; Lists, Part IV: Applying
; ====================

; Now that we can produce a list of lots of things, how can we combine
; those things into one thing?

(require picturing-programs)
; We made the following list:

(list)
; How do we make an image with all of those images beside each other?

; The following is an error [why?]: (beside (list ))
; The function ‘beside’ has contract
; images -> image

; but (list ) is not seven images, it's one list of images.

; The binary function ‘apply’ takes a function and a list.
;
; apply : function list -> any
;
; It uses the function once, using the items in the list as all the arguments.

; EXAMPLE.
; Using ‘apply’ to put the images from a list beside each other as one image.
; Shows the intermediate steps.

(apply beside (list ))

(beside )
; Unfortunately, the Stepper doesn't show that step.
;
; Summary of the result:

(check-expect (apply beside (list . )))

; Intermediate Step
; -----------------
; The intermediate step for 'apply' [which isn't shown in the Stepper] is:
; (apply f (list a b c ...))
; (f a b c ...)

; EXERCISE. Show the steps to produce the result of:

(apply beside (map rotate-cw (list . )))