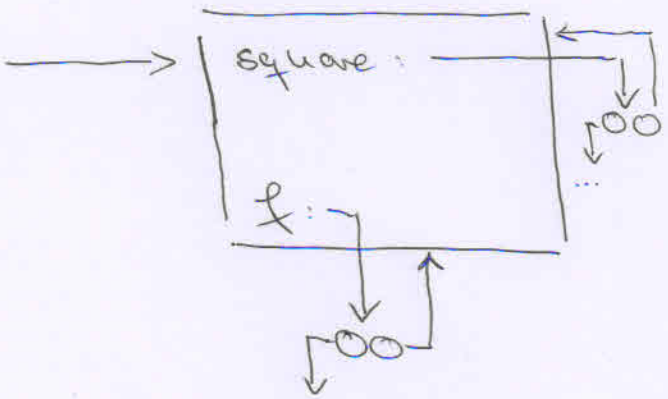
§2 Więcej o funkcjach

Funkcje jako argumenty i wyniki funkcji

Przykład:

```
(define (f g x) (g (+ x 1)))
```




```
> (f square 3)
=> (square (+ 3 1))
=> (square 4)
=> (* 4 4)
=> 16

> (f 5 3)
*=> (5 4)
```

↯ "5 nie jest funkcją!"

Funkcje jako "schematy"

```
(define (sum-integers a b)
  (if (> a b)
      0
      (+ a (sum-integers (+ a 1) b))))
```

$$\frac{1}{8}\pi = \frac{1}{1\cdot 3} + \frac{1}{5\cdot 7} + \frac{1}{9\cdot 11} + \cdots$$

```
(define (sum-pi a b)
  (if (> a b)
      0
      (+ (/ 1 (* a (+ a 2)))
         (sum-pi (+ a 4) b))))
```

```
(define (sum term next a b)
  (if (> a b)
      0
      (+ (term a)
         (sum term next (next a) b))))
```

a więc

```
(define (plus1 n) (+ n 1))
(define (3d n) n)

(define (sum-integers a b) (sum 3d plus1 a b))
```

albo lepiej

```
(define (sum-pi a b)
  (sum (lambda (x) (/ 1 (* x (+ x 2))))
       (lambda (x) (+ x 4))
       a
       b))
```

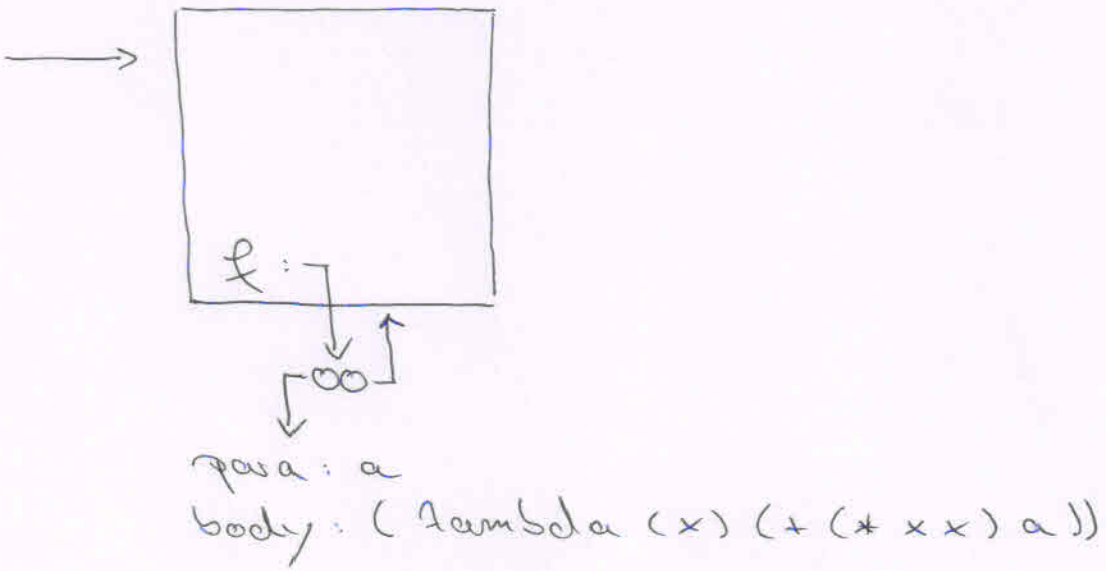
lub

```
(define (sum-pi a b)
  (let ((pi-term (lambda (x) (/ 1 (* x (+ x 2)))))
        (pi-next (lambda (x) (+ x 4))))
    (sum pi-term pi-next a b)))
```

$f_a(x) = x^2 + a$

$\longrightarrow (f\ a)$ "=" f_a

```
(define (f a)
  (lambda (x) (+ (* x x) a)))
```




```
> (f 5)
=> (lambda (x) (+ (* x x) 5))

> ((f 5) 3)
=> ((lambda (x) (+ (* x x) 5)) 3)
=> (+ (* 3 3) 5)
*=> 14
```

$$Df(x) = \frac{f(x+dx) - f(x)}{dx}$$

czyli (deive f dx) = Df

dla danego f i dx

```
(define (deive f dx)
  (lambda (x)
    (/ (- (f (+ x dx))
          (f x))
       dx)))
```

Więc 2

```
(define (cube x) (* x x x))
```

dostaniemy

```
> ((deive cube 0.001) 5)
75,015
```