; Abstracting and Naming a Process.

; Abstracting/Generalizing from Specific Examples/Cases/Instances.
;
; A specific concrete example has parts that are general, and
; parts that are specific to that example. In multiple examples
; the general parts don't change, but the specific parts do.
;
; Abstract by using names where the specific parts are.

; To abstract the process that we applied to a giraffe image and
; a cat image, let's use the name an-image in the place where
; earlier we had a specific image or the name of a specific image.
;
; (above (beside an-image (flip-horizontal an-image))
;    (beside (flip-vertical an-image)
;      (flip-horizontal (flip-vertical an-image))))

; Naming the Process in Racket.
;
; Rather than naming a value [image, number, string of text, truth],
; we're naming a function expecting a certain number of arguments.
; Make up a name for the function [here we'll use 'kaleidoscope'],
; and define the form in which it will be used:
;
; (define (kaleidoscope an-image)
;  ..)

; In Literate terms we're defining a new verb and associated verb clause.
; In Math and CS terms: our own function [or operation or procedure]
; having one parameter/place-holder [or taking one argument].

; Fill in the the meaning of the clause, in terms of the parameter name.
(requires picturing-programs)

(define (kaleidoscope an-image)
  (above
    (beside an-image
      (flip-horizontal an-image))
    (flip-vertical
      (beside an-image
        (flip-horizontal an-image)))))

; Now kaleidoscope a lot of images!

(kaleidoscope (giraffe an-image))
(define pandas (kaleidoscope
(scale 1/2 (kaleidoscope pandas))
(scale 1/4 (kaleidoscope (kaleidoscope pandas))))
(define triangles (kaleidoscope (rotate 40 (triangle 25 "solid" "aquamarine"))))
(kaleidoscope triangles)
(kaleidoscope (kaleidoscope triangles))