . $y;
    echo "\nDivision: " . division($x, $y);
?>
```

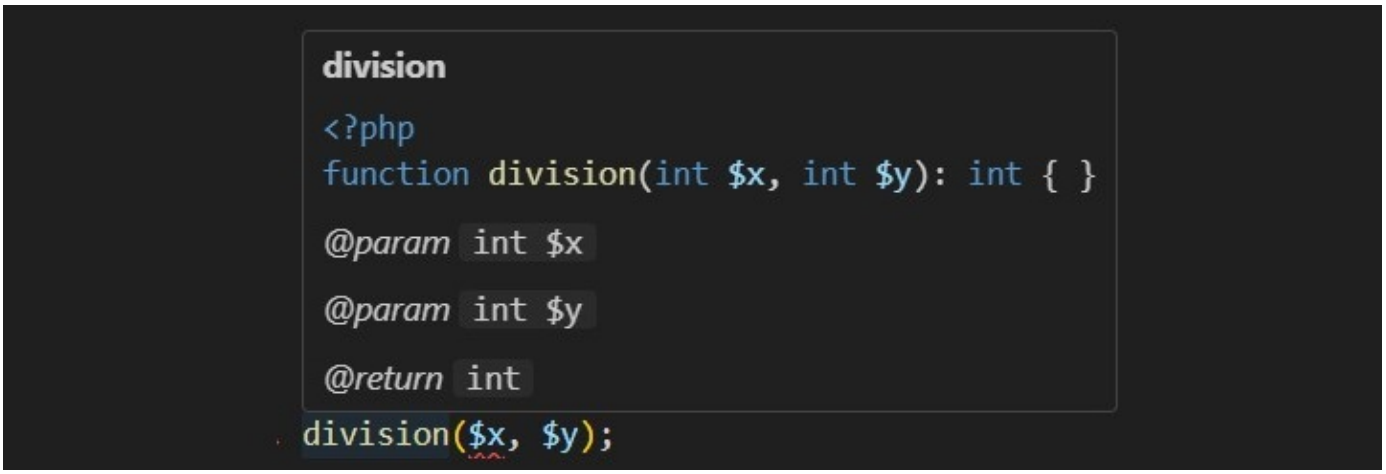
Since the type checking has not been set to strict_types=1, the division take place even if one of the parameters is a non-integer.

```
First number: 20.5
Second number: 10
Division: 2
```

However, as soon as you add the declaration of strict_types at the top of the script, the program raises a fatal error message.

```
Fatal error: Uncaught TypeError: division(): Argument #1 ($x) must be of type int,
Stack trace:
#0 div.php(12): division(20.5, 10)
#1 {main}
   thrown in div.php on line 3
```

VS Code warns about the error even before running the code by displaying error lines at the position of error –



Example

To make the division() function return a float instead of int, cast the numerator to float, and see how PHP raises the fatal error –

</> Open Compiler

```php
<?php
   // declare(strict_types=1);
   function division(int $x, int $y): int {
      $z = (float)$x/$y;
      return $z;
   }

   $x=20;
   $y=10;

   echo "First number: " . $x;
   echo "\nSecond number: " . $y;
   echo "\nDivision: " . division($x, $y);
?>
```

Uncomment the **declare** statement at the top and run this code here to check its output. It will show an error –

```
First number: 20
Second number: 10PHP Fatal error:  Uncaught TypeError: division(): Return value must b
Stack trace:
#0 /home/cg/root/14246/main.php(13): division()
#1 {main}
  thrown in /home/cg/root/14246/main.php on line 5
```