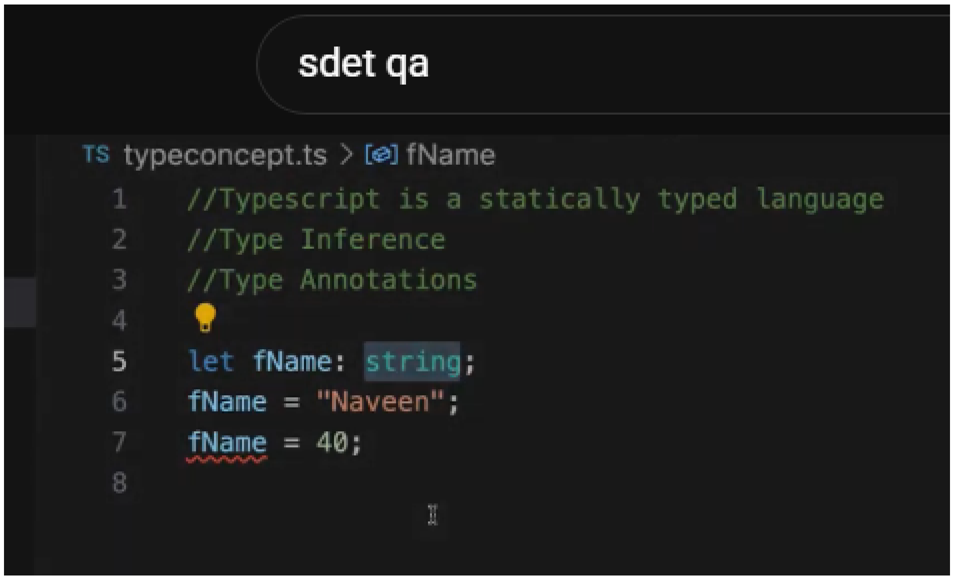
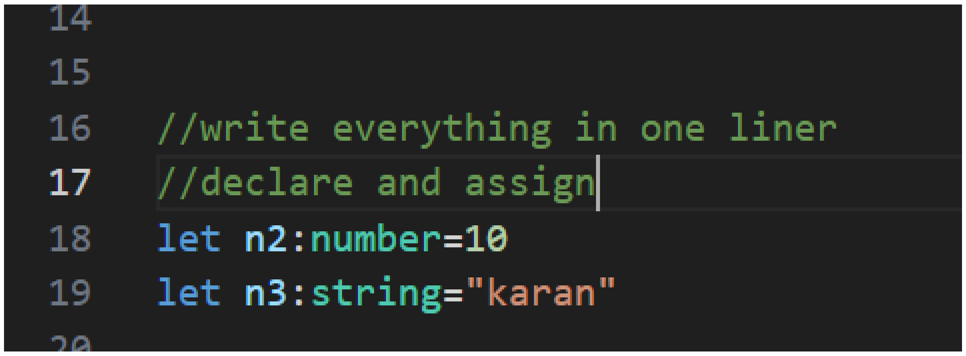
When we try to give different data type then compile error-



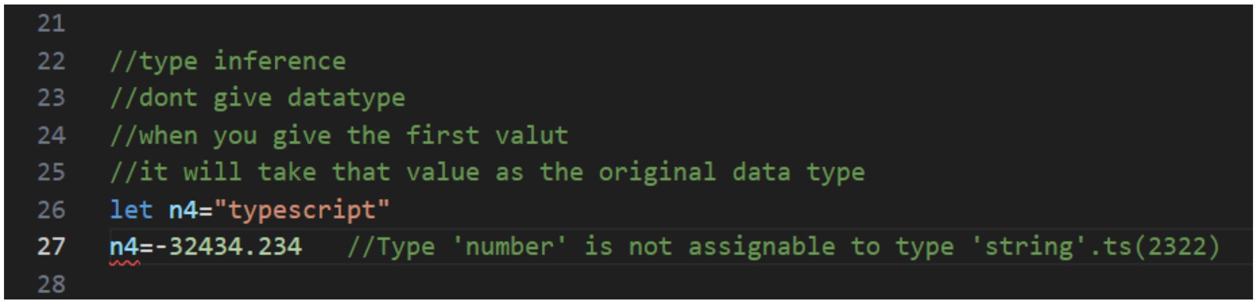
```
//Typescript is a statically typed language
//Type Inference
//Type Annotations

let fName: string;
fName = "Naveen";
fName = 40;
```

Assign in same line -

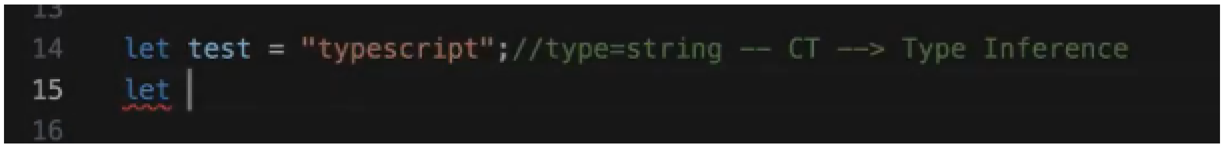


```


//write everything in one liner
//declare and assign
let n2:number=10
let n3:string="karan"
```

Even if you dont give data type then no error-

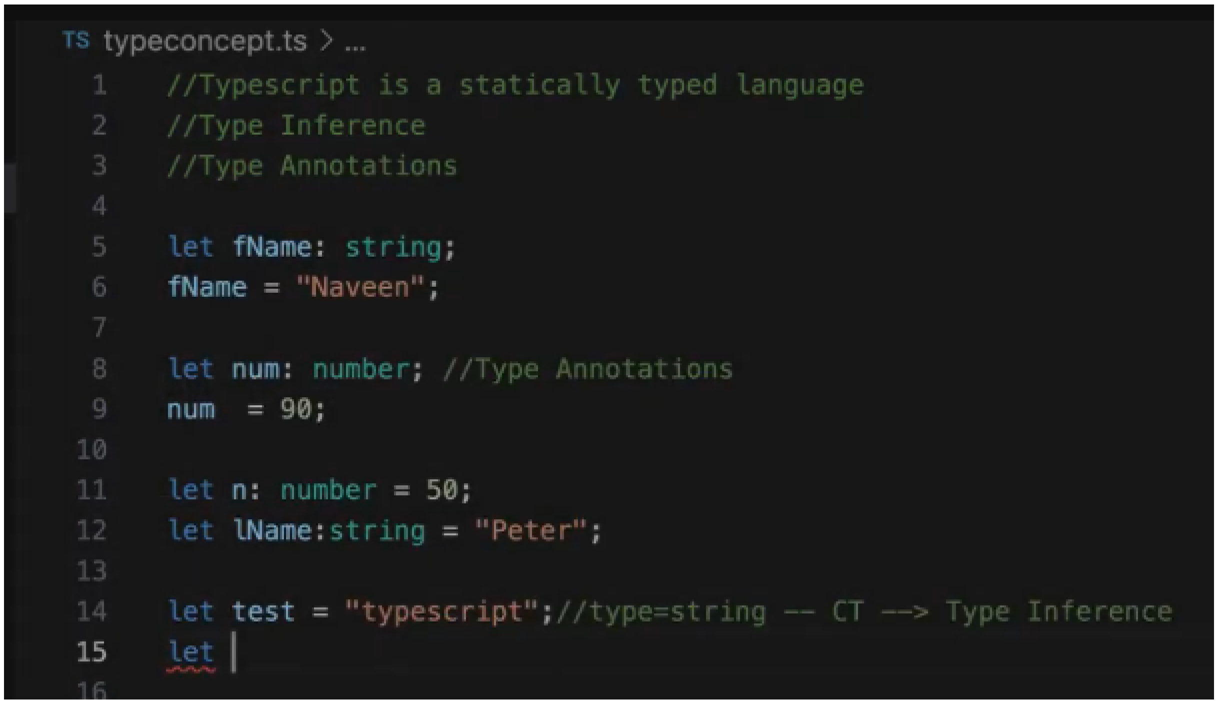


```

//type inference
//dont give datatype
//when you give the first valut
//it will take that value as the original data type
let n4="typescript"
n4=-32434.234   //Type 'number' is not assignable to type 'string'.ts(2322)

```



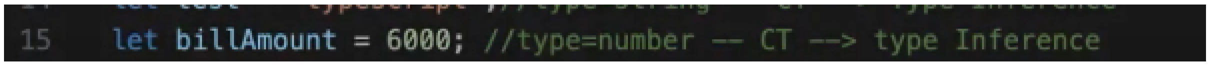
```
let test = "typescript";//type=string -- CT --> Type Inference
let 

```

Type inference versus type annotation –
In type annotation we explicitly mention data type.
In inference we dont have to give.

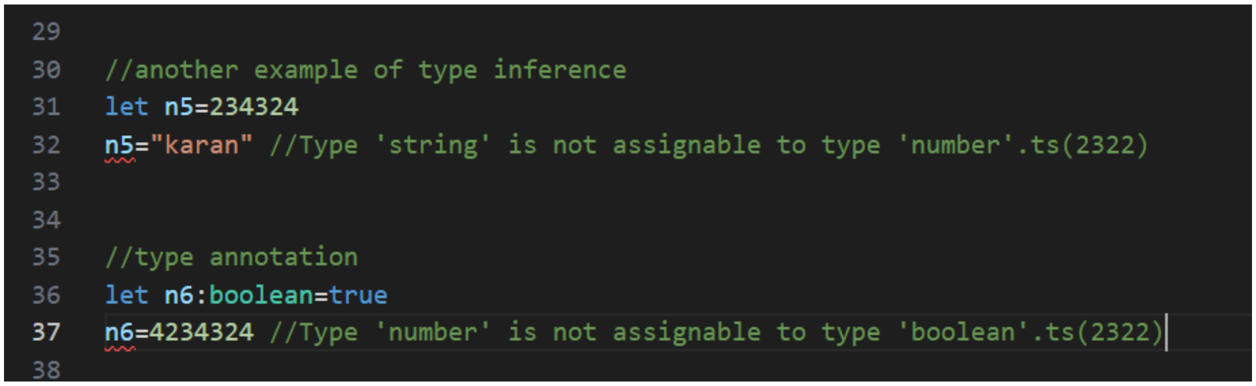


typeconcept.ts > ...

```
//Typescript is a statically typed language
//Type Inference
//Type Annotations

let fName: string;
fName = "Naveen";

let num: number; //Type Annotations
num  = 90;

let n: number = 50;
let lName:string = "Peter";

let test = "typescript";//type=string -- CT --> Type Inference
let 
```

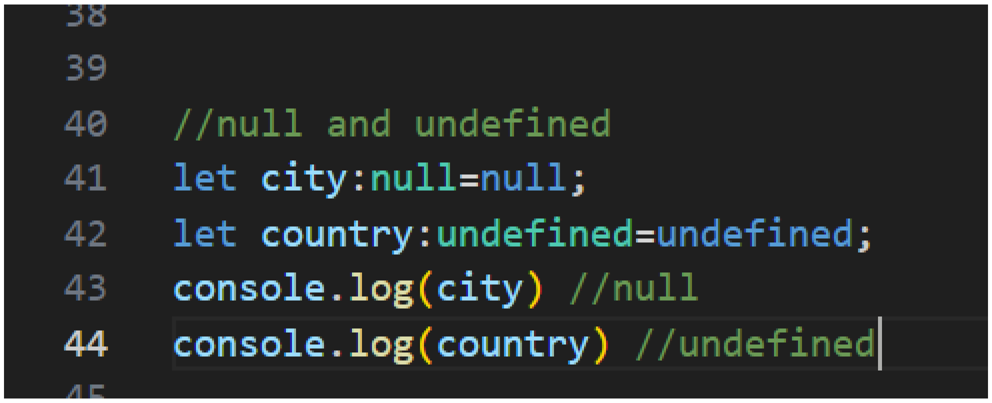


```
let billAmount = 6000; //type=number -- CT --> type Inference
```

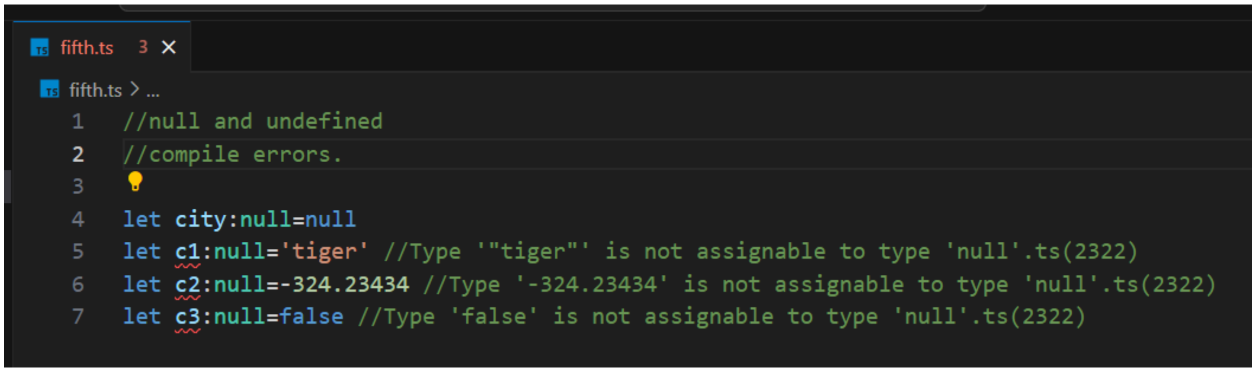


```

//another example of type inference
let n5=234324
n5="karan" //Type 'string' is not assignable to type 'number'.ts(2322)


//type annotation
let n6:boolean=true
n6=4234324 //Type 'number' is not assignable to type 'boolean'.ts(2322)
```

Define null and undefined type –

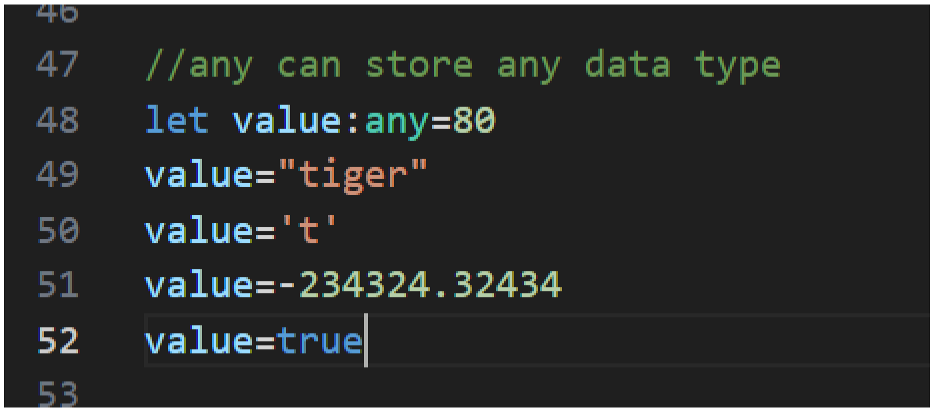


```
//null and undefined
let city:null=null;
let country:undefined=undefined;
console.log(city) //null
console.log(country) //undefined
```



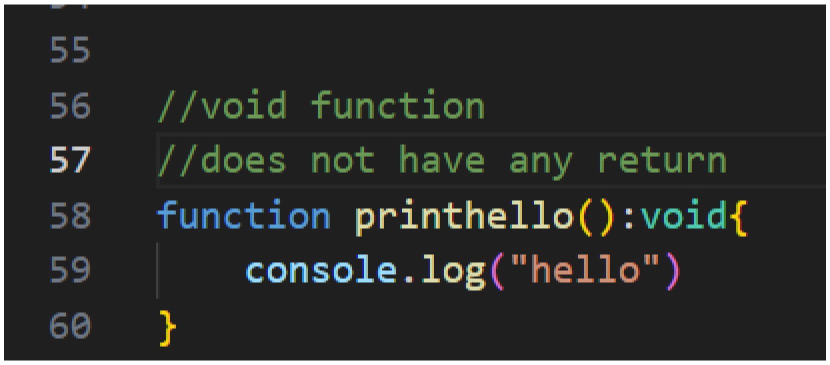
```
//null and undefined
//compile errors.

let city:null=null
let c1:null='tiger' //Type '"tiger"' is not assignable to type 'null'.ts(2322)
let c2:null=-324.23434 //Type '-324.23434' is not assignable to type 'null'.ts(2322)
let c3:null=false //Type 'false' is not assignable to type 'null'.ts(2322)
```

Any -

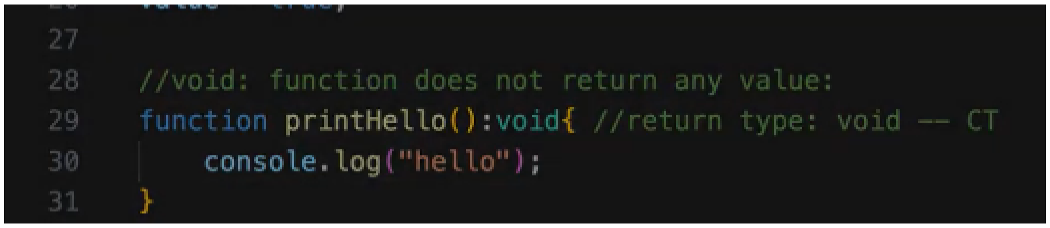


```
//any can store any data type
let value:any=80
value="tiger"
value='t'
value=-234324.32434
value=true
```

Void return type for function –

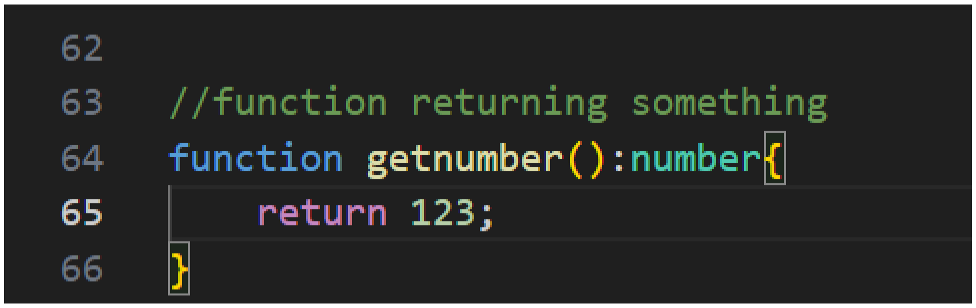


```
//void function
//does not have any return
function printhello():void{
    console.log("hello")
}
```

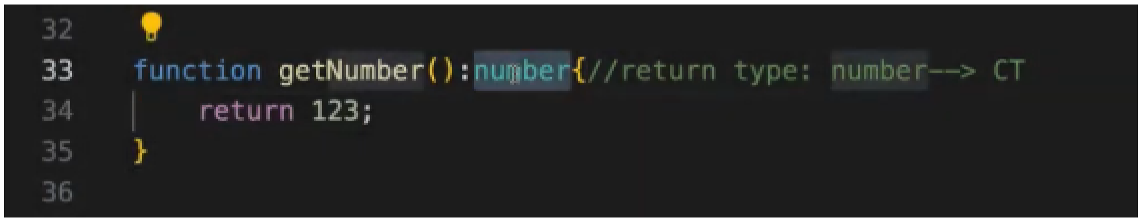


```
//void: function does not return any value:
function printHello():void{ //return type: void -- CT
    console.log("hello");
}
```

Return number-

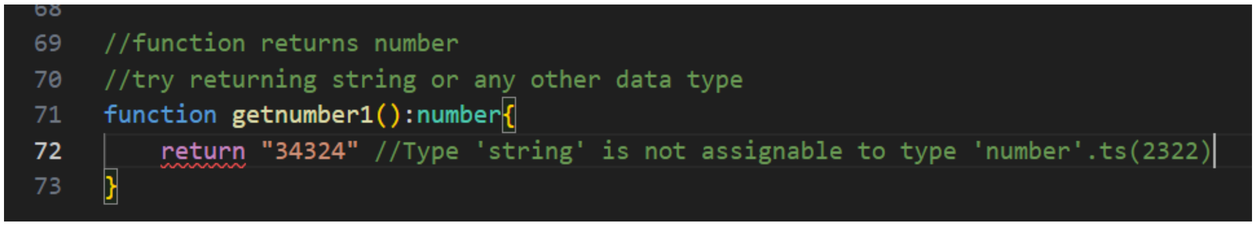


```
//function returning something
function getnumber():number{
    return 123;
}
```



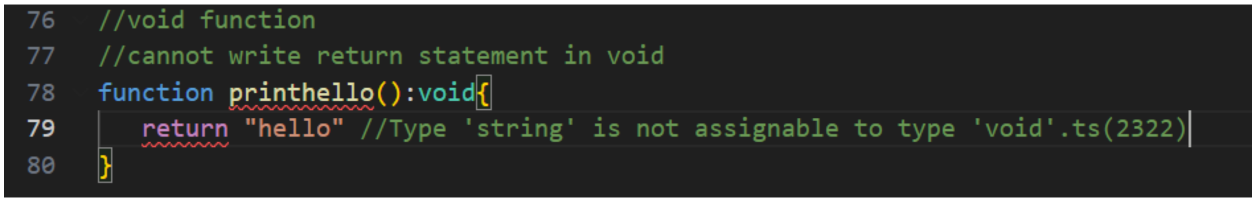
```
function getNumber():number{//return type: number--> CT
    return 123;
}
```

Cannot return other types-



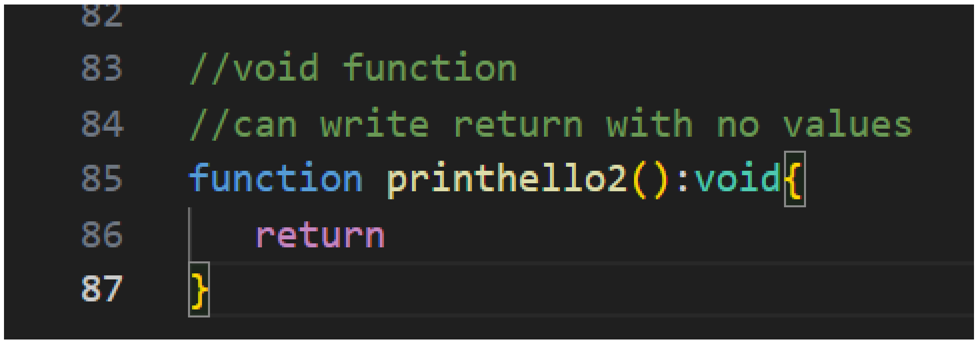
```
//function returns number
//try returning string or any other data type
function getnumber1():number{
    return "34324" //Type 'string' is not assignable to type 'number'.ts(2322)
}
```

Void and return cannot be together-



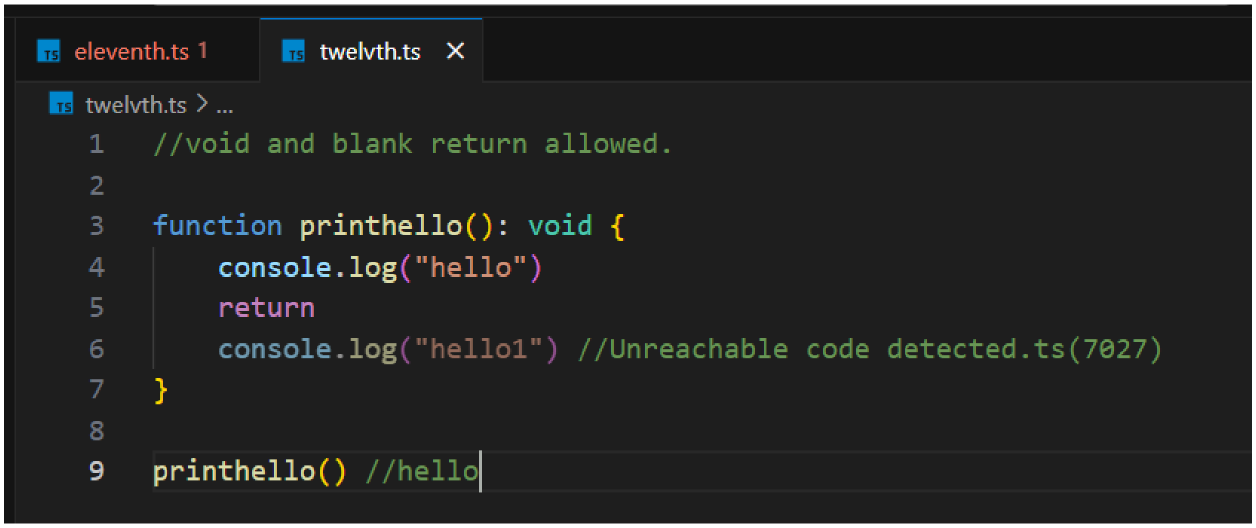
```
//void function
//cannot write return statement in void
function printhello():void{
    return "hello" //Type 'string' is not assignable to type 'void'.ts(2322)
}
```

Void and blank return allowed-



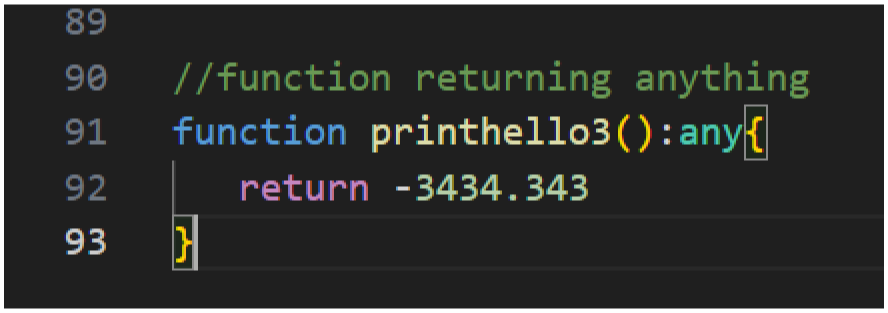
```
//void function
//can write return with no values
function printhello2():void{
    return
}
```

Some variations tried for console-



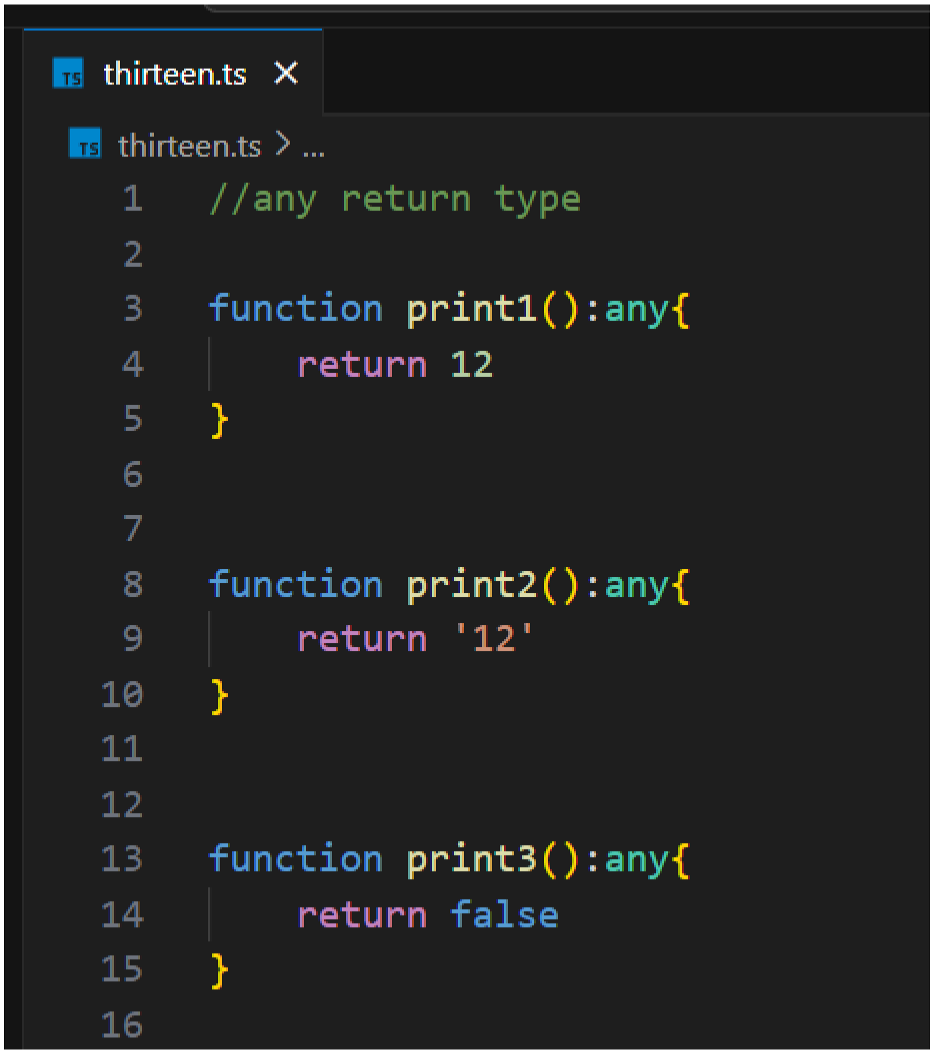
```
//void and blank return allowed.

function printhello(): void {
    console.log("hello")
    return
    console.log("hello1") //Unreachable code detected.ts(7027)
}

printhello() //hello
```

Any return type-



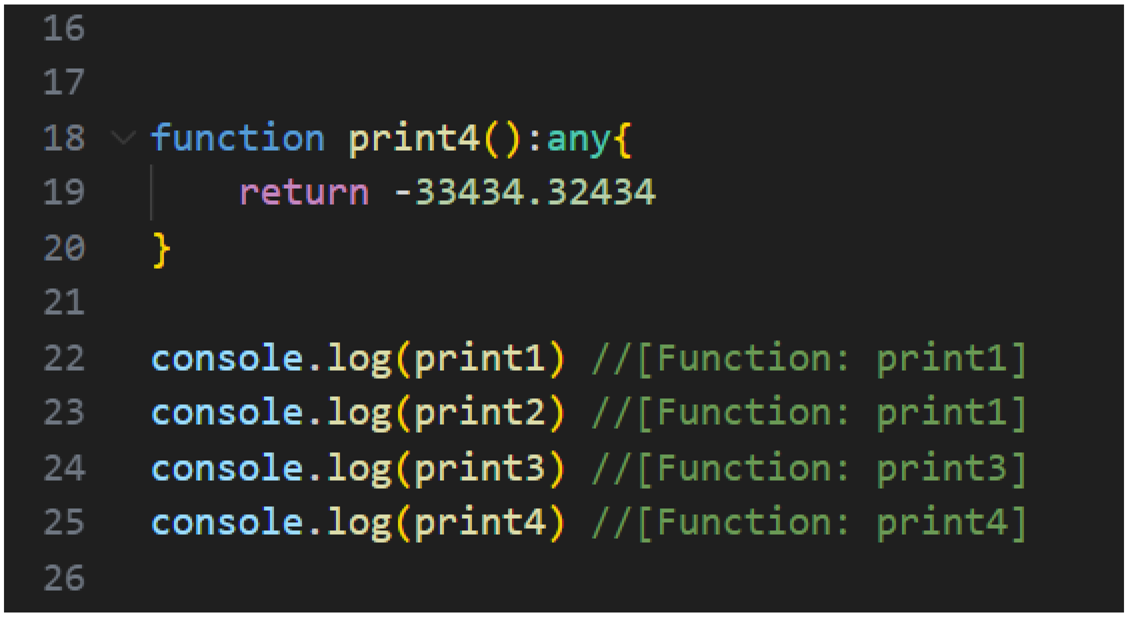
```
//function returning anything
function printhello3():any{
    return -3434.343
}
```

Variations tried–

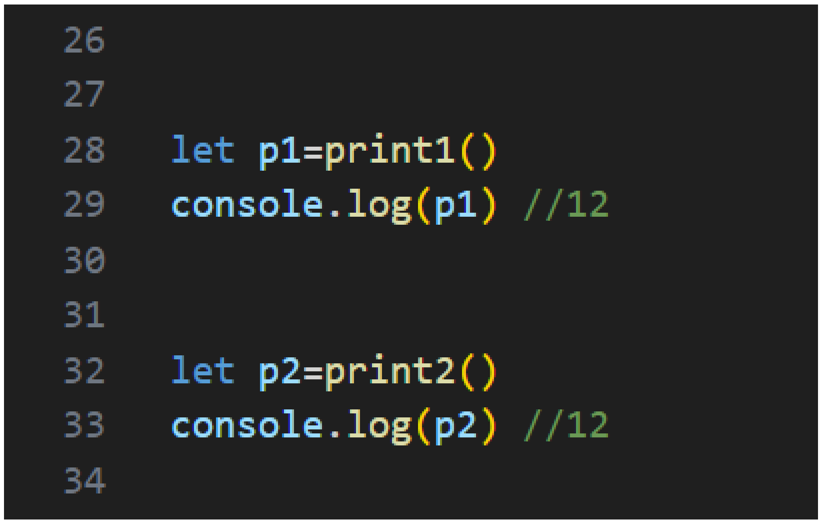


thirteen.ts

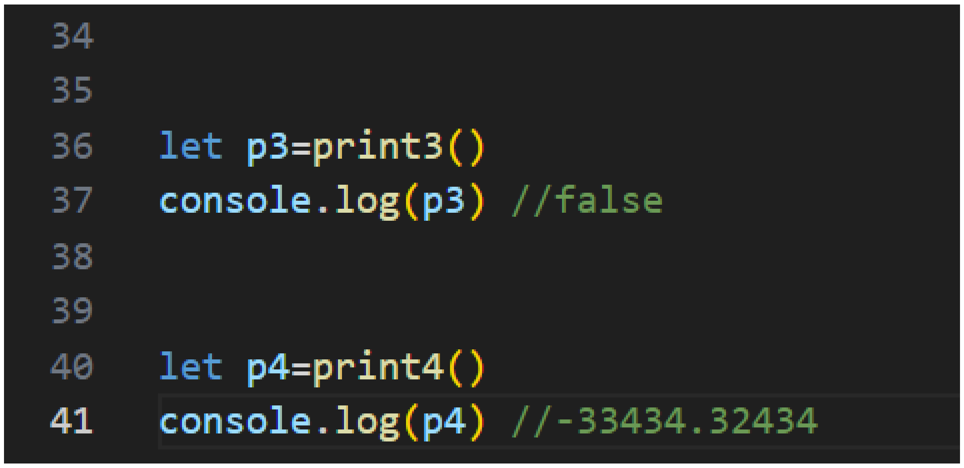
```
//any return type

function print1():any{
    return 12
}


function print2():any{
    return '12'
}


function print3():any{
    return false
}

```



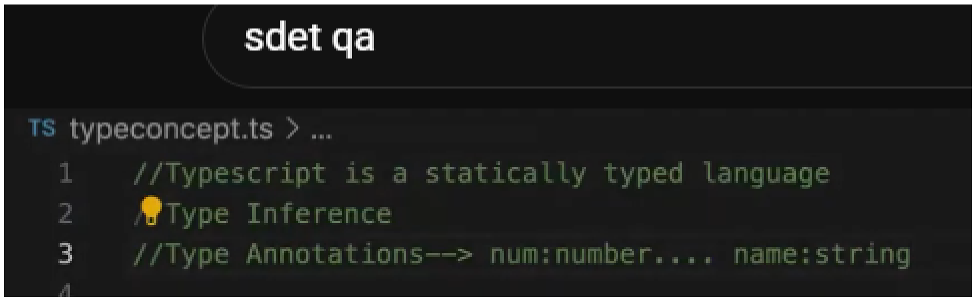
```
function print4():any{
    return -33434.32434
}

console.log(print1) //[Function: print1]
console.log(print2) //[Function: print1]
console.log(print3) //[Function: print3]
console.log(print4) //[Function: print4]
```



```
let p1=print1()
console.log(p1) //12


let p2=print2()
console.log(p2) //12
```

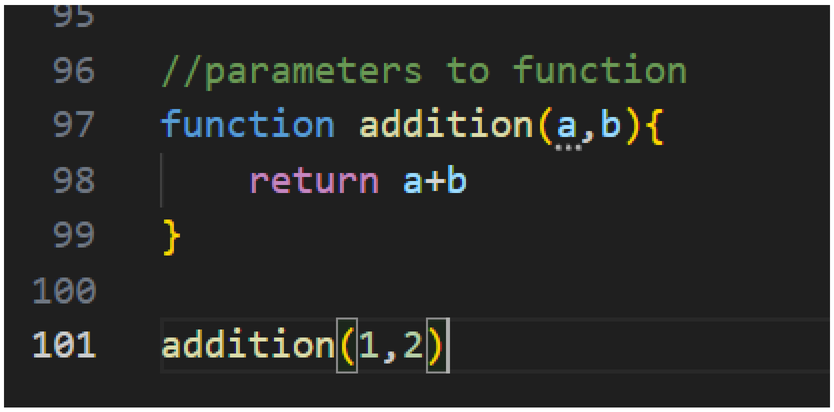


```
let p3=print3()
console.log(p3) //false


let p4=print4()
console.log(p4) //-33434.32434
```

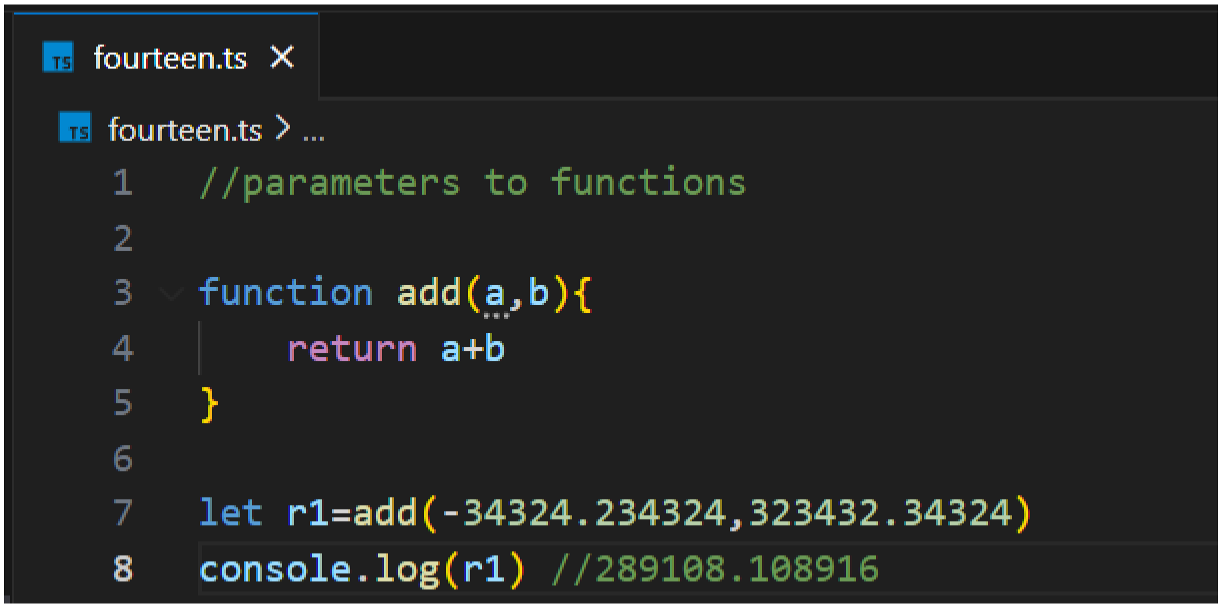


```
//Typescript is a statically typed language
//Type Inference
//Type Annotations--> num:number.... name:string
```

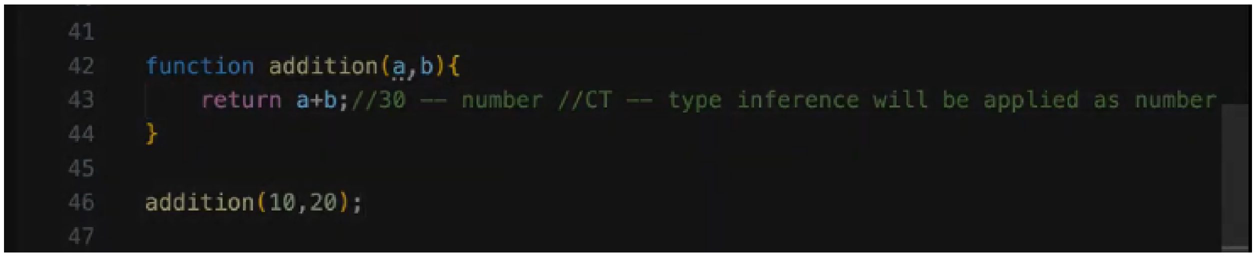
Parameters to function-



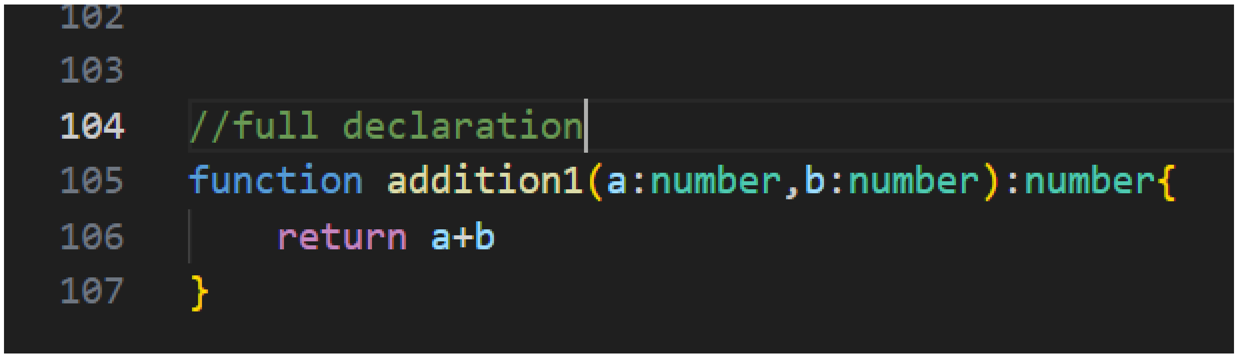
```
//parameters to function
function addition(a,b){
    return a+b
}

addition(1,2)
```

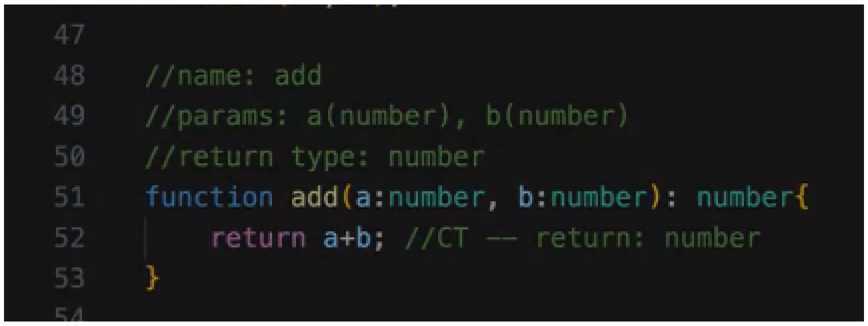


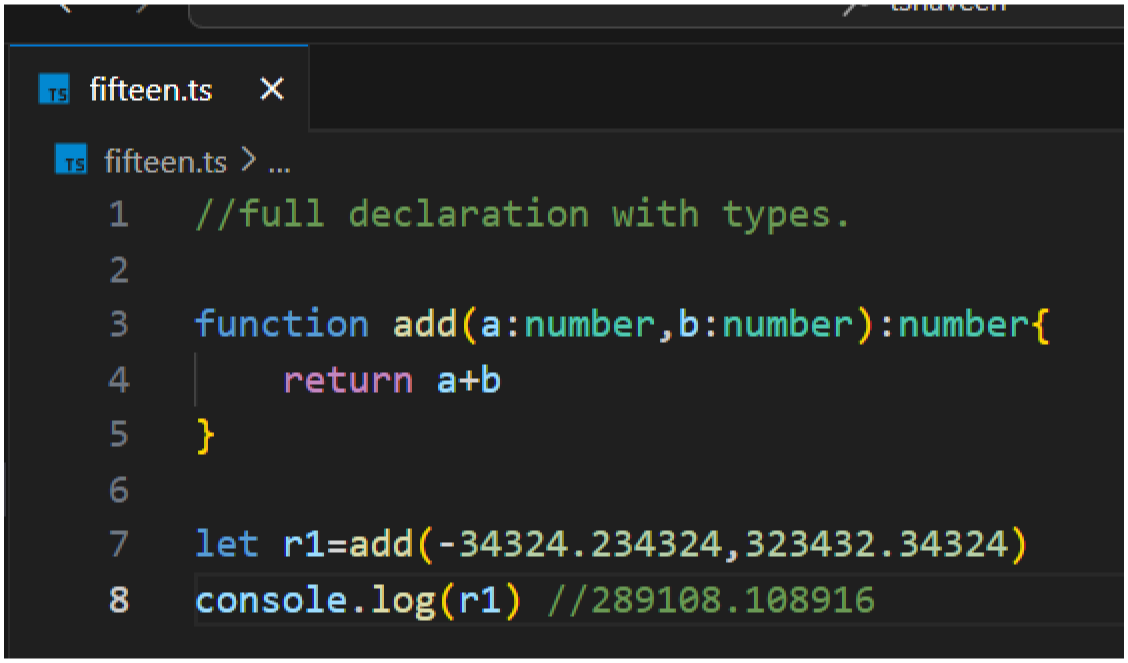
```
//parameters to functions

function add(a,b){
    return a+b
}

let r1=add(-34324.234324,323432.34324)
console.log(r1) //289108.108916
```


```
function addition(a,b){
    return a+b;//30 -- number //CT -- type inference will be applied as number
}

addition(10,20);
```

Full declaration -


```
//full declaration
function addition1(a:number,b:number):number{
    return a+b
}
```


```
//name: add
//params: a(number), b(number)
//return type: number
function add(a:number, b:number): number{
    return a+b; //CT -- return: number
}
```



```
//full declaration with types.

function add(a:number,b:number):number{
    return a+b
}

let r1=add(-34324.234324,323432.34324)
console.log(r1) //289108.108916
```