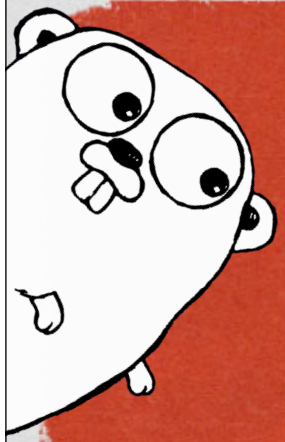


# GO TUTORIAL WRAP-UP

Curt Clifton
Rose-Hulman Institute of Technology

# CORRECTIONS AND CLARIFICATIONS

```
func getHandle(t *int) **int {
    th := new(*int)
    *th = t
    return th
}
```

th := &t

```
// Omit parameter names
func foo(int, string) (float, float) {
    return 2.13, 8.15
}
```

```
func main() {
    // Pointers and handles
    x := new(int)
    *x = 42
    xh := getHandle(x)
    fmt.Println("xh:", xh)
    fmt.Println("*xh:", *xh)
    fmt.Println("**xh:", **xh)

    // ----------------------------------------
    a, b := foo(42, "hello")
    fmt.Println("foo result:", a, b)
}
```

# CORRECTIONS AND CLARIFICATIONS

```
func getHandle(t *int) **int {
    th := new(*int)
    *th = t
    return th
}


// Omit parameter names
func foo(int, string) (float, float) {
    return 2.13, 8.15
}
```

```
func main() {
    // Pointers and handles
    x := new(int)
    *x = 42
    xh := getHandle(x)
    fmt.Println("xh:", xh)
    fmt.Println("*xh:", *xh)
    fmt.Println("**xh:", **xh)


    // ------------------------------------------------
    a, b := foo(42, "hello")
    fmt.Println("foo result:", a, b)
}
```

# CORRECTIONS AND CLARIFICATIONS

```
func bar(a int) (abs int) {
    if a < 0 {
        abs = -a
    } else {
        abs = a
    }
    fmt.Println("abs: ", a)
    return
}
```

Named return parameter

Can omit expressions

Photo by Tambako the Jaguar - http://flic.kr/p/1swko

# CAT: USING INTERFACES

SVN Update and Open *GoIntro2/cat.go*

# POLYMORPHISM VIA INTERFACES

Photo by Ajith (□□□□□□□□) - http://flic.kr/p/8KBtfV

```
// Insertion sort
func Sort(data Sortable) {
    for i := 1; i < data.Len(); i++ {
        for j := i; j > 0 && data.Less(j, j-1); j-- {
            data.Swap(j, j-1)
        }
    }
}
```

*sort* package calls this *sort.Interface*

```
// Insertion sort
func Sort(data Sortable) {
    for i := 1; i < data.Len(); i++ {
        for j := i; j > 0 && data.Less(j, j-1); j-- {
            data.Swap(j, j-1)
        }
    }
}

// Minimum method set for sorting
type Sortable interface {
    Len() int
    Less(i, j int) bool
    Swap(i, j int)
}
```

An interface type is a set of methods

```
// Minimum method set for sorting
type Sortable interface {
    Len() int
    Less(i, j int) bool
    Swap(i, j int)
}

// Need to define a local alias for []int so we can add methods
type IntArray []int

func (p IntArray) Len() int {
    return len(p)
}

func (p IntArray) Less(i, j int) bool {
    return p[i] < p[j]
}

func (p IntArray) Swap(i, j int) {
    p[i], p[j] = p[j], p[i]
}
```

Q2

```
type IntArray []int

func (p IntArray) Len() int {
    return len(p)
}

func (p IntArray) Less(i, j int) bool {
    return p[i] < p[j]
}

func (p IntArray) Swap(i, j int) {
    p[i], p[j] = p[j], p[i]
}

func main() {
    data := []int{42, 2, 13, 1024, 8, 15}
    a := IntArray(data)
    Sort(a)
    fmt.Println(data)
}
```

Q3

# STRINGER

Canonical name

```
type Stringer interface {
   String() string
}


// fmt.Stringer
func (fn FullName) String() string {
   return fn.firstName + " " + fn.lastName
}
```

Q4

# STRINGER

```
type Stringer interface {
    String() string
}
```

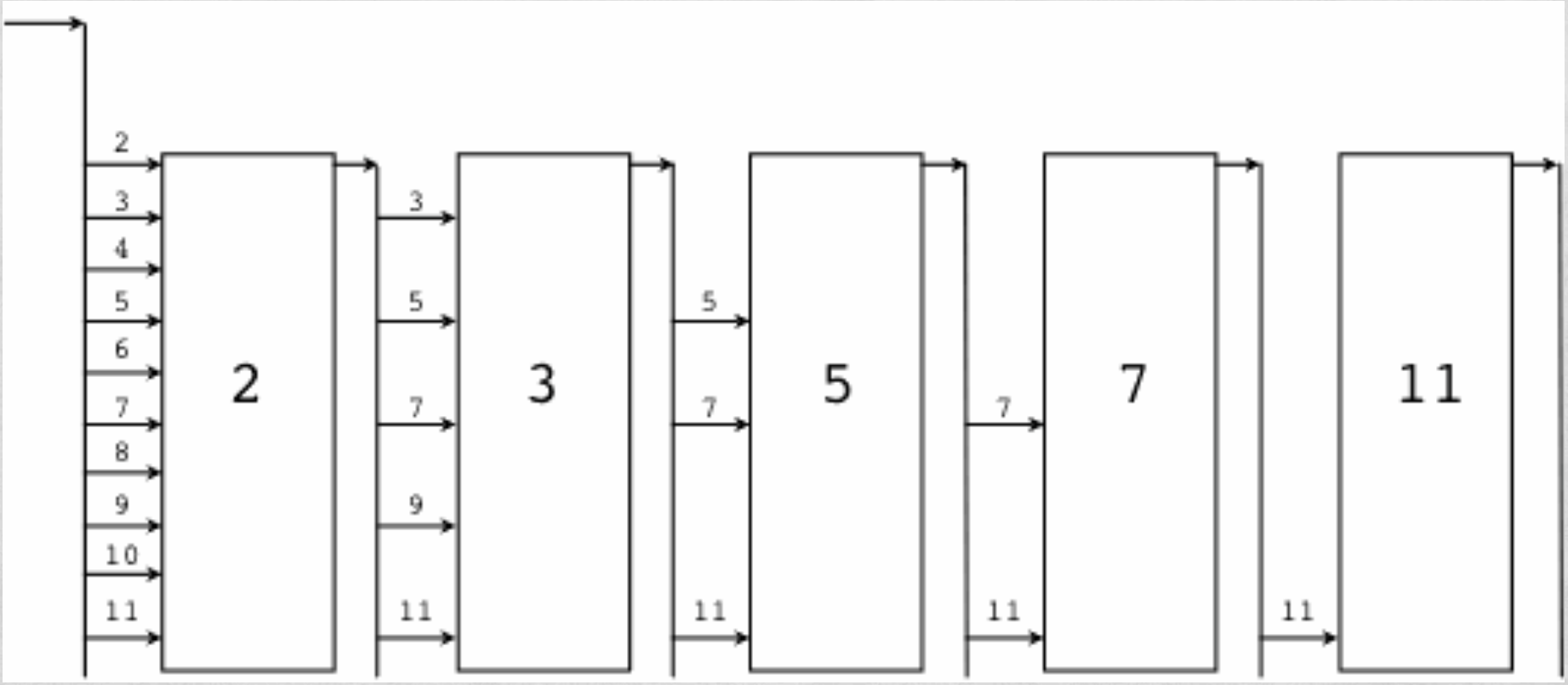
```
// fmt.Stringer
func (fn FullName) String() string {
    return fn.firstName + " " + fn.lastName
}
```

```
// Inside fmt package
s, ok := v.(Stringer)
if ok {
    result = s.String()
} else {
    result = defaultOutput(v)
}
```


Photo by davidgsteadman - http://flic.kr/p/73utgP

# SIEVE!

# PRIME NUMBER SIEVE




http://golang.org/doc/go_tutorial.html

# CHANNELS

- Connect two concurrent computations

Photo by coolmonfrere - http://flic.kr/p/jkU4F

# GOROUTINES

```
func generate(out chan int) {
    for i:=2; ; i++ {
        out <- i
    }
}

func main() {
    ch := make(chan int)
    go generate(ch)
    for {
        fmt.Print(<-ch, " ")
    }
}
```

Send *i* over channel

- Concurrently executing computations
- Run in parallel
- Share the same heap (be careful!)

Go Go Gadget! Generate!

Receive value from channel