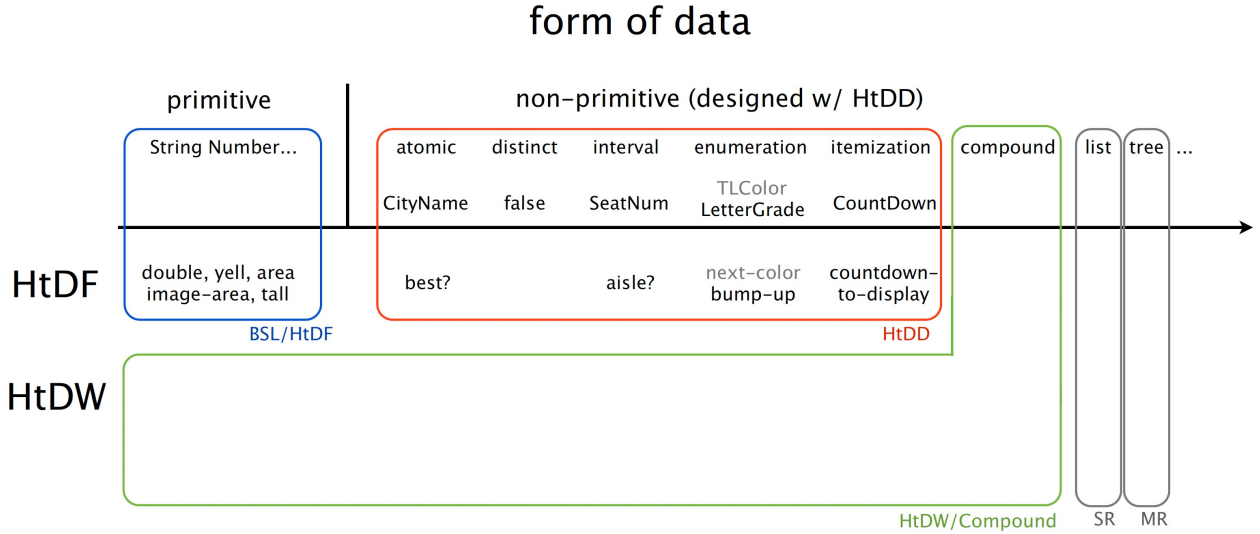

form of data
primitive
non-primitive (designed w/ HtDD)
String Number...
atomic
distinct
interval
enumeration
itemization
compound
list
tree
...
CityName
false
SeatNum
TLColor
LetterGrade
CountDown
HtDF
double, yell, area
image-area, tall
best?
aisle?
next-color
bump-up
countdown-
to-display
BSL/HtDF
HtDD
HtDW
HtDW/Compound
SR
MR

lecture 05 – 2 main topics

- revising an existing program
  - in this case a world program
  - you will see this in problem sets 4 and 5, can also show up in exam problems
  - and is nearly all of the work developers do
  - as always, we will work systematically, re-running the recipes as needed
- compound data
  - “2 or more items of information that naturally belong together”

- also, variations on enumeration templating
  - large enumerations
  - enumerations where an additional parameter is complex

SPD Checklists

See full recipe page for details

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

;; My world program  (make this more specific)
(@htdw WS)
;; =================
;; Constants:


;; =================
;; Data definitions:

(@htdd WS)
;; WS is ... (give WS a better name)



;; =================
;; Functions:

(@htdf main)
(@signature WS -> WS)
;; start the world with (main ...)
;;
(@template-origin htdw-main)
(define (main ws)
  (big-bang ws          ;WS
    (on-tick   tock)    ;WS -> WS
    (to-draw   render) ;WS -> Image
    (on-mouse  ...)     ;WS Integer Integer MouseEvent -> WS
    (on-key    ...)))  ;WS KeyEvent -> WS

(@htdf tock)
(@signature WS -> WS)
;; produce the next ...
;; !!!
(define (tock ws) ws)

(@htdf render)
(@signature WS -> Image)
;; render ...
;; !!!
(define (render ws) empty-image)
```

HtDW

1. Domain analysis (use a piece of paper!)
    1. Sketch program scenarios
    2. Identify constant information
    3. Identify changing information
    4. Identify big-bang options
2. Build the actual program
    1. Constants (based on 1.2 above)
    2. Data definitions (based on 1.3 above)
    3. Functions
        1. main first (based on 1.4 and 2.2 above)
        2. wish list entries for big-bang handlers
    4. Work through wish list until done

```
on-tick
to-draw
on-key
on-mouse
```

HtDD

First identify form of information, then write:

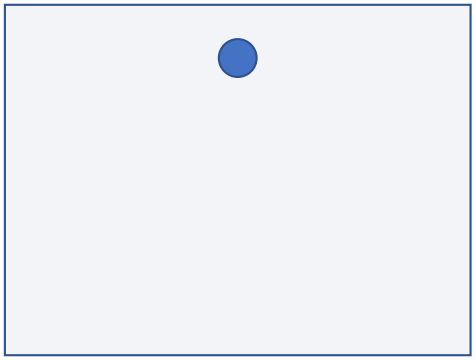
1. A possible structure definition (not until compound data)
2. A type comment that defines type name and describes how to form data
3. An interpretation to describe correspondence between information and data.
4. One or more examples of the data.
5. A template for a 1 argument function operating on data of this type.

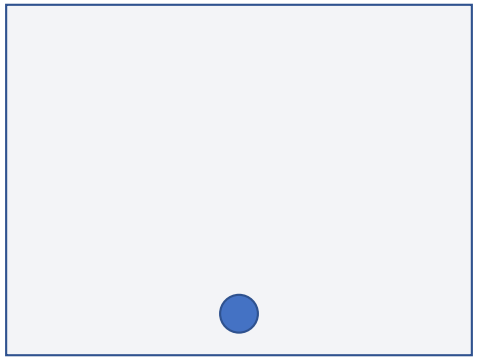
HtDF

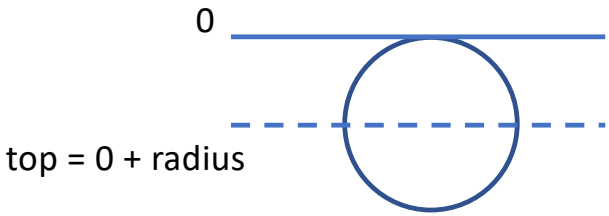
1. Signature, purpose and stub.
2. Define examples, wrap each in check-expect.
3. Template and inventory.
4. Code the function body.
5. Test and debug until correct

Test guidelines

1. at least 2
2. different argument/field values
3. code coverage
4. points of variation in behavior
5. 2 long / 2 deep







Constant

width
height
center x
speed
spider radius

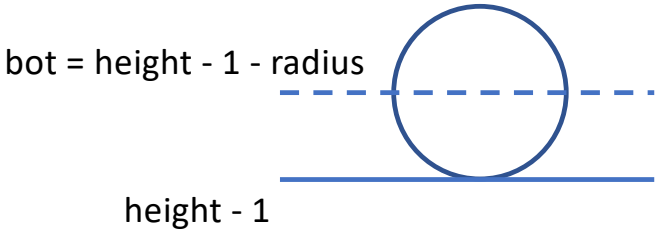
spider image
mts

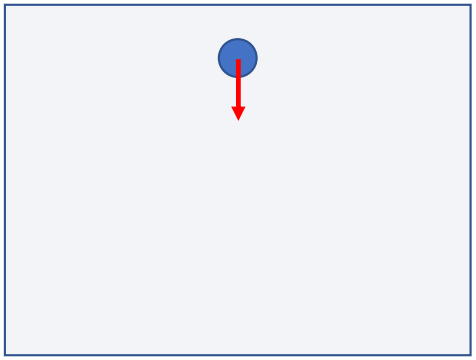
Changing

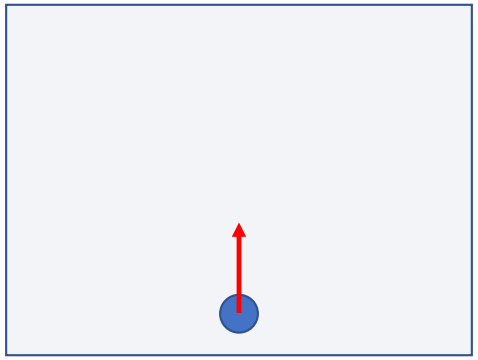
spider y

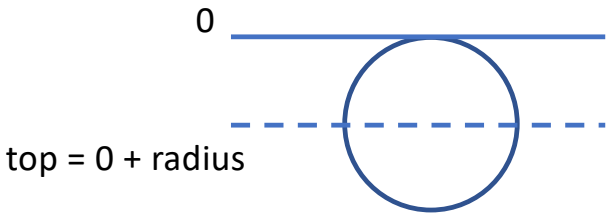
BB options

on-tick
to-draw
~~on-key~~
~~on-mouse~~









Constant

width
height
center x
~~speed~~
spider radius

spider image
mts

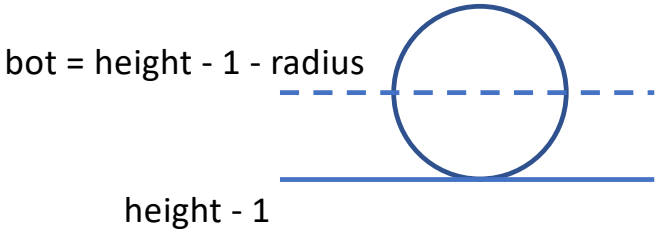
Changing

spider y
spider dy

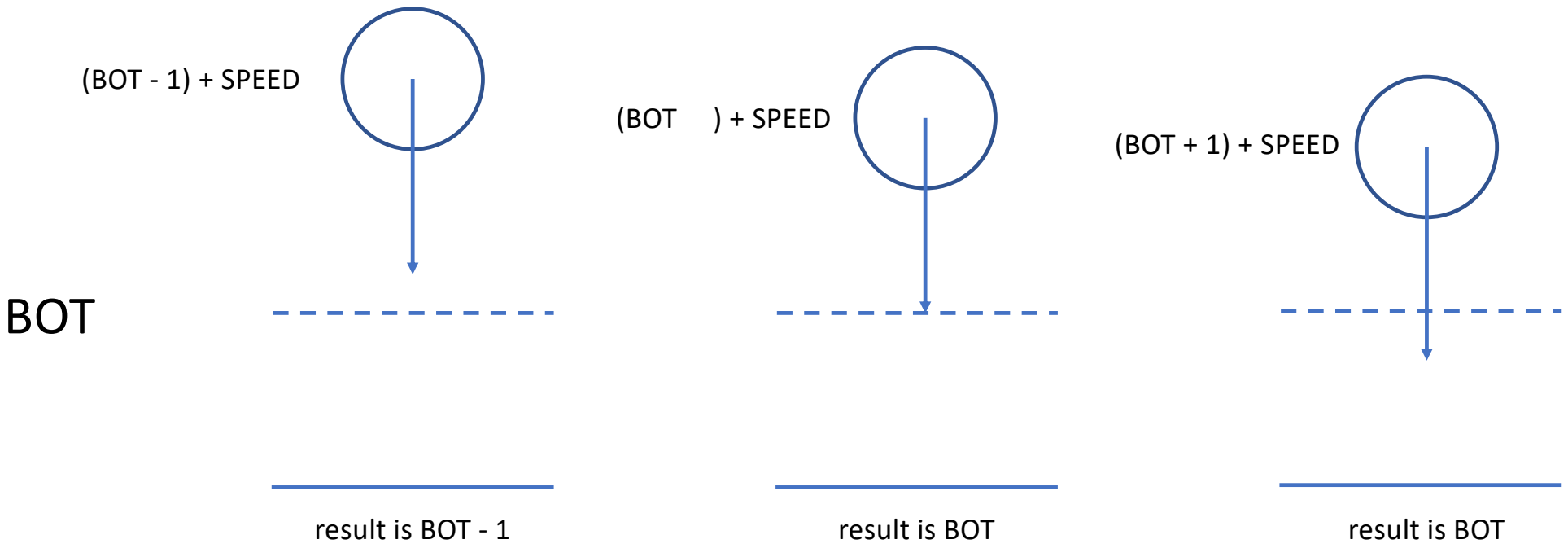
BB options

on-tick
to-draw
on-key
~~on-mouse~~

change direction at, and
don't go past top/bot



boundary case analysis for tock



boundary case analysis for tock

plus symmetric (mirror image) cases with TOP