git & golang book

Read golang book and work with git

golang book

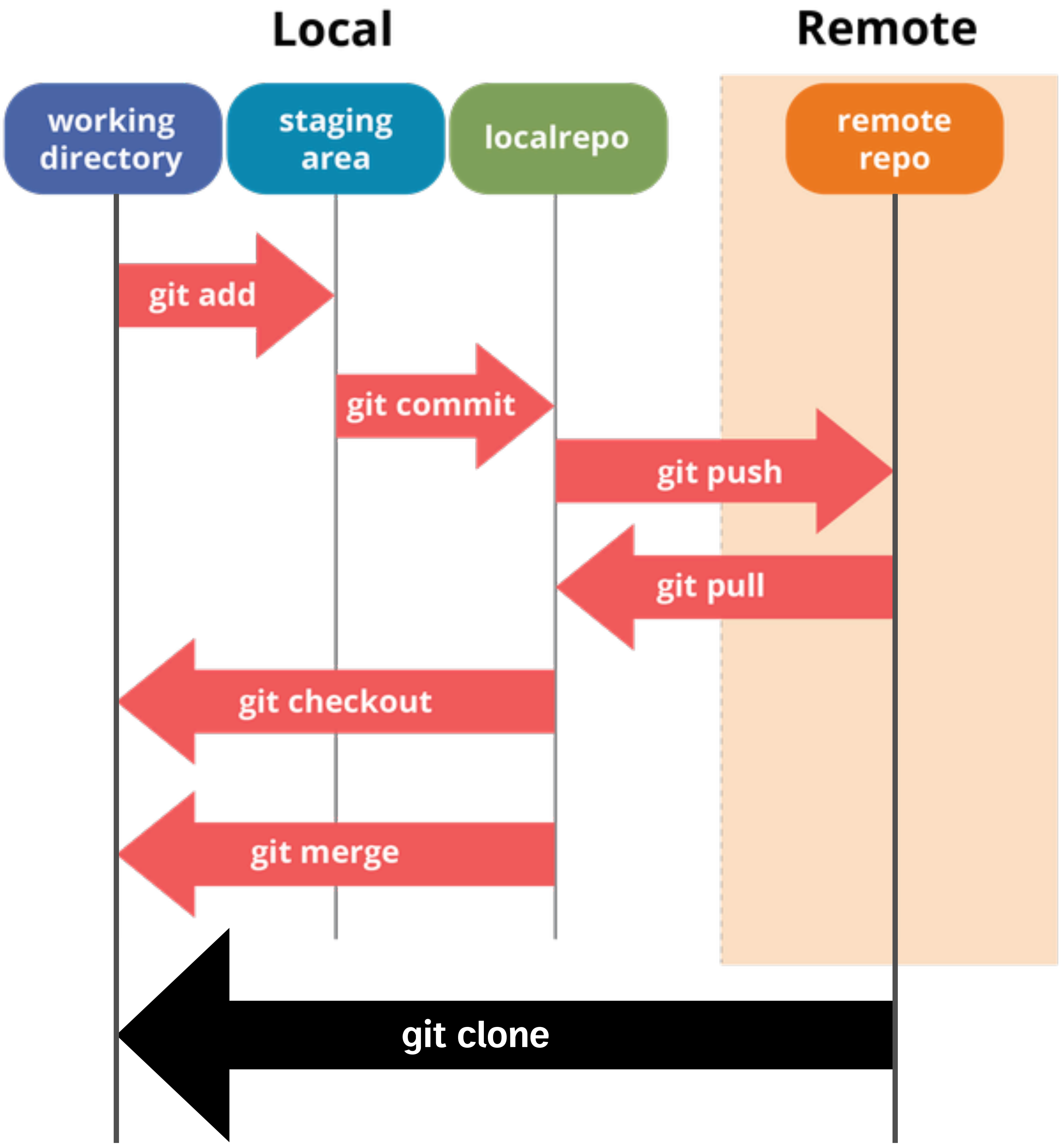
git

github account:

https://github.com/[yourname]

example:

https://github.com/boyone


Local
Remote
working directory
staging area
localrepo
remote repo
git add
git commit
git push
git pull
git checkout
git merge
git clone

Clone go-101

clone go-101 to your workspace:

```
>git clone https://github.com/boyone/go-101.git
```

golang book [1]

make working directory: [windows]

```
>md src\dojo\golang-book
```

make working directory: [linux/Mac]

```
>mkdir -p src/dojo/golang-book
```

golang book [2]

go to golang-book directory:

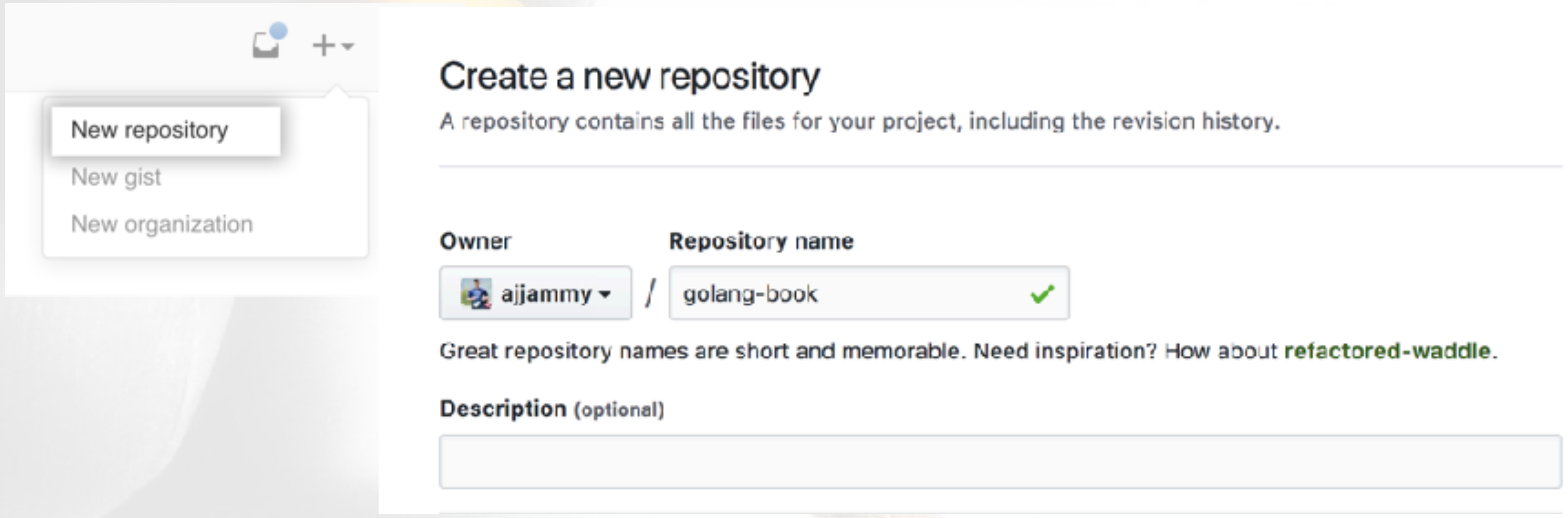
```
>cd src\dojo\golang-book
```

git init for golang-book directory:

```
>git init
```

golang book [3]

create github repository:



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: ajjammy / Repository name: golang-book

Great repository names are short and memorable. Need inspiration? How about refactored-waddle.

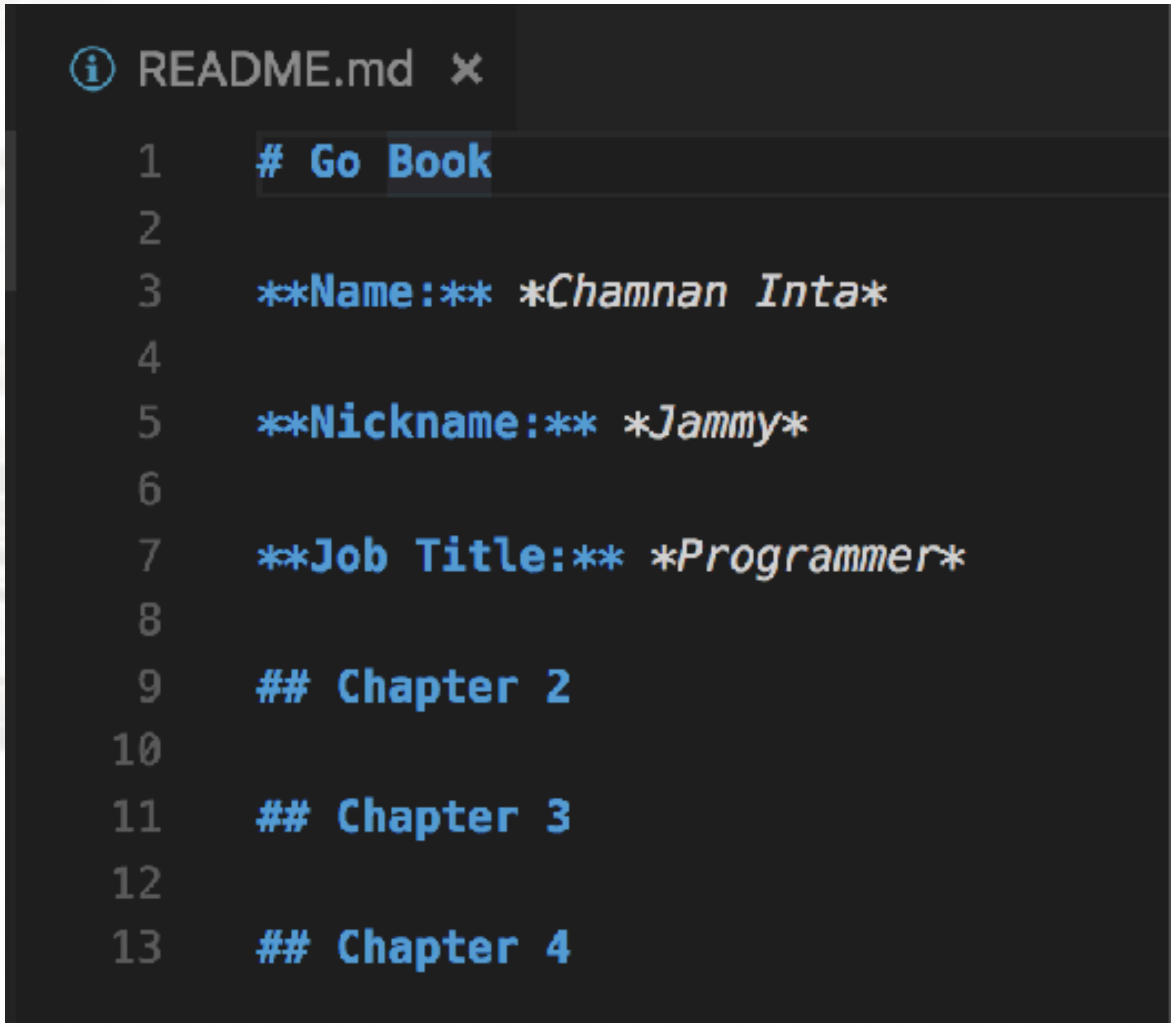
Description (optional)

add remote:

```
>git remote add origin https://github.com/<user>/golang-book.git
>git remote -v
```

golang book [4]

create README.md file:


README.md

```
# Go Book

**Name:** *Chamnan Inta*

**Nickname:** *Jammy*

**Job Title:** *Programmer*

## Chapter 2

## Chapter 3

## Chapter 4
```

golang book [5]

git add / git commit :

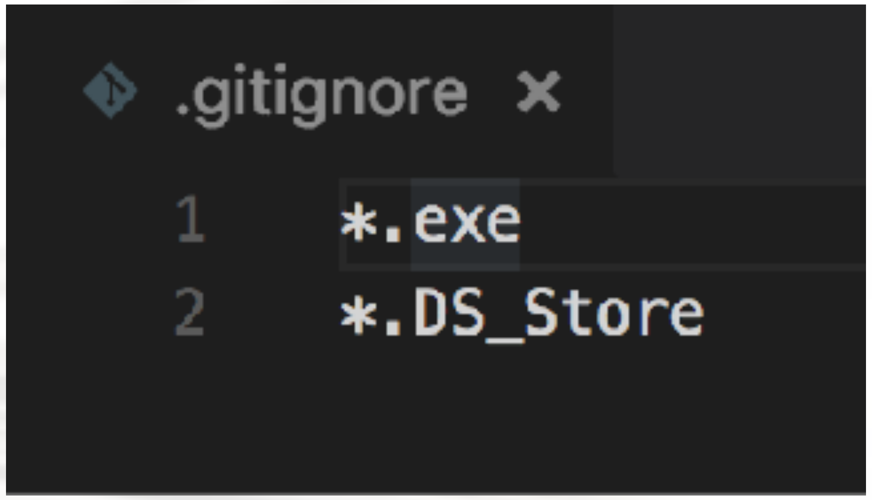
```
>git add README.md
>git commit -m “Add README.md file”
```

git push

```
>git push -u origin master
```

golang book [6]

create .gitignore file:



```
*.exe
*.DS_Store
```

golang book [7]

git add / git commit :

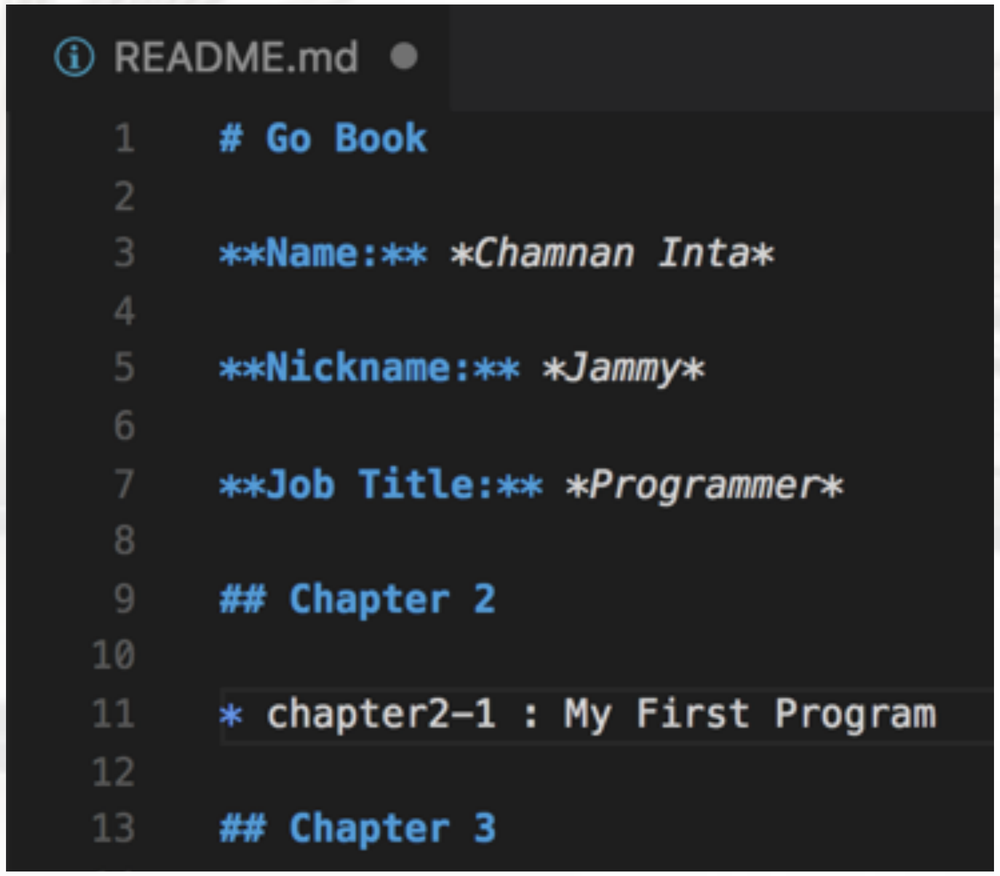
```
>git add .gitignore
>git commit -m “Add .gitignore file”
```

git push

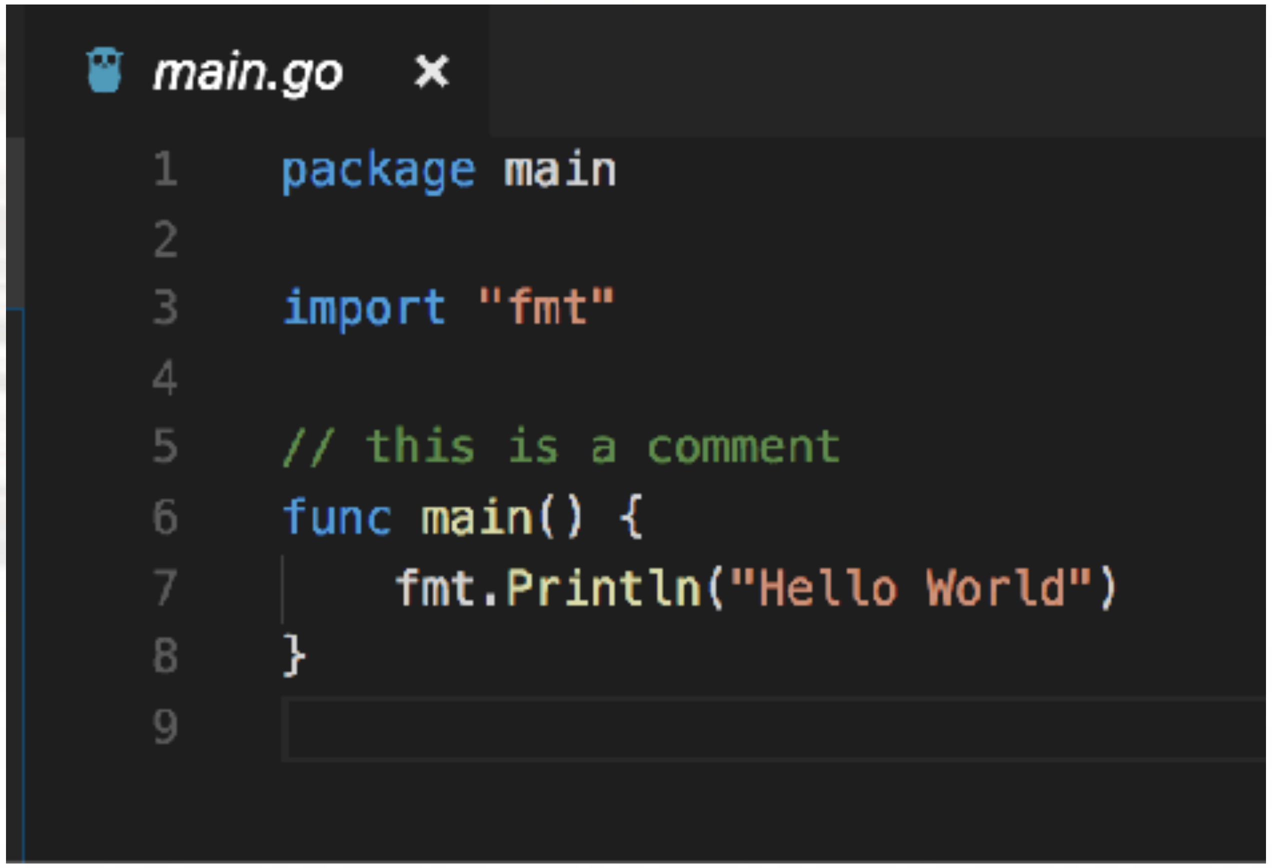
```
>git push
```

golang book [8]

1. Read book chapter 2
2. Update README.md
3. Create main.go file at directory golang-book/chapter2-1


README.md

```
# Go Book

**Name:** *Chamnan Inta*

**Nickname:** *Jammy*

**Job Title:** *Programmer*

## Chapter 2

* chapter2-1 : My First Program

## Chapter 3
```


main.go

```go
package main

import "fmt"

// this is a comment
func main() {
    fmt.Println("Hello World")
}

```

golang book [9]

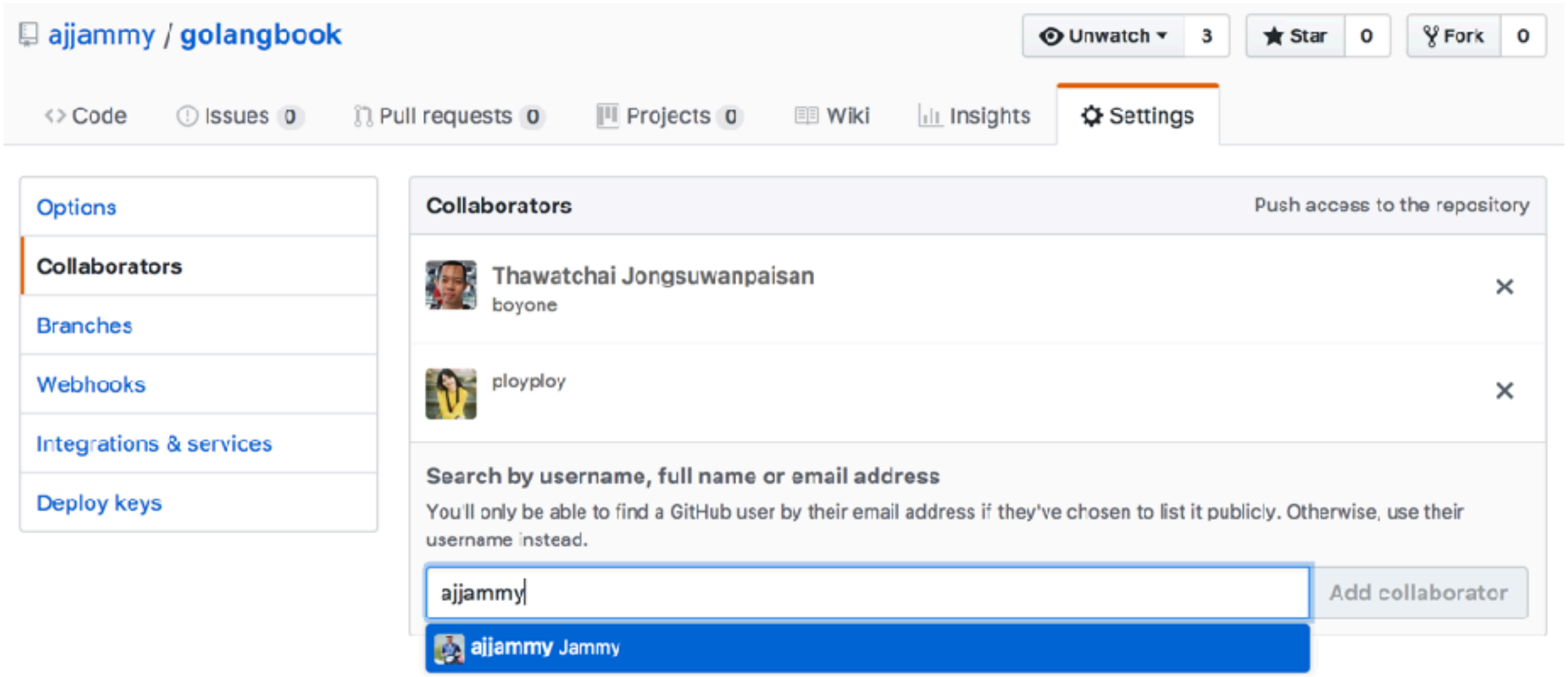
git add / git commit :

```
>git add .
>git commit -m “Add chapter2-1 My first program”
```

git push

```
>git push
```

add collaborators



ajjammy / golangbook

Collaborators — Push access to the repository

Thawatchai Jongsuwanpaisan
boyone

ployploy

Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

ajjammy

Exercise

Read book

Chapter 1 to 4

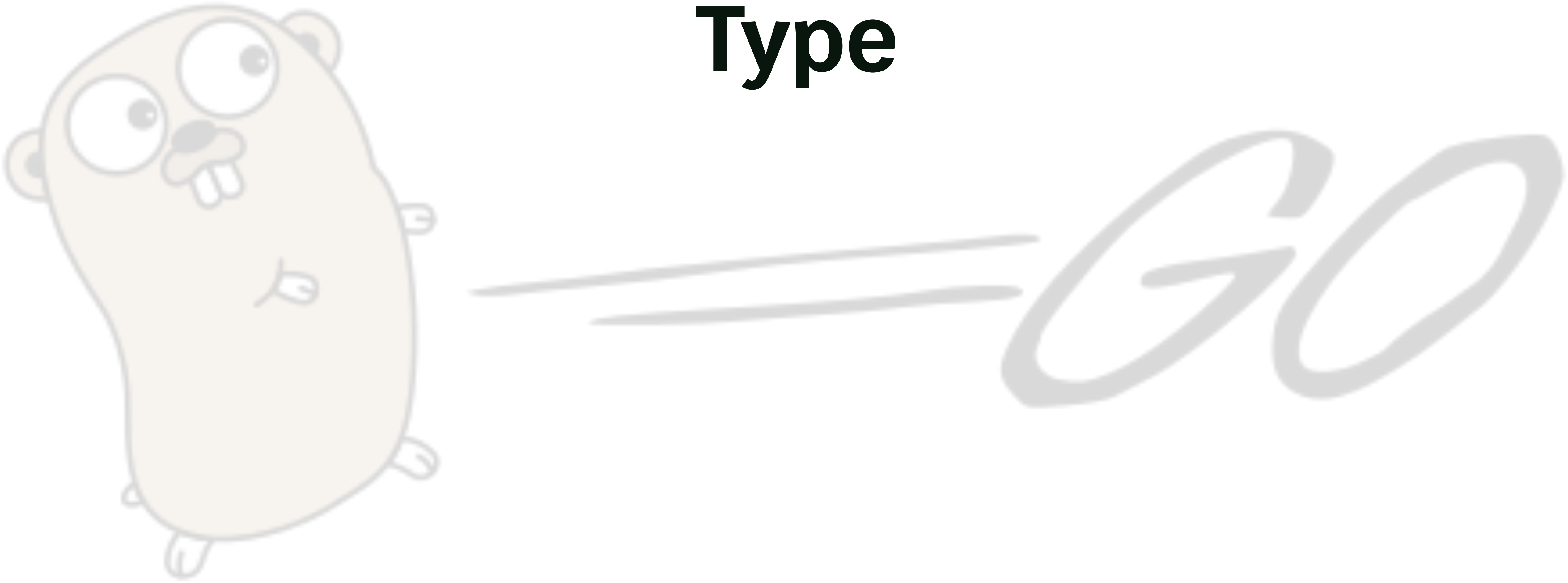
Create folder chapter<…>-…

Create file main.go

Update README.md file

```
git add
git commit
git push
```

Type



golang : type Zero Value

```go
package main

import "fmt"

func main() {
    fmt.Println("=====Zero Value=====")
    var number int
    var str string
    var boolean bool
    fmt.Printf("number: %v\n", number)
    fmt.Printf("str: '%v'\n", str)
    fmt.Printf("boolean: %v\n", boolean)
}
```

golang : type Strings

```go
package main

import "fmt"

func main() {
    fmt.Println("=====String=====")
    backticks := `hello world!,
today's good day.`
    fmt.Println(backticks)

    doubleQuotes := "hello world!,\ntoday's good day."
    fmt.Println(doubleQuotes)
}
```

golang : type Floating point [1]

```
package main

import "fmt"

func main() {
	fmt.Println("=====Floating point=====")
	third := 1.0 / 3.0
	fmt.Printf("third = %v\n", third)
	fmt.Printf("third + third + third = %v\n", third+third+third)
}
```

golang : type Floating point [2]

```go
package main

import "fmt"

func main() {
	fmt.Println("=====Comparing floating point=====")
	fmt.Println("0.1 + 0.2 == 0.3 is", 0.1+0.2 == 0.3)
	num := 0.1
	num += 0.2
	fmt.Println("num == 0.3 is", num == 0.3)
	fmt.Println("num is", num)
}
```

Variables

golang : Variables [1]

create main.go in folder chapter4-1 :

```go
package main

import "fmt"

func main() {




}
```

run -> no error -> push to your git repository

golang : Variables [2]

create main.go in folder chapter4-2 :

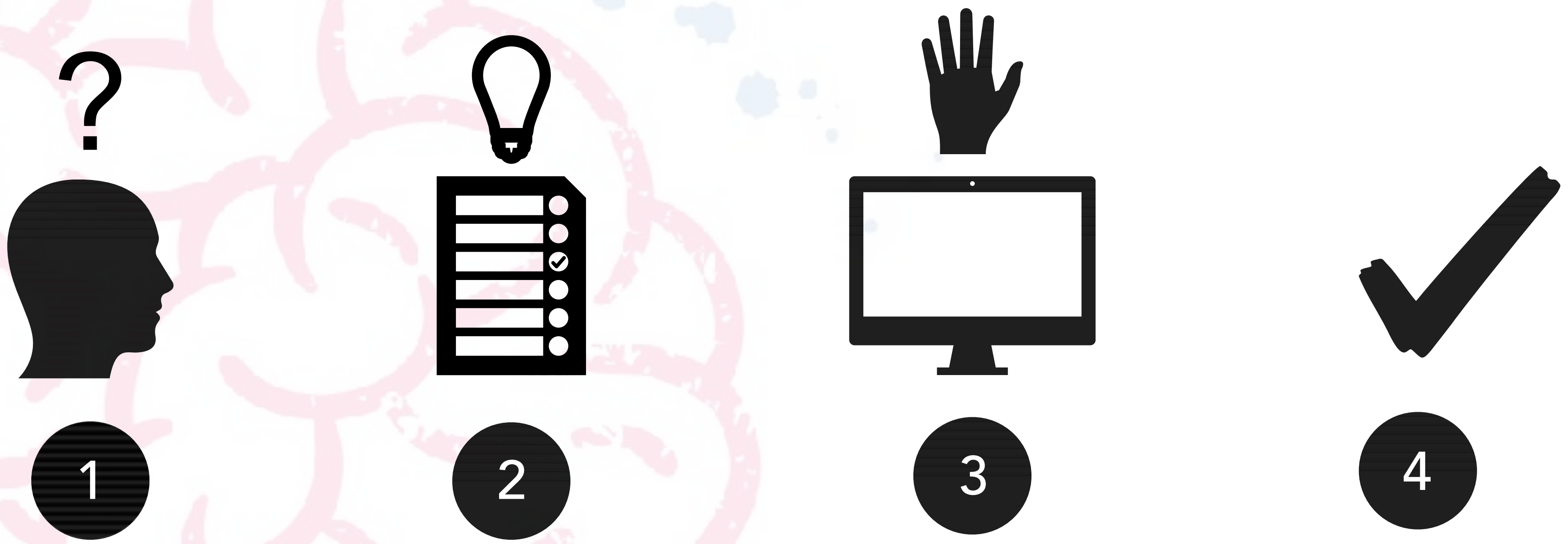
```go
package main

import "fmt"

func main() {
    fmt.Print("Enter a number: ")
    var input float64
    fmt.Scanf("%f", &input)
    output := input * 2
    fmt.Println(output)
}
```

run -> no error -> push to your git repository

Exercise

Modify main.go in folder chapter4-2 for solve

Problem No.5 of Chapter 4 :

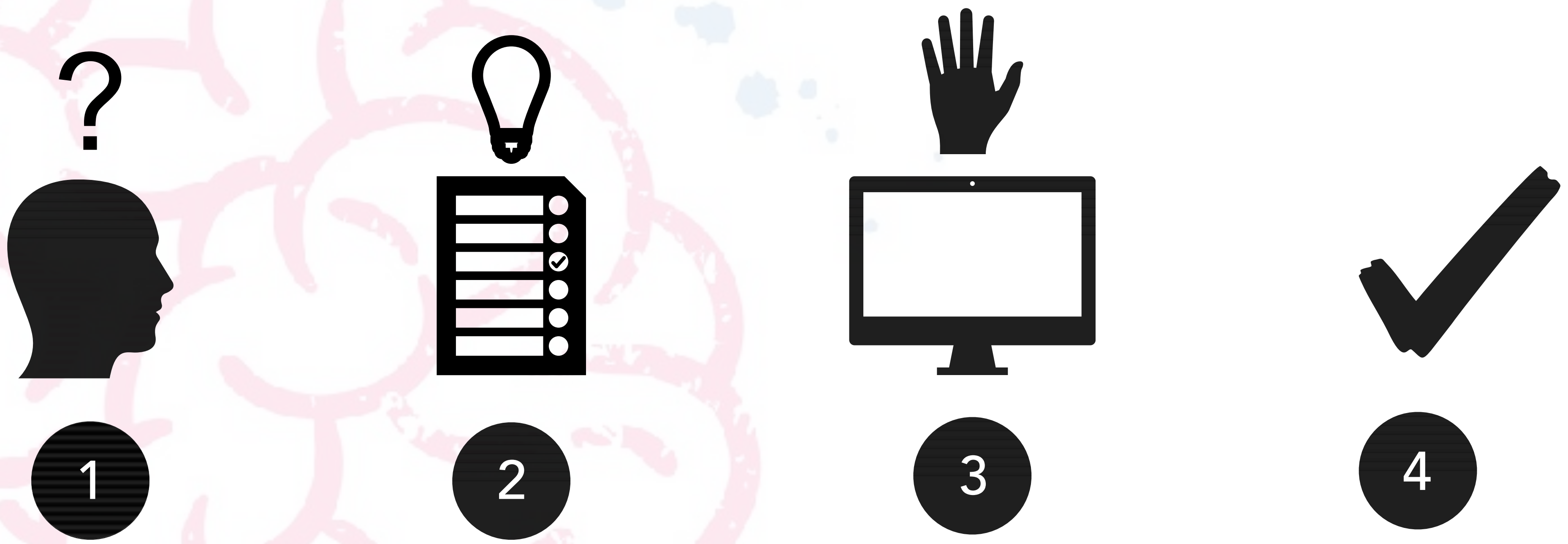


run -> no error -> push to your git repository

Exercise

Create main.go in folder chapter4-3 for solve

Problem No.6 of Chapter 4 :



run -> no error -> push to your git repository

Conditions

golang : Conditions

create main.go in folder chapter5-1 :

```go
package main

import "fmt"

func main() {
	fmt.Println(“1”)
	fmt.Println(“2”)
	fmt.Println(“3”)
	fmt.Println(“4”)
	fmt.Println(“5”)
	fmt.Println(“6”)
	fmt.Println(“7”)
	fmt.Println(“8”)
	fmt.Println(“9”)
	fmt.Println(“10”)

}
```

golang : Conditions [for]

```go
package main

import "fmt"

func main() {
      func main() {
      number := 1
      for number <= 10 {
      fmt.Println(number)
      number = number + 1
  }


}
```

golang : Conditions [if]

create main.go in folder chapter5-2 :

```go
package main

import "fmt"

func main() {
  for number := 1; number <= 100; number++ {
        if number%15 == 0 {
          fmt.Println(number, "FizzBuzz")
        } else if number%3 == 0 {
          fmt.Println(number, "Fizz")
        } else if number%5 == 0 {
          fmt.Println(number, "Buzz")
        } else {
          fmt.Println(number)
        }
    }
}
```

golang : Conditions [switch case]

create main.go in folder chapter5-3 :

```go
package main

import "fmt"

func main() {
    switch i {
        case 0:
          fmt.Println("Zero")
        case 1:
          fmt.Println("One")
        case 2:
          fmt.Println("Two")
        case 3:
          fmt.Println("Three")
        case 4:
          fmt.Println("Four")
        case 5:
          fmt.Println("Five")
        default:
          fmt.Println("Unknown Number")
       }
}
```

Exercise

create exercise.go in folder chapter5-4 :

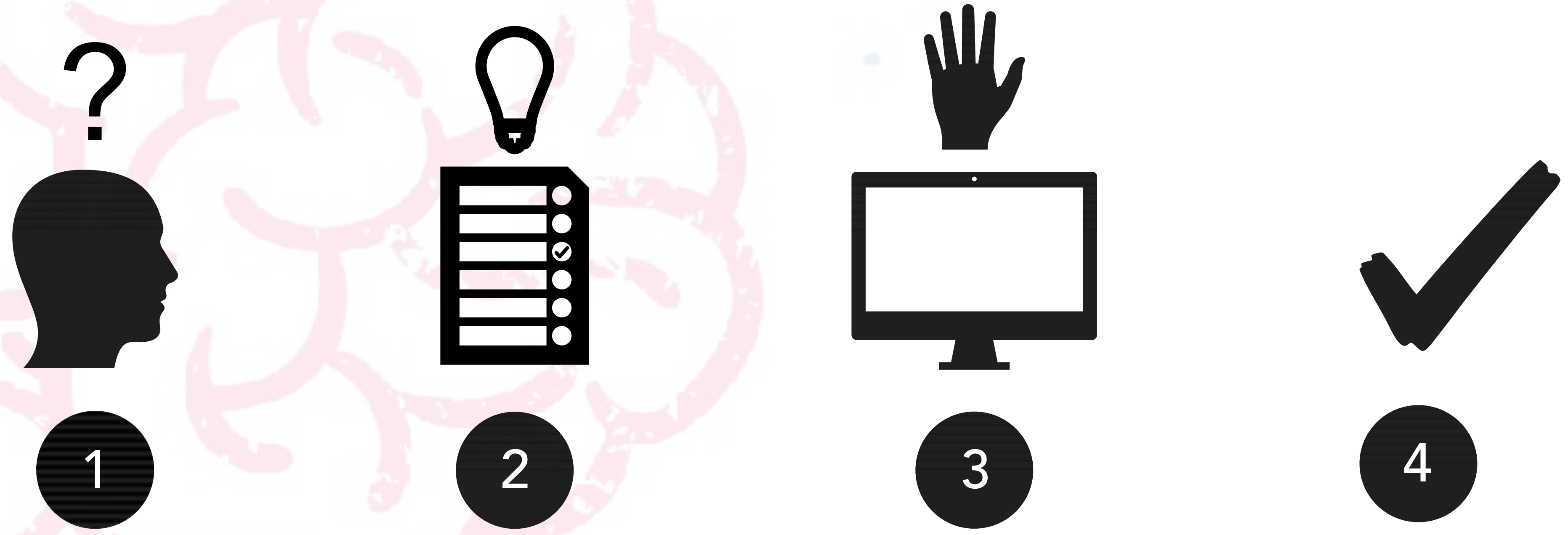
โปรแกรมจะให้ใส่ตัวเลขได้ไม่เกิน 5 ครั้ง

ถ้าเจอตัวเลขที่สุ่มมาจะแสดงคำว่า เจอแล้ว และจบการทำงาน

ถ้าเลขที่ใส่มากกว่าจะแสดงคำว่า มากไป

ถ้าเลขที่ใส่น้อยกว่าจะแสดงคำว่า น้อยไป

ถ้าใส่เกิน 5 ครั้งจะแสดงคำว่า เกินพอ และจบการทำงาน



run -> no error -> push to your git repository

func

Arrays, Slices

Maps

Pointers