(require picturing-programs)

; string -> image
; Dark blue image of the text in 'a-string' using a font of size 10.
(define (string->image a-string)
  (text a-string 10 "darkblue"))

; encircle : image -> image
; Version of 'an-image' with a dark orange ellipse around it.
(define (encircle an-image)
  (overlay an-image
    (ellipse
      (round (* 1.25 (image-width an-image)))
      (round (* 1.75 (image-height an-image)))
      "outline"
      "darkorange")))

; list-of-strings->image : list-of-strings -> image
(define (list-of-strings->image los)
  (encircle
   (apply beside
     (map encircle
      (map string->image
           los))))))

; list-of-lists-of-strings->image : list-of-lists-of-strings -> image
(define (list-of-lists-of-strings->image lolos)
  (encircle
   (apply beside
     (map list-of-strings->image
           lolos)))))

(check-expect
  (list-of-lists-of-strings->image (list (list "cat" "dog" "snail")
                                      (list "rhinoceros" "hippo")
                                      (list "panda" "monkey")
                                      (list "dolphin" "porpoise"))))

; Now let's make 'nested-strings->image' that can go arbitrarily deep into nested lists such as:

(define lll
  (list (list (list "cat" "dog" "snail")
              (list (list "rhinoceros" "hippo")
                    (list "panda" "monkey")
                    (list "dolphin" "porpoise")
                    (list "aardvark" "mongoose")
                    (list "ferret" "duck"))))
(check-expect (nested-strings->image "cat") (string->image "cat")))

(check-expect (nested-strings->image (list "cat" "dog" "snail"))
  (encircle
    (string->image "cat")))

(check-expect (nested-strings->image (list "cat" "dog" "snail"))
  (encircle
    (apply beside
      (map encircle (map string->image (list "cat" "dog" "snail"))))))

(check-expect (nested-strings->image (list "cat" "dog" "snail"))
  (encircle
    (apply beside
      (map nested-strings->image (list "cat" "dog" "snail"))))))

(check-expect (nested-strings->image (list "cat" "dog" "snail"))
  (encircle
    (apply beside
      (map nested-strings->image (list (list "rhinoceros" "hippo")
        (list "panda" "monkey")))))

(check-expect (nested-strings->image (list (list "rhinoceros" "hippo")
  (list "panda" "monkey")))
  (encircle
    (apply beside
      (map nested-strings->image (list (list "rhinoceros" "hippo")
        (list "panda" "monkey")))))))

; nested-strings->image : list-or-string -> image
(define (nested-strings->image list-or-string)
  (encircle
    (cond [(string? list-or-string) (string->image list-or-string)]
        [else (apply beside
               (map nested-strings->image list-or-string)))])))

(check-expect (nested-strings->image (first lll)))

(check-expect (scale 1/2 (nested-strings->image lll)))