Golang → Philosophy → Statically

Built-in conc

Fast compilaⁿ

G/C

Installaⁿ

run/build

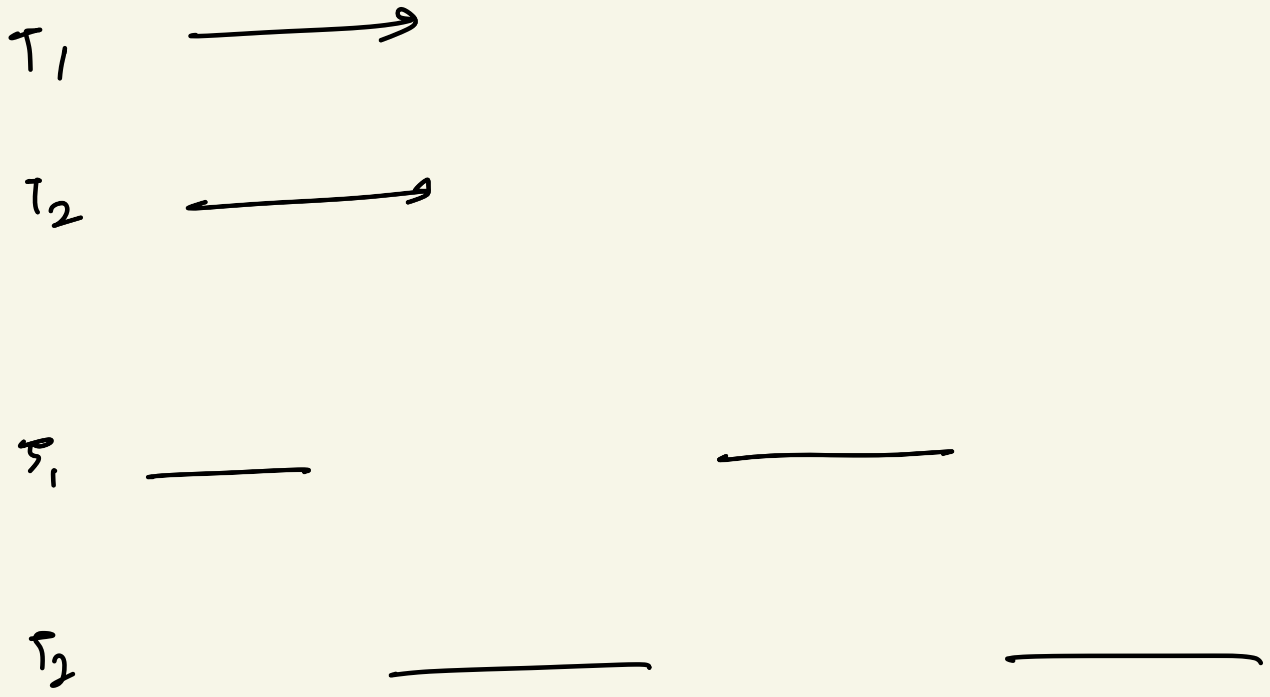
GoRoot v/s GoPath

First Go Code.

Module v/s Package.

Data types

Go	Nodejs	Java
	Event [illegible]	JVM


T_1
T_2
S_1
S_2

int float —— –

if else

switch

for

nil

⇒ unsigned ~~&~~ · int

byte v/s rune

$\updownarrow$

ASCII

Unicode

int v/s float

float

101.1234

① width |101.1234 | → 7

② Precision |___. 1234| → 4

%f

$$\underbrace{\%.\,\underset{\text{width}}{9}.\,\underset{\text{Precision}}{2}\,f}$$

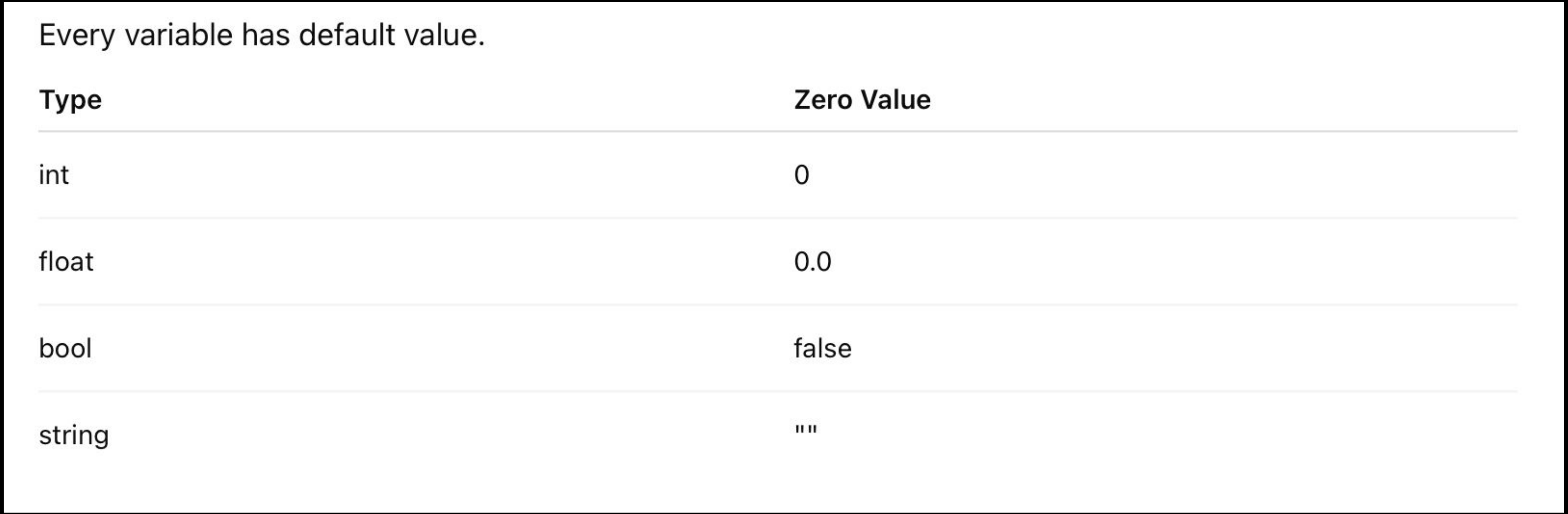
Golang-

Basic Types, Control flow statements

Go Fundamentals

- Values
  - *Int, float, boolean, string*
- Variables
  - *Basic declaration, short-hand declaration, type inference, block declaration, zero value*
- Constants
  - *Basic declaration, short-hand declaration, type inference, block declaration, zero value*
- For
  - *Classic for, while loop, range for loop, infinite loop*
- If/Else
  - *Basic if-else, statement if-else*
- Switch
  - *Basic switch, Without condition*

Zero Value Concept



Every variable has default value.

Type	Zero Value
int	0
float	0.0
bool	false
string	""

Go: What is the output?

```
x := 10


if x := 20; x > 15 {

    fmt.Println(x)

}


fmt.Println(x)
```

Output:

Answer

```
x := 10


if x := 20; x > 15 {
    fmt.Println(x)
}


fmt.Println(x)
```

```
Output: 20 10
```

Go: What is the output?

```
const x int

fmt.Println(x)
```

Output:

Remember

:=

1. := works only inside function
2. Go has only for loop
3. Switch has automatic break
4. Zero value exists
5. Constants cannot change and must be initialized at the time of declaration
6. Go is statically typed
7. Scope of variables declared inside if, for, or switch is limited to that block only
8. Short-hand declaration (:=) performs both declaration and initialization together

Type Conversion and Type Inference

float32( )

Float64( )

↳ Go automatically detec^n of data type

Summary

✓ int => A number WITHOUT decimal places (e.g., -5, 10, 12 etc)

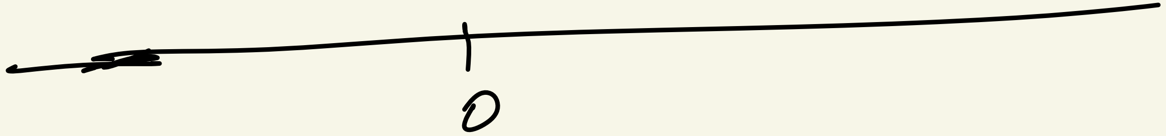
✓ float64 => A number WITH decimal places (e.g., -5.2, 10.123, 12.9 etc)

✓ string => A text value (created via double quotes or backticks: "Hello World", `Hi everyone')

✓ bool => true or false

uint => An unsigned integer which means a strictly non-negative integer (e.g., 0, 10, 255 etc) unsigned signed

256

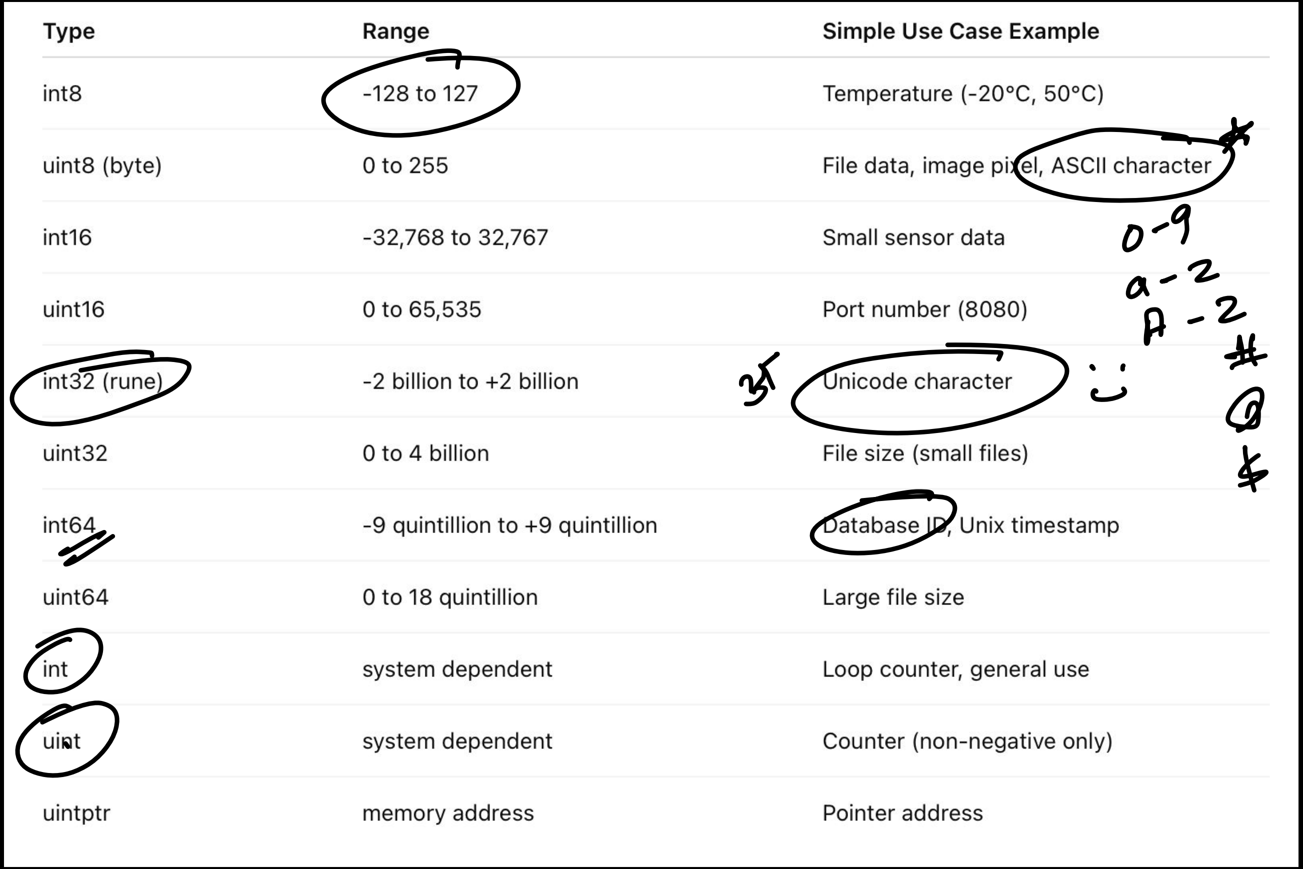


int8 + − − +

uint8 + + 0 − 255

Range Calculation: Simple Approach

Type: int

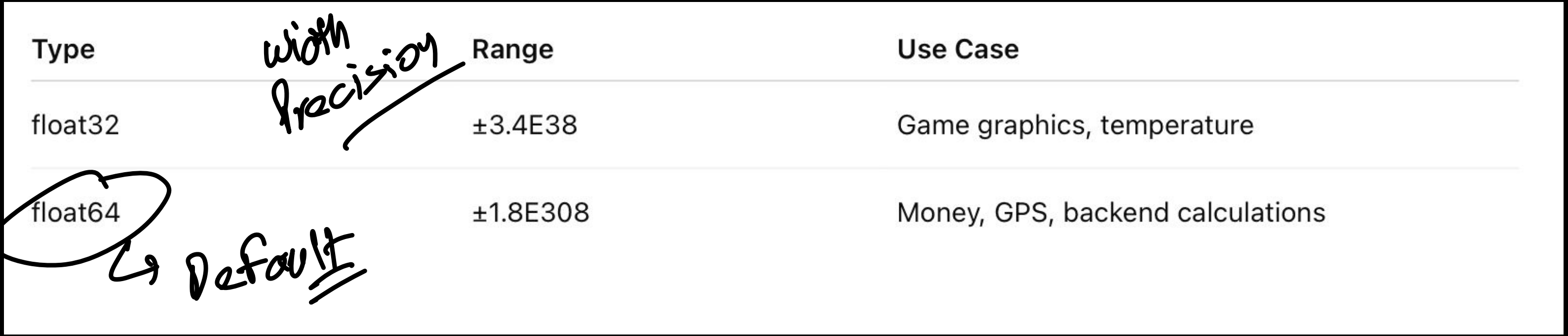


Type	Range	Simple Use Case Example
int8	-128 to 127	Temperature (-20°C, 50°C)
uint8 (byte)	0 to 255	File data, image pixel, ASCII character
int16	-32,768 to 32,767	Small sensor data
uint16	0 to 65,535	Port number (8080)
int32 (rune)	-2 billion to +2 billion	Unicode character
uint32	0 to 4 billion	File size (small files)
int64	-9 quintillion to +9 quintillion	Database ID, Unix timestamp
uint64	0 to 18 quintillion	Large file size
int	system dependent	Loop counter, general use
uint	system dependent	Counter (non-negative only)
uintptr	memory address	Pointer address

*
0-9
a-z
A-Z
#
@
$

ॐ

Float types:

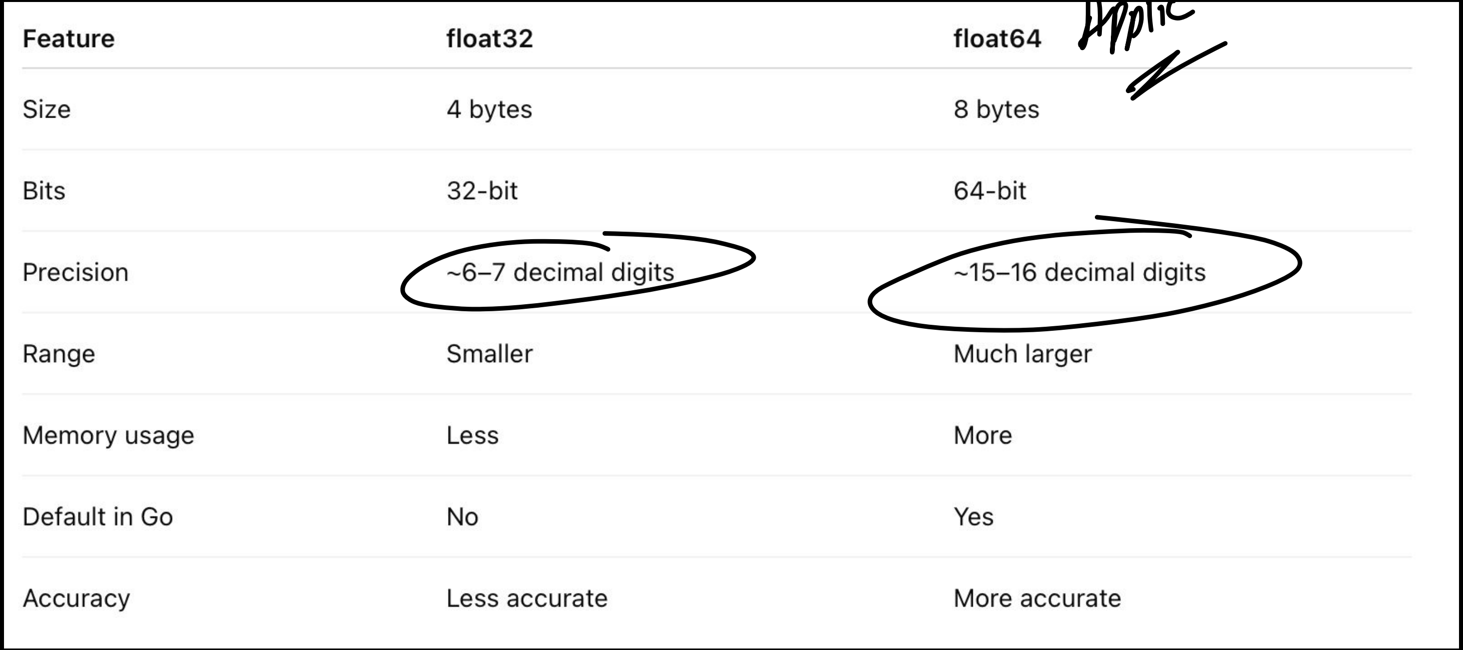


Type	Range	Use Case
float32	±3.4E38	Game graphics, temperature
float64	±1.8E308	Money, GPS, backend calculations

width Precision

float64 → Default

Float32 vs float64



Applic

Feature	float32	float64
Size	4 bytes	8 bytes
Bits	32-bit	64-bit
Precision	~6–7 decimal digits	~15–16 decimal digits
Range	Smaller	Much larger
Memory usage	Less	More
Default in Go	No	Yes
Accuracy	Less accurate	More accurate

Rune

What is Rune?

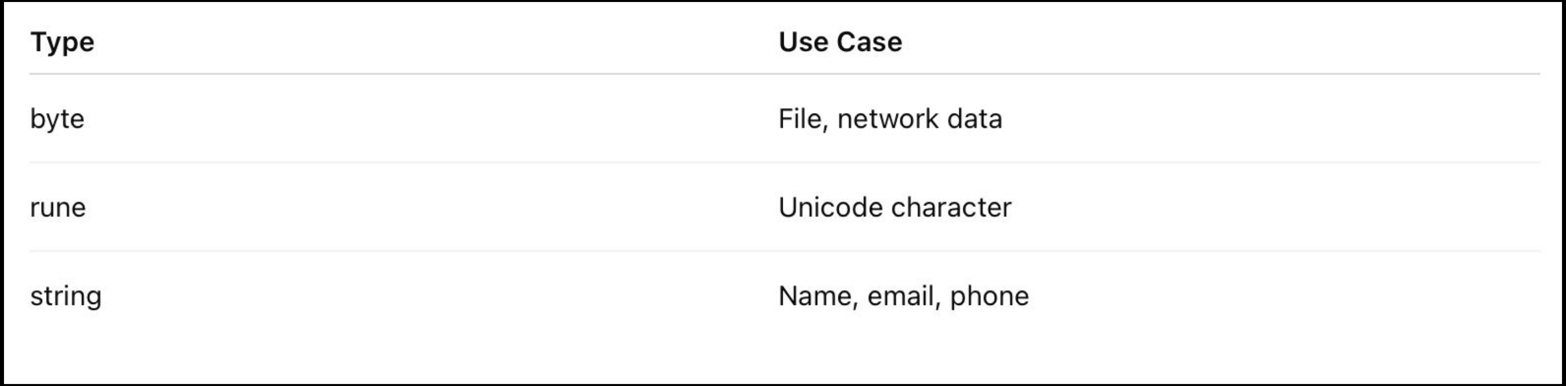
- Rune is built-in data type that represents a single Unicode code point

ASCII Character 0-9 a-z A-Z , $ @

Unicode Character ☺ अ

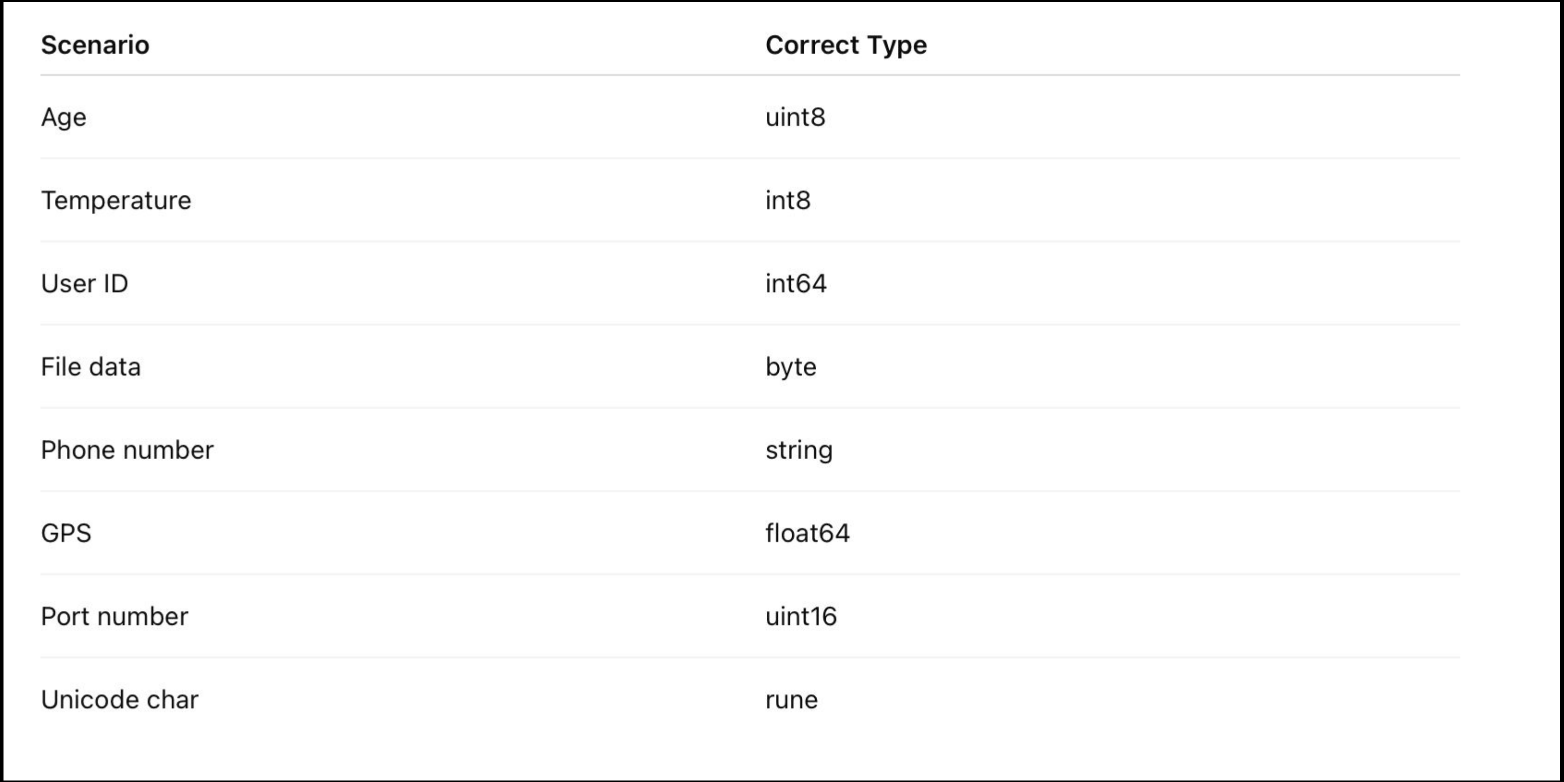
What are Unicode Characters?

Byte vs rune



Type	Use Case
byte	File, network data
rune	Unicode character
string	Name, email, phone

Summary



Scenario	Correct Type
Age	uint8
Temperature	int8
User ID	int64
File data	byte
Phone number	string
GPS	float64
Port number	uint16
Unicode char	rune

Practice Scenario: Database User ID

User ID: 9834567891

Answer:

Type: int64

Reason: Value is very large, exceeds int32 limit

Practice Scenario: Age Storage

Age range: 0 to 120

Answer:

Type: uint8

Reason: Age is never negative and fits in 0–255

Practice Scenario: GPS Location

Latitude: 19.076090

Answer:

Type: float64

Reason: GPS requires high precision

Practice Scenario: Bank Balance

Balance: 1000.75

Answer:

Type: float64

Reason: Decimal values present

Interview Questions

You are designing a backend system.

Which type is BEST for database User ID?

Options:

A. int8

B. uint32

[✓ C. int64]

D. uint8

Answer

Correct Answer: C. int64

Explanation:

Database BIGINT = signed int64

Industry standard for IDs

Main Function

- package main → the starting package of the application.
- func main → where program execution starts
- Why fmt package has no main function?
    - *Answer: it is a library, not an executable program.*
- Library : fmt (no main function required)
- Executable: Go Project (requires main to execute as program)

Libraries

Fmt

Math