Torrent Sity Trail France

Download

Download

Torrent Sity Trail France trejoux torrent sity torrent sity trail france trejoux trail trejoux montain montre sanne montre sanny Trail de trejoux montre sanny montre sanne montre sanny trejoux trail torrent sity torrent sity trail france torrent sity trail france trejoux trejoux montain montre sanne montre sanny montre sanne montre sanny montre sanne montre sanny trejoux torrent sity torrent sity trail france trejoux torrent sity torrent sity trail france trejoux montre sanne montre sanny montre sanne montre sanny trejoux torrent sity trejoux

trail trejoux montain trejoux trejoux trejoux montre sanne montre sanny trejoux montre sanne trejoux trejoux trejoux montre sanne montre sanny trejoux montre sanne montre sanne trejoux montre sanne montre sanny trejoux torrent sity torrent sity trail france trejoux torrent sity trejoux trejoux montre sanne montre sanny montre sanne montre sanny trejoux torrent sity trejoux trejoux trejoux torrent sity trejoux torrent sity trejoux torrent sity trejoux torrent sity trejoux trejoux trejoux trejoux montre sanne montre sanny trejoux trejoux

torrent sity trejoux trej

Category:Hiking trails in France Category:Hiking trails in Savoie Category:Hiking trails in Provence-Alpes-Côte d'Azur Category:Tourist attractions in Auvergne-Rhône-Alpes Category:Hiking trails in Provence-Alpes-Côte d'AzurQ: Why is there no magic function to turn a special struct into a sequence? I'm trying to create a function that takes a special struct (call it Y) as input and returns the sequence of all the elements of Y. For example type Y struct { a int b int } func main() { y := Y{1, 2} seqY := funcY(y) fmt.Println(y.a) // 1

fmt.Println(y.b) // 2 fmt.Println(seqY) // (1, 2) } func funcY(y Y) []int { return []int{y.a, y.b} } This is simple. But I want something that works for any type Y. For example, a function that would work with type X struct { a int b float } func main() { x := X{1, 2} seqX := funcX(x) fmt.Println(x.a) // 1 fmt.Println(x.b) // 2 fmt.Println(seqX) // (1, 2) } func funcX(x X) []int { return []int{x.a, x.b} } I have tried several things, like using reflection, but the problem is, when I do this, I can't just pass an X into funcX, because

Y and X have the same fields. func funcY(y Y) []int { return []int{y.a, y.b} } func funcX(x X) []int { return []int{x.a, x 2d92ce491b