Introduction to Golang
Part 3

Calistus Igwilo
https://linkedin.com/in/calistus-igwilo
https://twitter.com/CalistusIgwilo

ASCII & Unicode

- American Standard Code for Information Interchange
- Character coding - each character is associated with a 7 (8) bit number
  - "A" = 0x41
- This is sufficient for the English alphabets

ASCII & Unicode

- Unicode is a 32bit Character code
- UTF-8 is variable length（can go from 8 to 32 bits）
  - 8bit UTF code are same as ASCII
- In Go, the default is UTF-8

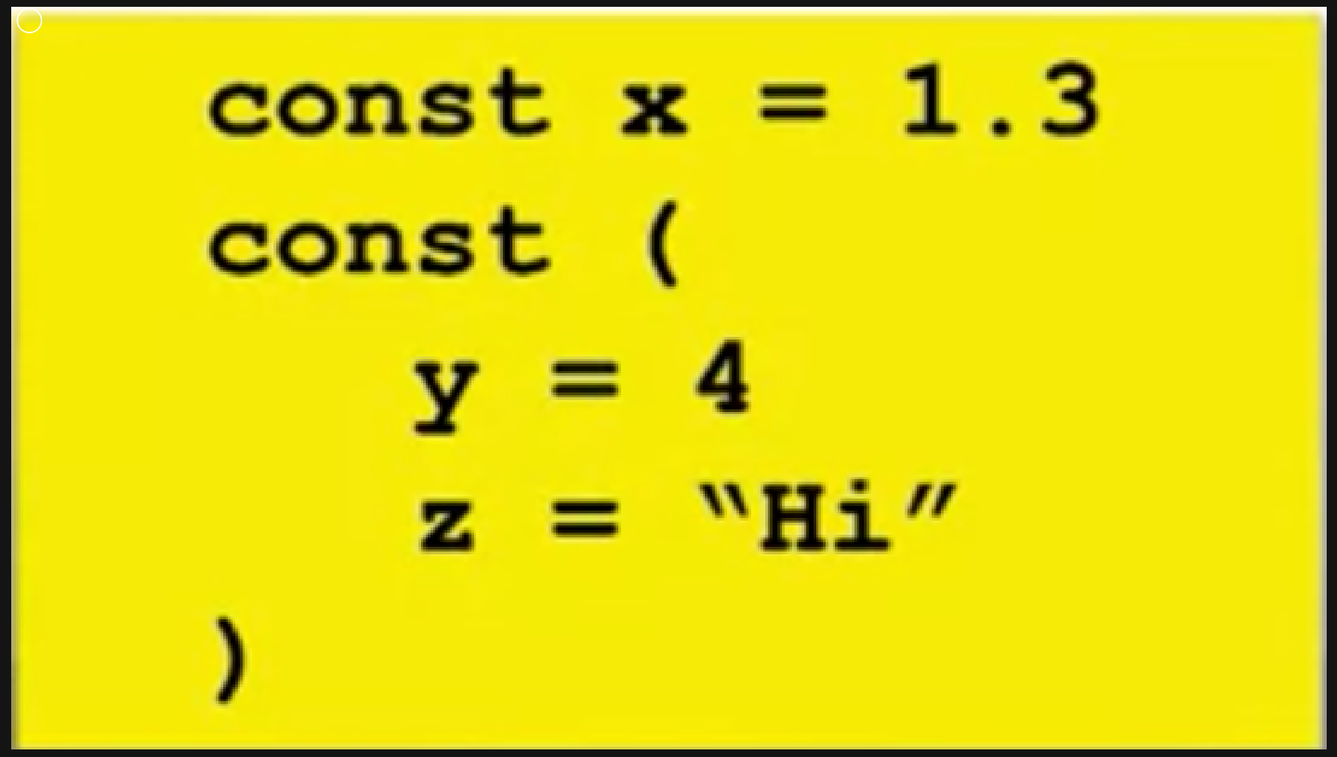
Strings

- **Arbitrary sequence of bytes represented in UTF-8**
  - **Read-only**
  - Often meant to be printed

- **String literals: denoted by double quotes**
  - **x := "Hi there"**
- Each byte is a rune (UTF-8 code point)

Constants

- Expression whose value is known at runtime
- Type is inferred from right hand side



Control Structures

Statements which alter control flows

- ```
  if <condition> {
      <statement>
  }
  ```
  ```

Control Structures

- Expression <condition> is evaluated
  - <statements> are evaluated if condition is true
  - if (y > 0) {
  - fmt.Printf("Positive")
  - }

For Loop

- Iterates while condition is true
- May have initialization and update operation
    - for <init>; <condition>; <update> {
    - statements
    - }

For Loop Forms

```
for i:=0; i<10; i++ {
    fmt.Printf("hi ")
}
```

```
i = 0
for i < 10 {
    fmt.Printf("hi ")
    i++
}
```

```
for {
    fmt.Printf("hi ")
}
```

Break

break exists the containing loop

```
i := 0
for i < 10 {
    i++
    if i == 5 { break }
    fmt.Printf("hi ")
}
```

Continue

continue skips the rest of the current iteration

```
i := 0
for i < 10 {
    i++
    if i == 5 { continue }
    fmt.Printf("hi ")
}
```

Scan

- Reads user input
- Takes a pointer as an argument
- Typed data is written to a pointer

```
fmt.Printf("Number of
apples?")

num, err :=
fmt.Scan(&appleNum)

fmt.Printf(appleNum)
```

Exercise 1

Write a program which prompts the user to enter a floating point number and prints the integer which is a truncated version of the floating point number that was entered. Truncation is the process of removing the digits to the right of the decimal place.

Solution 1

```go
package main

import "fmt"

var floatNum float64

func main(){
    fmt.Println("Enter a float value : ")
    fmt.Scanf("%f", &floatNum)
    //fmt.Printf("Truncated float: %.0f ", floatNum)  This line rounds
    fmt.Printf("Truncated float number: %d", int(floatNum))  // correct

}
```