**Hello World**

package main

import “fmt”

func main() {

fmt.Println(“Hello World”)

}

**Packages**

**Executable packages**

Debe tener el package main, crea un ejecutable

Debe tener una función llamada main

**Reusable packages**

No debe ser main

**Import**

fmt es una librería standard de go (format)

Se pueden utilizar librerías que hagan acciones específicas (como en cualquier lenguaje)

**Variables**

Go es de tipado dinámico, o sea, no cambia el tipo de dato según la asignación

Inicialización

card := “Ace of spades”

Se usa := para inicializar un valor

card = “Five of diamonds”

Solo se usa = para reasignar un valor

Se pueden declarar variables fuera de una función, pero no se pueden asignar afuera de una función

**Functions**

func main() {

something := nameOfFunction()

}

func nameOfFunction() type {

// body

return VariableOftype

}

**Arrays, slices and for loops**

Array es un tipo que solo contiene elementos y ya, no se pueden agregar elementos a un array (quizás parecidos a tuplas en Python)

Slice es un tipo que puede ser modificado, es decir, agregar elementos, etcétera

**Declarar slice**

slice := []type{“element 1”, “element 2”}

**Para agregar elementos al slice**

slice = **append**(slice, “element 3”)

**For en go**

for index, element := range slice {

// body, iterate through slice items

}

**Custom type**

deck.go

package main

import “fmt”

type deck []string //Creating deck type as []string

func (d deck) print() { //Declaring a function for the deck type

for i, card := range d {

fmt.Println(i, card)

}

}

// Receiver, si se tiene un receiver, las variables de tipo ‘deck’ pueden usar este método, similar a this en Javascript/Java o self en P ython

main.go

package main

func main() {

card := deck{“First”}

card = append(card, “Second”)

card.print() //Implementing function

}

**More about slices**

Slicing slices

Basicamente es lo mismo que Python

deck[index incluyente:index no incluyente]

deck[:index no incluyente]

deck[index incluyente:]

**Standard packages (**[**https://pkg.go.dev/std**](https://pkg.go.dev/std)**)**

ioutil

strings

**Cast**

Ejemplo

[]byte(“Hi there”)

Transforma una string en un byte slice

Nota: En go se deben utilizar todas las variables que se declaren, no se pueden quedan variables sin usar, a menos que se declaren como guión bajo (\_)

Para regresar más de un valor en una función se necesita poner en la firma con paréntesis, es decir

func deal(d deck, handSize int) **(deck, deck)** {

// body

}

Se pueden importer más de un package con paréntesis ejemplo:

import (

"fmt"

"strings"

)