Some DrRacket Tidbits

You can download DrRacket from:

https://www.racket-lang.org/, and click the **DOWNLOAD** button on the top right. (Either DrRacket 8.14 or DrRacket 8.13 should be fine for CS 111 for Fall 2024.)

How to set the Racket Language Level:

- go to the **Language** menu at the top of the window, and select the **Choose Language...** command;
- in the window that opens up, look for **Teaching Languages**, and look for **How to Design Programs** right under **Teaching Languages**
  - If necessary, click on the small blue triangle to the left of **How to Design Programs** to expand the list of languages in that category.
  - Select the **Beginning Student** language and click the **OK** button in the lower-right corner.
  - Back in DrRacket, click the **Run** button on the top-right corner of the window to finish setting the desired language level.
- You should now see something like the following at the top of the Interactions window:

```
Welcome to DrRacket, version 8.14 [cs].
Language: Beginning Student; memory limit: 128 MB.
```

Saving your Definitions window

Use the **File** menu's **Save Definitions** or **Save Definitions as...** commands, or the **Save** button.

TIP: if you are using `vlab.humboldt.edu`, save to your Google Drive!

To use the image and universe modules:

To be able to use functions and other items from the latest versions of the `image` and `universe` modules, put these two expressions at the beginning of your Definitions window:

```
(require 2htdp/image)
(require 2htdp/universe)
```

A selection of functions from the image module:

```
; signature: circle: number string string -> image
; purpose: expects a radius in pixels, either "solid" or
;    "outline", and a color written as a string, and
;    returns a circle image with that radius, style, and color

(circle 30 "solid" "red")

; signature: image-width: image -> number
; purpose: expects an image, and returns its width in pixels

(image-width (circle 30 "solid" "red"))   should return: 60
```

```
; signature: image-height: image -> number
; purpose: expects an image, and returns its height in pixels

(image-height (circle 30 "solid" "red")) should return: 60

; signature: rectangle: number number string string -> image
; purpose: expects a width and height in pixels, either "solid"
;    or "outline", and a color expressed as a string, and returns
;    a rectangular image of that width, height, style, and color

(rectangle 50 20 "outline" "blue")

; signature: regular-polygon: number number string string -> image
; purpose: expects the length of each side, the number of sides,
;    either "solid" or "outline", and a color expressed as a string,
;    and returns an image that is a regular polygon with that many
;    sides, each of that length, in that mode and color

(regular-polygon 15 6 "solid" "purple")

; signature: overlay: image image ... -> image
; purpose: expects any number of images, and returns a new image
;    that is the first image on top of the second image on top of the
;    third image and so on, all lined up by their pinholes(centers)

(overlay (circle 15 "solid" "red")
         (rectangle 40 60 "outline" "blue")
         (regular-polygon 70 5 "solid" "green"))

; signature: beside: image image ... -> image
; purpose: expects any number of images, and returns a new image
;    that's the 1st image to the left of the 2nd image to the left of
;    the 3rd image and so on, lined up by their pinholes (centers)

(beside (rectangle 10 60 "solid" "blue")
        (rectangle 10 40 "solid" "green")
        (rectangle 10 60 "solid" "blue")
        (rectangle 10 40 "solid" "green"))

; signature: above: image image ... -> image
; purpose: expects any number of images, and returns a new image
;    that is the first image above the second image above the third
;    image and so on, lined up by their pinholes (centers)

(above (rectangle 40 20 "solid" "blue")
       (rectangle 30 20 "solid" "green")
       (rectangle 40 20 "solid" "blue")
       (rectangle 30 20 "solid" "green"))

; signature: text: string number string -> image
; purpose: expects a string, a desired font-size, and a color
;    expressed as a string, and returns an image of text in that
;    font-size and color

(text "Hi!" 15 "black")
```

```
(overlay (text "Hi!" 40 "black")
         (rectangle 60 50 "solid" "yellow")
         (rectangle 80 55 "solid" "brown"))
```

To give a name a value

You can give a name to a value -- you can define an identifier and give it a value -- by using the `define` operation. The syntax is:

`(define` *name-you-choose* *expression*`)`

This expression doesn't have a value, but it does have an important side-effect -- after this, that name you have chosen can be used as a simple expression whose value is that expression's value.

For example, after the expression:

```
(define BLUE-DOT (circle 5 "solid" "blue"))
```

...you can now use `BLUE-DOT` anywhere an image expression can be used, and it will be an image of a little solid blue circle of radius 5 pixels.

To write a test using check-expect

The `check-expect` operation expects two expressions: the one you want to test, and one giving the expected value for that expression:

`(check-expect` *expression-to-test* *expected-value-expression*`)`

For example, to test if the expression `(number->string (+ 1 6))` is really `"7"`, you'd use:

```
(check-expect (number->string (+ 1 6)) "7")
```

Getting help in DrRacket

- You should be able to find DrRacket documentation on the web at: https://docs.racket-lang.org/
- DrRacket also has a help system (although it has more than just Beginning Student information in it...!)
- If you click on the **Help** menu, and select **Racket Documentation** or **Help Desk**, you can reach some useful Racket resources.
  - under the **Help** menu, if you select **Racket Documentation**,
  - type `2htdp/image` in the search textfield in the top left corner,
  - and click on the match "**2htdp/image module**",
  - the left-hand side of that section lists many possible image-related operations,
  - each a link to documentation for that operation.
- If you enter a function name in the **Help** menu and there are multiple versions of that function, remember that you are in the **HtDP2 - Beginning Student** level at this point. Click the option that includes `2htdp`.