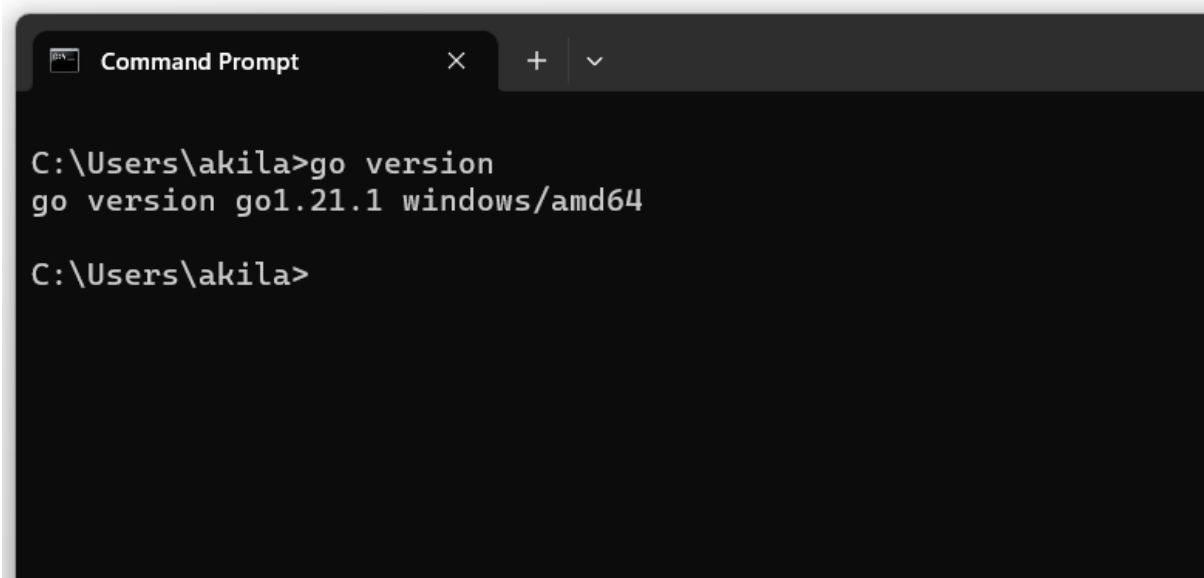


1. Go installed

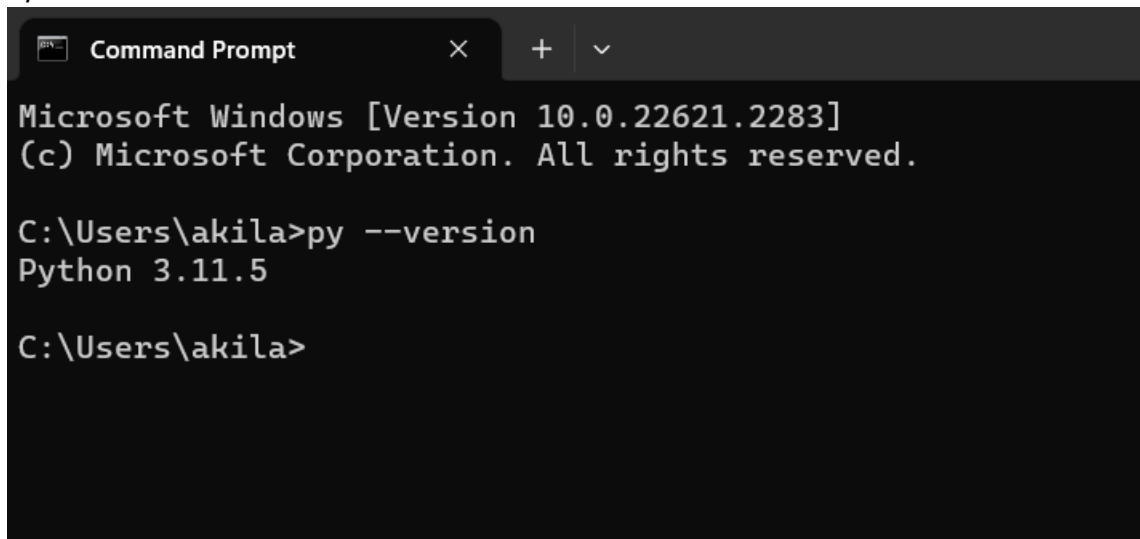


```
Command Prompt

C:\Users\akila>go version
go version go1.21.1 windows/amd64

C:\Users\akila>
```

2. Python installed



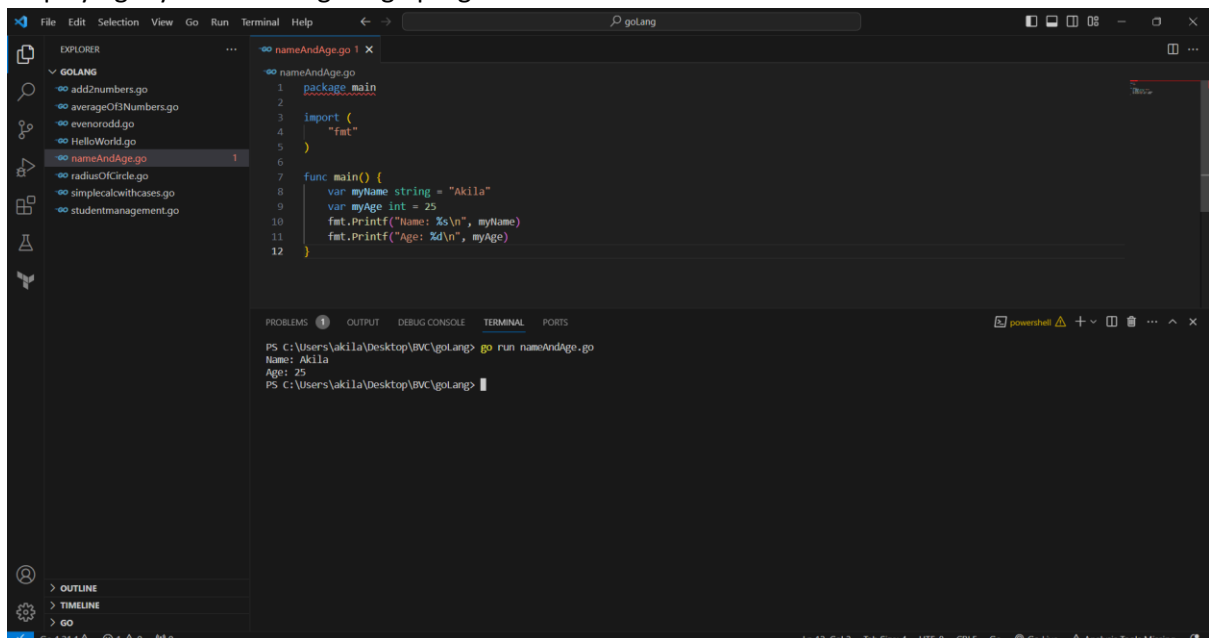
```
Command Prompt

Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

C:\Users\akila>py --version
Python 3.11.5

C:\Users\akila>
```

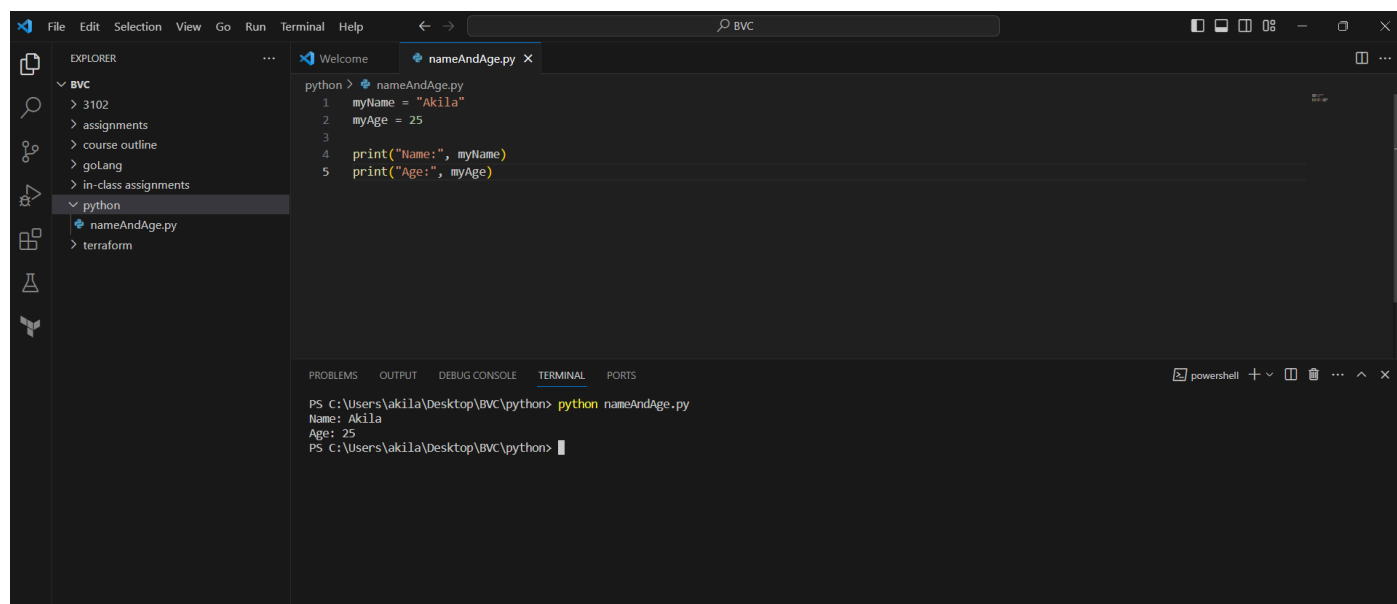
3. Displaying my name and age in go program



```
nameAndAge.go 1 X
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func main() {
8     var myName string = "Akila"
9     var myAge int = 25
10    fmt.Printf("Name: %s\n", myName)
11    fmt.Printf("Age: %d\n", myAge)
12 }
```

```
PS C:\Users\akila\Desktop\BVC\gotlang> go run nameAndAge.go
Name: Akila
Age: 25
PS C:\Users\akila\Desktop\BVC\gotlang>
```

4. Print name and age in python program



The screenshot shows the Visual Studio Code editor with a file named `nameAndAge.py` open. The code in the editor is as follows:

```
python > nameAndAge.py
1 myName = "Akila"
2 myAge = 25
3
4 print("Name:", myName)
5 print("Age:", myAge)
```

The Explorer sidebar on the left shows the project structure with folders like `BVC`, `3102`, `assignments`, `course outline`, `goLang`, `in-class assignments`, `python`, and `terraform`. The `python` folder contains `nameAndAge.py`.

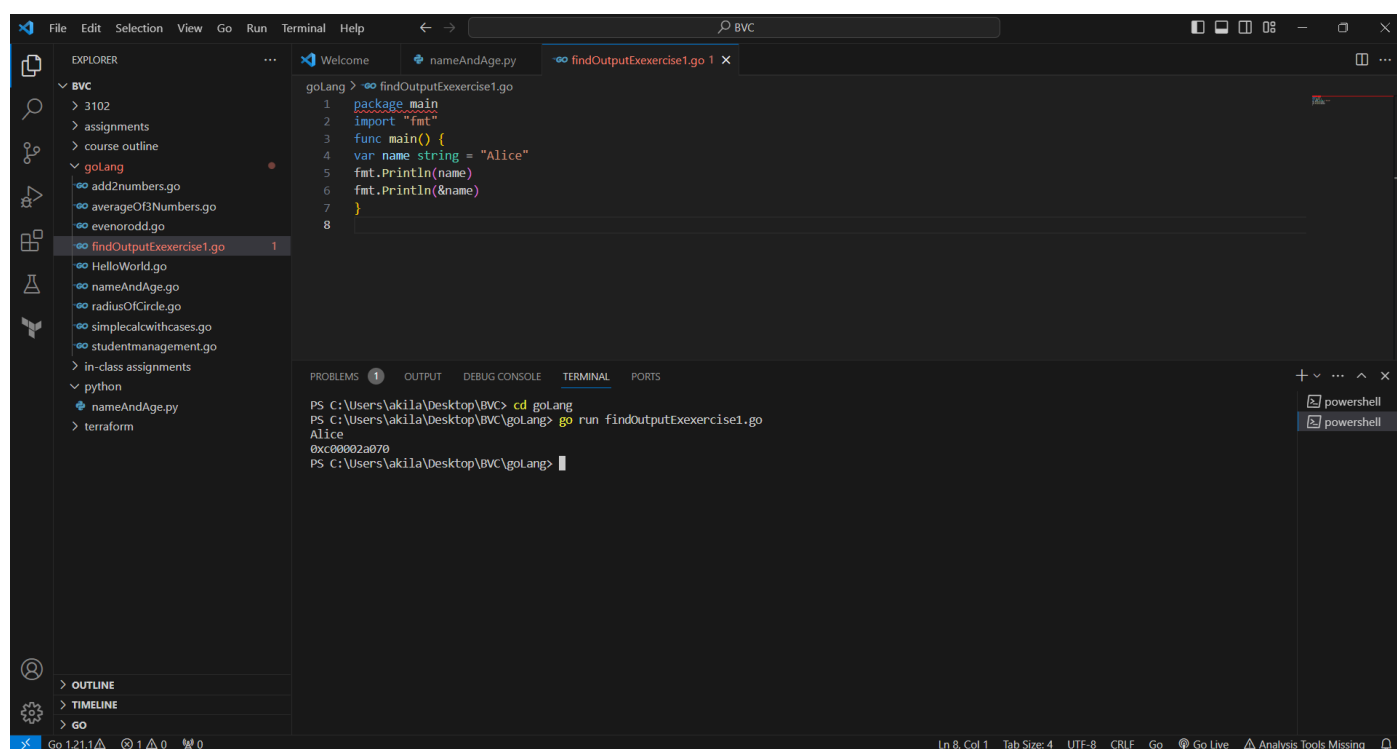
The Terminal at the bottom shows the command `python nameAndAge.py` being executed, resulting in the output:

```
PS C:\Users\akila\Desktop\BVC\python> python nameAndAge.py
Name: Akila
Age: 25
PS C:\Users\akila\Desktop\BVC\python>
```

5. Different ways of creating string variable in go.

- `var name string` ----- initialised string typed variable without assigning a value
- `var name string = "Akila"` ----- initialised string typed variable with assigning a value
- `name := "Akila"` ----- directly assigning a string value;

6. Run the code and see the output



The screenshot shows the Visual Studio Code editor with a file named `findOutputExexercice1.go` open. The code in the editor is as follows:

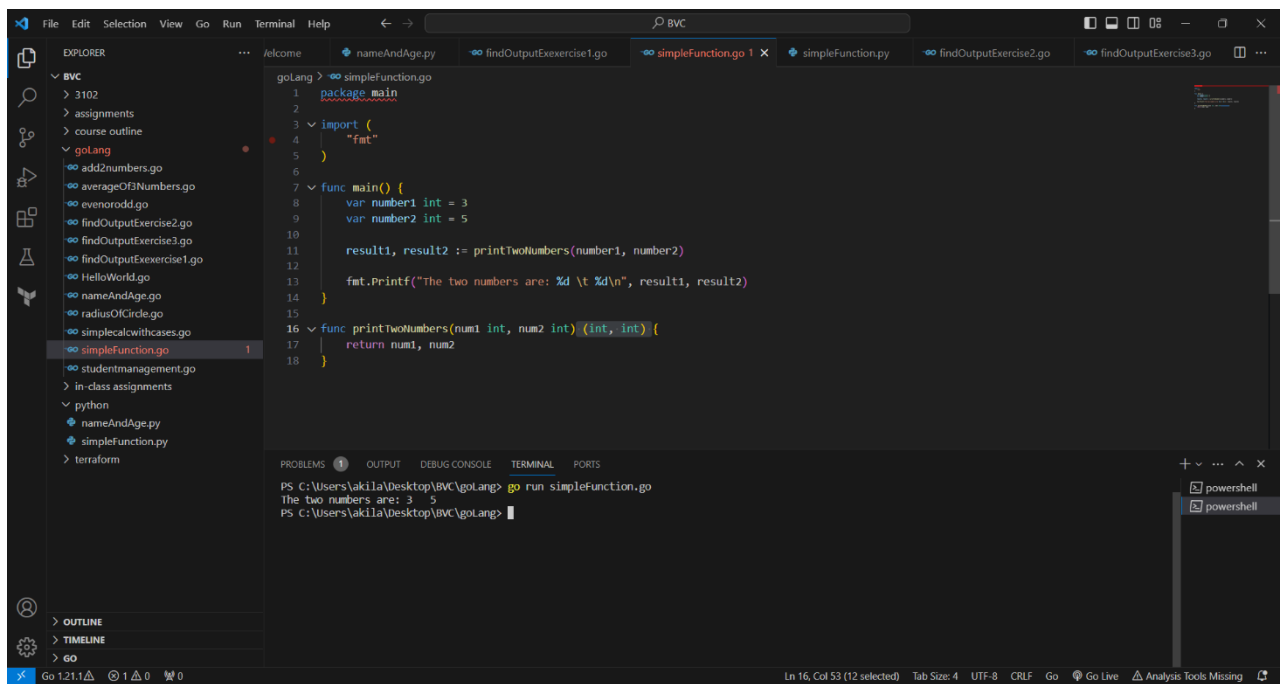
```
goLang > findOutputExexercice1.go
1 package main
2 import "fmt"
3 func main() {
4     var name string = "Alice"
5     fmt.Println(name)
6     fmt.Println(&name)
7 }
8
```

The Explorer sidebar on the left shows the project structure with folders like `BVC`, `3102`, `assignments`, `course outline`, `goLang`, `in-class assignments`, `python`, and `terraform`. The `goLang` folder contains several Go files, including `findOutputExexercice1.go`.

The Terminal at the bottom shows the command `go run findOutputExexercice1.go` being executed, resulting in the output:

```
PS C:\Users\akila\Desktop\BVC> cd goLang
PS C:\Users\akila\Desktop\BVC\goLang> go run findOutputExexercice1.go
Alice
0xc00002a870
PS C:\Users\akila\Desktop\BVC\goLang>
```

7. Simple function to print two numbers in goLang

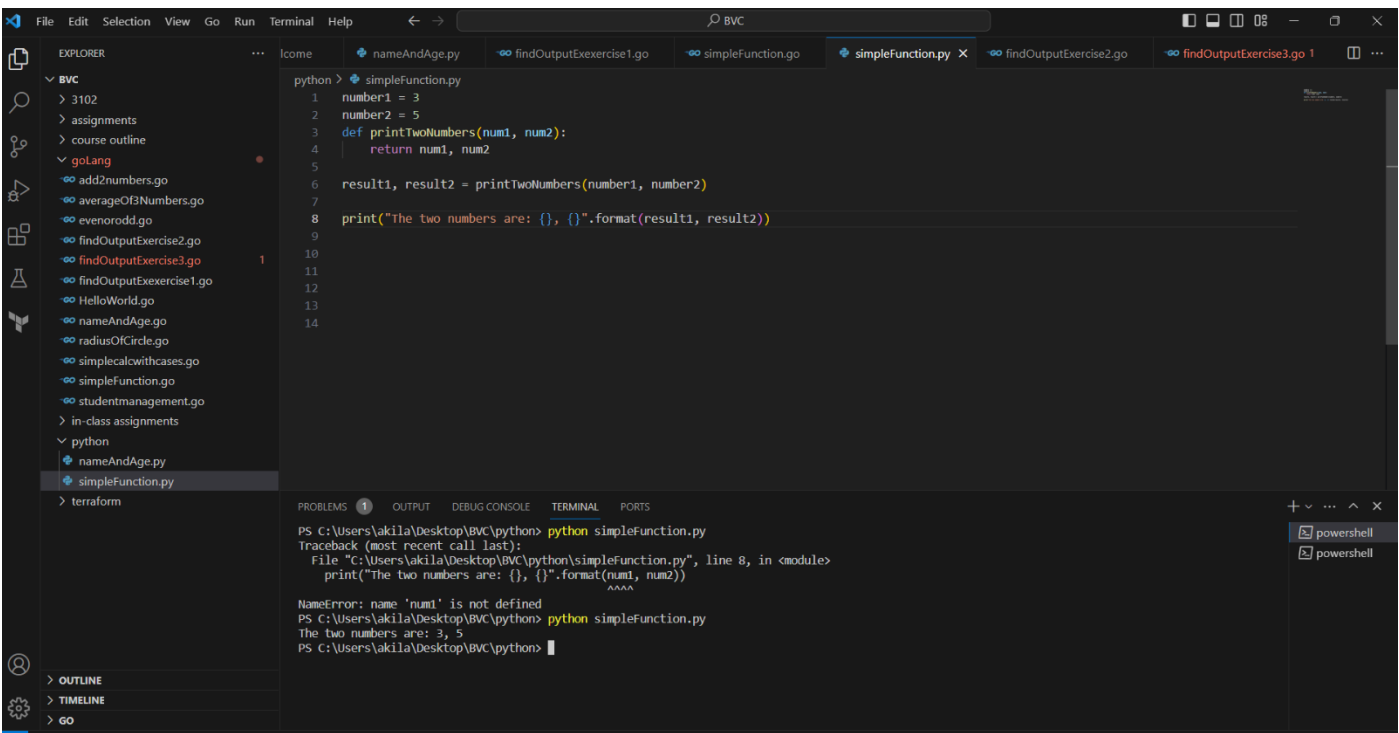


The screenshot shows the Visual Studio Code editor with a Go file named `simpleFunction.go` open. The code defines a `main` function that initializes two variables, `number1` and `number2`, and calls a `printTwoNumbers` function. The `printTwoNumbers` function is defined to take two integers and return them. The terminal output shows the program running successfully and printing "The two numbers are: 3 5".

```
goLang > simpleFunction.go
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func main() {
8     var number1 int = 3
9     var number2 int = 5
10
11     result1, result2 := printTwoNumbers(number1, number2)
12
13     fmt.Printf("The two numbers are: %d %d\n", result1, result2)
14 }
15
16 func printTwoNumbers(num1 int, num2 int) (int, int) {
17     return num1, num2
18 }
```

```
PS C:\Users\akila\Desktop\BVC\goLang> go run simpleFunction.go
The two numbers are: 3 5
PS C:\Users\akila\Desktop\BVC\goLang>
```

8. Print two numbers using function in python

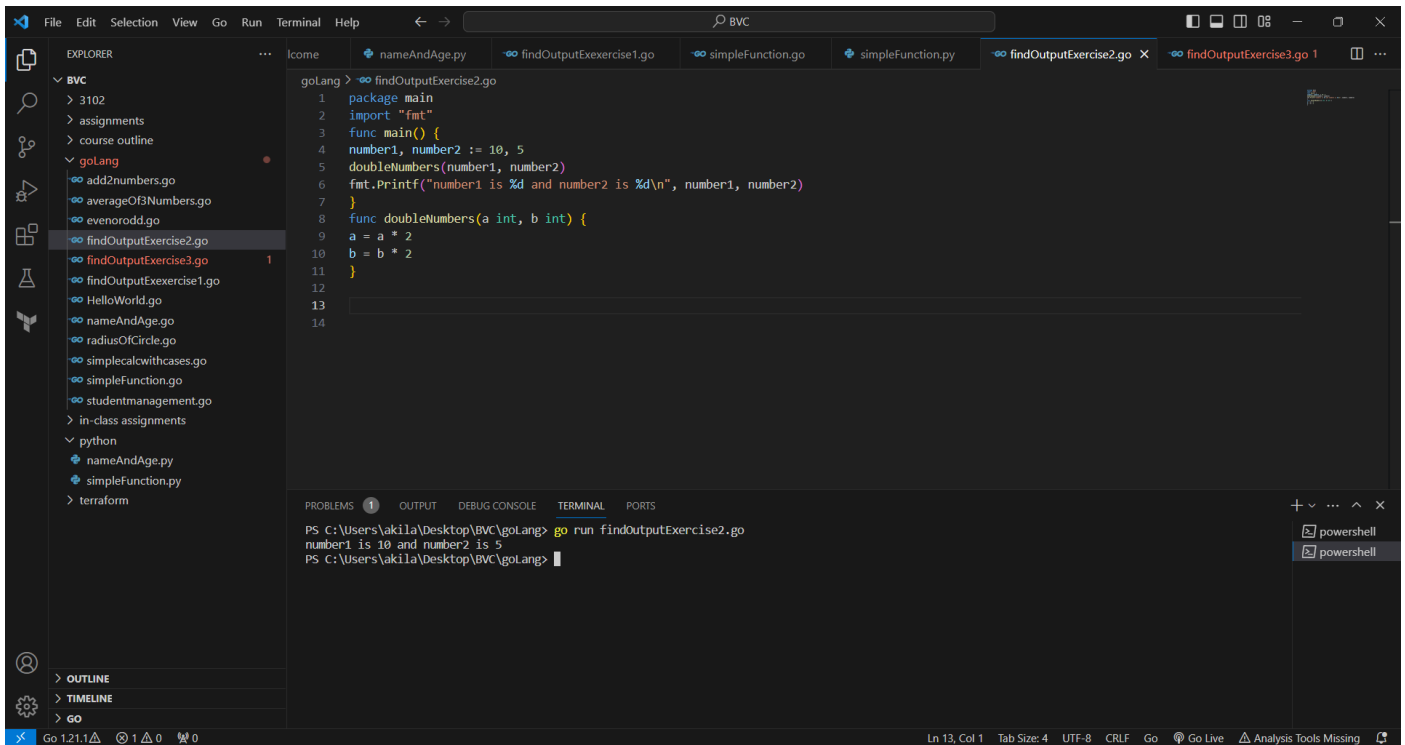


The screenshot shows the Visual Studio Code editor with a Python file named `simpleFunction.py` open. The code defines a `printTwoNumbers` function that takes two arguments and returns them. The `main` function calls `printTwoNumbers` with the values 3 and 5. The terminal output shows the program running successfully and printing "The two numbers are: 3, 5".

```
python > simpleFunction.py
1 number1 = 3
2 number2 = 5
3 def printTwoNumbers(num1, num2):
4     return num1, num2
5
6 result1, result2 = printTwoNumbers(number1, number2)
7
8 print("The two numbers are: {}, {}".format(result1, result2))
9
10
11
12
13
14
```

```
PS C:\Users\akila\Desktop\BVC\python> python simpleFunction.py
Traceback (most recent call last):
  File "C:\Users\akila\Desktop\BVC\python\simpleFunction.py", line 8, in <module>
    print("The two numbers are: {}, {}".format(num1, num2))
    ^^^^^
NameError: name 'num1' is not defined
PS C:\Users\akila\Desktop\BVC\python> python simpleFunction.py
The two numbers are: 3, 5
PS C:\Users\akila\Desktop\BVC\python>
```

9. Run the code and explain it



The screenshot shows the Visual Studio Code interface with a Go file named `findOutputExercise2.go` open. The code defines a `main` function that initializes `number1` to 10 and `number2` to 5, then calls `doubleNumbers` with these values. The `doubleNumbers` function multiplies its arguments by 2 but does not return the results. The `main` function then prints the values of `number1` and `number2`. The terminal output shows the command `go run findOutputExercise2.go` being executed, resulting in the output `number1 is 10 and number2 is 5`.

```
goLang > findOutputExercise2.go
1 package main
2 import "fmt"
3 func main() {
4     number1, number2 := 10, 5
5     doubleNumbers(number1, number2)
6     fmt.Printf("number1 is %d and number2 is %d\n", number1, number2)
7 }
8 func doubleNumbers(a int, b int) {
9     a = a * 2
10    b = b * 2
11 }
12
13
14
```

```
PS C:\Users\ak11a\Desktop\BVC\goLang> go run findOutputExercise2.go
number1 is 10 and number2 is 5
PS C:\Users\ak11a\Desktop\BVC\goLang>
```

In the above code,

- The two variables `number1` and `number2` are initialized and assigned to the values 10 and 5 respectively.
- Then passing the variables to a function `doubleNumbers` in line 5 as arguments.
- Inside the function we are multiplying the number by 2 but **we are not returning it**. So the scope of the values are inside the function only. It won't reflect in the main method.
- In line 6 we are printing the number 1 and number 2 ; So the value is 10 and 5 .
- If we returned the numbers then the values will get changed in line 6.

9.b

```
1 package main
2 import "fmt"
3 func main() {
4     number1, number2 := 10, 5
5     doubleNumbers(&number1, &number2)
6     fmt.Printf("number1 is %d and number2 is %d\n", number1, number2)
7 }
8 func doubleNumbers(a *int, b *int) {
9     *a = *a * 2
10    *b = *b * 2
11 }
12
```

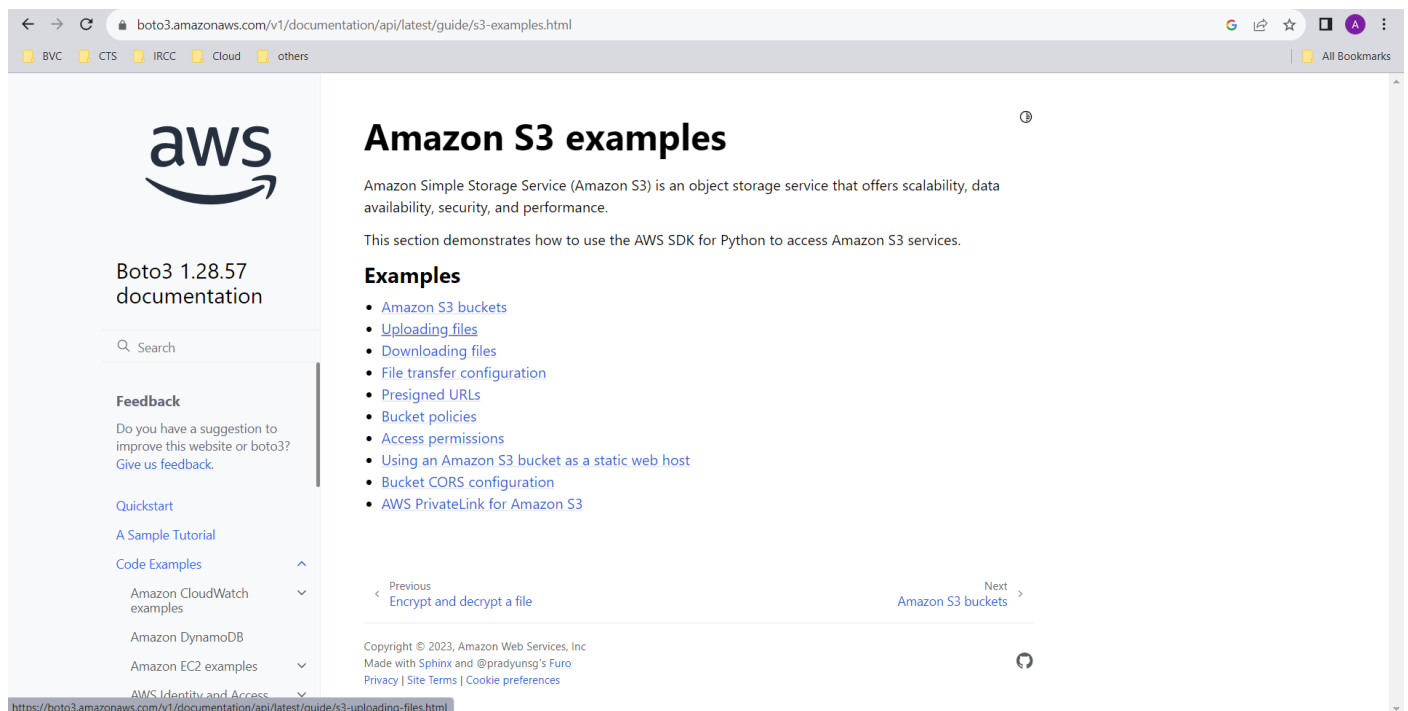
PS C:\Users\akila\Desktop\BVC\goLang> go run findOutputExercise3.go
number1 is 20 and number2 is 10
PS C:\Users\akila\Desktop\BVC\goLang>

In the above code,

- The two variables number1 and number2 are initialized and assigned to the values 10 and 5 respectively.
- Then passing the address of the variables to a function doubleNumbers in line 6 as arguments.
- ‘&’ gives the memory address of the variables instead of the values.
- Inside the function we are multiplying the value of the numbers by 2.
- ‘*’ gives the values in the given memory address.
- After multiplication we are storing the values in the same memory address.
- In line 6 we are printing the number 1 and number 2 ; So the value is 20 and 10 . As we stored the updated value in the same address.

10. Python SDK documentation for AWS S3 service.

<https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-examples.html>



Go SDK documentation for AWS S3 service.

← → ↻ docs.aws.amazon.com/sdk-for-go/api/service/s3/

BVC CTS IRCC Cloud others

All Bookmarks

AWS SDK for Go API ReferenceDeveloper GuideBlogFeedback

aws
am
auth
bearer
awserr
awsutil
client
metadata
corehandlers
credentials
ec2rolecreds
endpointcreds
plugincreds
processcreds
ssocreds
stscreds
crr
csm
defaults
ec2metadata
endpoints
request
session
signer

```
import "github.com/aws/aws-sdk-go/service/s3"
```

Overview
Examples
Constants

Overview ▾

Package s3 provides the client and types for making API requests to Amazon Simple Storage Service.

See <https://docs.aws.amazon.com/goto/WebAPI/s3-2006-03-01> for more information on this service.

See s3 package documentation for more information. <https://docs.aws.amazon.com/sdk-for-go/api/service/s3/>

Using the Client

To contact Amazon Simple Storage Service with the SDK use the New function to create a new service client. With that client you can make API requests to the service. These clients are safe to use concurrently.

See the SDK's documentation for more information on how to use the SDK. <https://docs.aws.amazon.com/sdk-for-go/api/>

See aws.Config documentation for more information on configuring SDK clients. <https://docs.aws.amazon.com/sdk-for-go/api/aws/#Config>

See the Amazon Simple Storage Service client S3 for more information on creating client for this service. <https://docs.aws.amazon.com/sdk-for-go/api/service/s3/#New>

Unload Managers