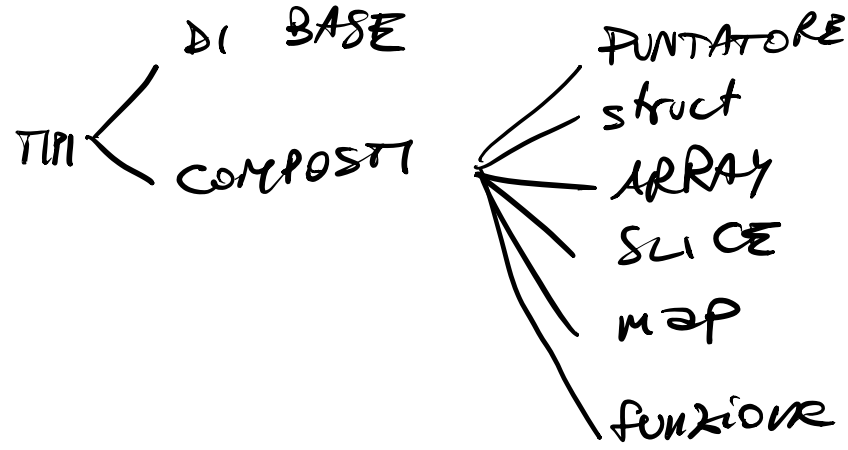
TIPI FUNZIONALI



SIGNATURA

```
func Pippo (x int, y float64) (b bool) {
    ...
}
func Pluto (a int, lung float64) bool {
    ...
}
type f func (int, float64) bool;
```

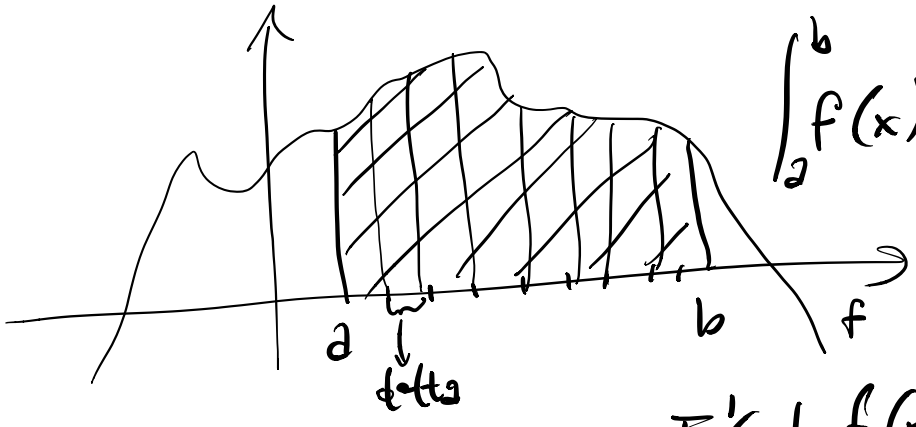
```
func main {
    var x f
    if os.Args[1] == "w" {
        x = Pippo
    } else {
        x = Pluto
    }
    b := x(5, 3.7)
    fmt.Println(b)
}
```

```
type funz func(float64) float64

func main() {
    var f funz
    if os.Args[1] == "s" {
        f = math.Sin
    } else {
        f = math.Cos
    }
    v, _ := strconv.Atof(os.Args[2])
    ris := f(v)
    fmt.Println(ris)
}
```

./pippo c 0.3

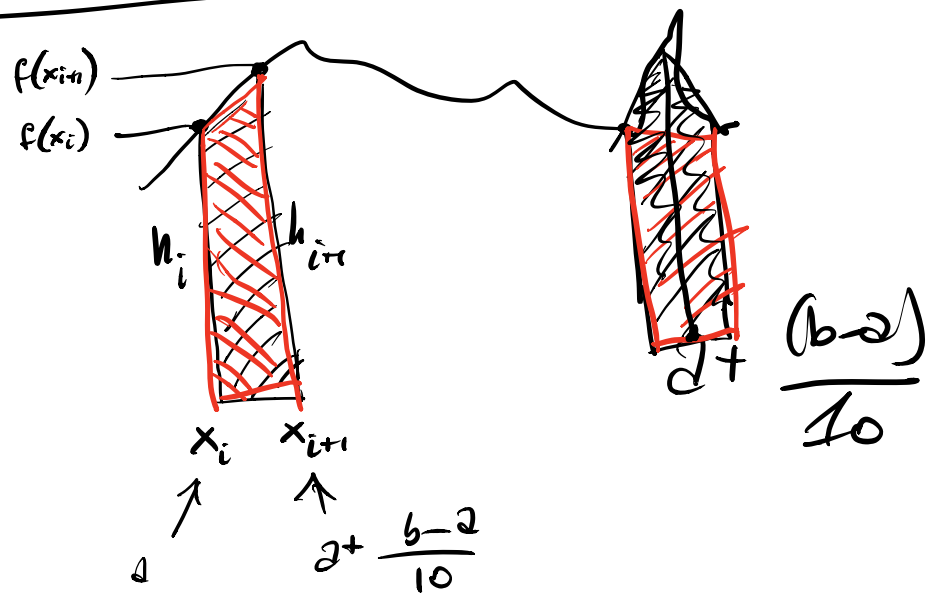
INTEGRAZIONE NUMERICA



$$\int_a^b f(x)dx = F(b)-F(a)$$

F $\qquad F'(x)=f(x)$

METODO DEI TRAPEZOI DI



$$x_i = a + i\,delta$$

type funz $\qquad$ func (float64) float64

```
func integrale Trap (f funz, a,b float64, n int)
                    float64 {
    delta := (b-a) / float64(n)
    area := 0
    for i:=0; i<n; i++ {
        xi := a + i * delta
        xipu := a+(i+1) * delta
        hi := f(xi)
        hipu := f(xipu)
        atrap := (hi + hipu) * delta /2
        area += atrap
    }
    return area
}

func main() {
    fmt.Println(integrale Trap(math.Sin, 3.7,
                100))
}
```

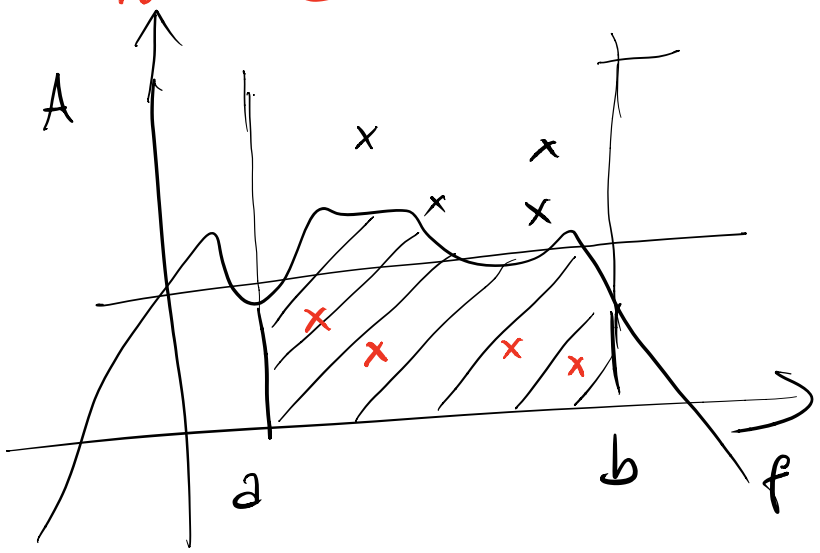
$$\int_3^7 \sin(x)\,dx = \left[-\cos(x)\right]_3^7 =$$

$$= \cos 3 - \cos 7$$

$\rightarrow x^2 + \sin(x^3) - \ln(x)$

```
fmt.Println(integrateTrap(
    λ [ func(x float64) float64 {
            return x*x + math.Sin(x*x*x) - math.Log(x)
        }, 3, 7, 100)
```

METODO DI INTEGRAZIONE MONTE-CARLO



$A \cdot (b-a)$

ESERCIZIO

- Scrivete degli unit test per entrambi i tipi di integrazione.

I test unitari devono sfruttare la vostra abilità di integrazione simbolica.

```
type testCase struct {
    f      func
    fprim. func
    a, b   float64
}
```

ESERCIZIO

1) Data una slice di interi, trovare se ci sono valori ripetuti.

```
func rip(x []int) bool {
    for i := 0; i < len(x); i++ {
        for j := 0; j < len(x); j++ {
            if i != j {
                if x[i] == x[j] {
                    return true
                }
            }
        }
    }
}
```

```
    return false
}

func rip(x []int) bool {
    var m map[int]bool
    for i:=0; i< len(x); i++ {
        _, ok := m[x[i]]
        if ok {
            return true
        }
        m[x[i]] = true
    }
    return false
}
```