Basic types

Go's basic types are

```
bool

string

int  int8  int16  int32  int64
uint uint8 uint16 uint32 uint64 uintptr

byte // alias for uint8

rune // alias for int32
     // represents a Unicode code point

float32 float64

complex64 complex128
```

information above is from is from https://tour.golang.org/basics/11

```
//this code is achieved with golang 1.17
package main

import (
    "encoding/binary"
    "fmt"
)

type dataRange struct {
    max interface{}
    min interface{}
}

func main() {
    dataTypes:=  []dataRange{
        //bool
        {max:true,min: false},
        //String in golang has no length limit and size is not fixed
        {max:"{Memory Size}",min:""},
        //int in 64bit system is int64, an int32 in 32bit system, size is not
fixed
        {max:1<<63 - 1,min:-1<<63},
        //int8
        {max:int8(1<<7-1),min:int8(-1<<7)},
        //int16
        {max:int16(1<<15-1),min:int16(-1<<15)},
        //int32
        {max:int32(1<<31-1),min:int32(-1<<31)},
        //int64
        {max:int64(1<<63-1),min:int64(-1<<63)},
        //uint in 64bit system is uint 64, an uint32 in 32bit system, size is
not fixed
```

```
        {max:uint(1<<64 - 1),min: uint(0)},
        //uint8
        {max:uint8(1<<8 - 1),min: uint8(0)},
        //uint16
        {max:uint16(1<<16 - 1),min: uint16(0)},
        //uint32
        {max:uint32(1<<32 - 1),min: uint32(0)},
        //uint64
        {max:uint64(1<<64 - 1),min: uint64(0)},
        //uintptr has the range of uint64 in 64bit system, has the range of uint
32 in 32bit system, size is not fixed
        {max:uintptr(1<<64-1),min: uintptr(0)},
        //byte in golang is actually just alias to uint8
        {max:byte(1<<8 - 1),min: byte(0)},
        //rune in golang is actually just alias to int32
        {max:rune(1<<31-1),min:rune(-1<<31)},
        //float32
        {max:float32(0x1p127 * (1 + (1 - 0x1p-23))),min:float32(-0x1p127 * (1 +
(1 - 0x1p-23)))},
        //float64
        {max:float64(0x1p1023 * (1 + (1 - 0x1p-52))),min:float64(-0x1p1023 * (1
+ (1 - 0x1p-52)))},
        //complex64
        {max:complex64(complex(float32(0x1p127 * (1 + (1 - 0x1p-23))),0x1p127 *
(1 + (1 - 0x1p-23)))),min:complex64(complex(float32(-0x1p127 * (1 + (1 - 0x1p-
23))),-0x1p127 * (1 + (1 - 0x1p-23))))},
        //complex128
        {max:complex128(complex(float64(0x1p1023 * (1 + (1 - 0x1p-
52))),float64(0x1p1023 * (1 + (1 - 0x1p-
52))))),min:complex128(complex(float64(-0x1p1023 * (1 + (1 - 0x1p-
52))),float64(-0x1p1023 * (1 + (1 - 0x1p-52)))))},
    }
    for _,dataType := range dataTypes{
        fmt.Printf("Type: %10T Size: %2d Value: %30v to %30v \n",
dataType.max,binary.Size(dataType.max), dataType.min, dataType.max)
    }
}
```

plain text code below

package main

import (
  "encoding/binary"
  "fmt"
)

type dataRange struct {
  max interface{}
  min interface{}
}

```
func main() {
    dataTypes:=  []dataRange{
        {max:true,min: false},
        {max:"{Memory Size}",min:""},
        {max:1<<63 - 1,min:-1<<63},
        {max:int8(1<<7-1),min:int8(-1<<7)},
        {max:int16(1<<15-1),min:int16(-1<<15)},
        {max:int32(1<<31-1),min:int32(-1<<31)},
        {max:int64(1<<63-1),min:int64(-1<<63)},
        {max:uint(1<<64 - 1),min: uint(0)},
        {max:uint8(1<<8 - 1),min: uint8(0)},
        {max:uint16(1<<16 - 1),min: uint16(0)},
        {max:uint32(1<<32 - 1),min: uint32(0)},
        {max:uint64(1<<64 - 1),min: uint64(0)},
        {max:uintptr(1<<64-1),min: uintptr(0)},
        {max:byte(1<<8 - 1),min: byte(0)},
        {max:rune(1<<31-1),min:rune(-1<<31)},
        {max:float32(0x1p127 * (1 + (1 - 0x1p-23))),min:float32(-0x1p127 * (1 + (1 - 0x1p-23)))},
        {max:float64(0x1p1023 * (1 + (1 - 0x1p-52))),min:float64(-0x1p1023 * (1 + (1 - 0x1p-52)))},
        {max:complex64(complex(float32(0x1p127 * (1 + (1 - 0x1p-23))),0x1p127 * (1 + (1 - 0x1p-
23)))),min:complex64(complex(float32(-0x1p127 * (1 + (1 - 0x1p-23))),-0x1p127 * (1 + (1 - 0x1p-
23))))},
        {max:complex128(complex(float64(0x1p1023 * (1 + (1 - 0x1p-52))),float64(0x1p1023 * (1 + (1 -
0x1p-52))))),min:complex128(complex(float64(-0x1p1023 * (1 + (1 - 0x1p-52))),float64(-0x1p1023 *
(1 + (1 - 0x1p-52)))))},
    }
    for _,dataType := range dataTypes{
        fmt.Printf("Type: %10T Size: %2d Value: %30v to %30v \n",
dataType.max,binary.Size(dataType.max), dataType.min, dataType.max)
    }
}
```