Yellow highlighted portions are omitted from handout given to students and are filled in during lecture.

Expressions

Expression	Values	Operators
(+ 2 3 5)	2 3 5	+
(* 3.1 2.5)	3.1, 2.5	*
(+ (* 3 2.2) 7)	7, 6.6	*, +
(string-append “a” “b” “c”)	“a”, “b”, “c”	string-append
(circle 20 “solid” “red”)	20, “solid”, “red”	circle
(substring “abcd” 0 2)	“abcd”, 0, 2	substring
(beside )		beside

Yellow highlighted portions are omitted from handout given to students and are filled in during lecture.

Calling Versus Defining a function

CALLING a function	**DEFINING a function**

`(fn-name arg-1 arg-2)`	`(define (fn-name arg-name-1 arg-name-2)`
	`  (an expression using`
	`  arg-name-1 and arg-name-2))`

`(+ 3 3)`
 	
`(circle 20 “solid” “red”)`
	`(define (bulb c)`
	`  (circle 20 “solid” c))`

Yellow highlighted portions are omitted from handout given to students and are filled in during lecture.

Functions Versus Data,
Similar concept to define your own Data type

Functions		
Use them…	**Define them…**	
CALL **by name, passing the expected value(s)**	**Already defined for us** **BSL built-in/primitive functions/operators**	**Define our own** **using HtDF recipe**

`(+ 4 5)`
`(* (+ 4 3) 5)`
`(substring "Hi" 0 1)`
 	
`(bulb "red")`
 	
`(pos? -1)`
`(pos? (- 5 7))`
	+, * , /, -
	string-append
	substring
	positive?
	
	More in helpdesk…
		`(define (bulb c)`
		`(circle 20 "solid" c))`
		
		`(define (pos? n)`
		`(> n 0))`

Data Types		
Use them…	**Define them…**	
In the signature of our function	**BSL built-in/primitive types**	**Define our own** **using HtDF recipe**

`;; Natural -> Natural`
 	
`;; String Number -> String`
 	
`;; Number -> Boolean`
 	
`;; TeamName -> Boolean`
 	
`;; Age -> Boolean`
	Number
	Integer
	Natural
	
	String
	Boolean
	
	More in helpdesk…
		TeamName
		
		Age

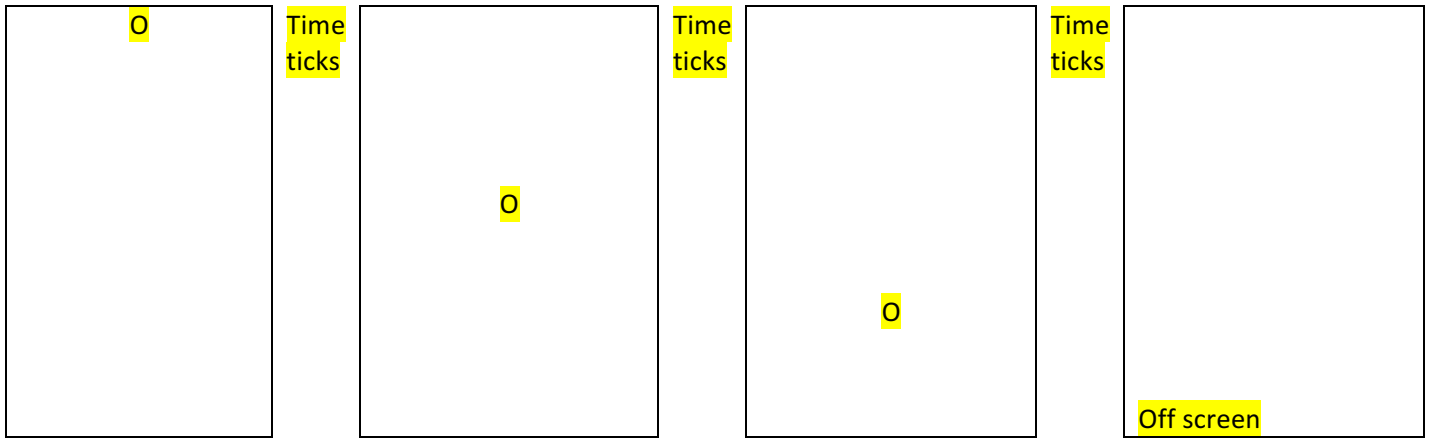
Yellow highlighted portions are omitted from handout given to students and are filled in during lecture.

Information Versus Data

Information	➔ represent <- interpret	Data
Problem Domain		**Program**
Street lights Red Yellow Green		0 1 2 Natural[0, 2]
Character health 1 life 2 lives 10 lives Limit? dead		 Natural Natural[1,10] - constrained false
Team name Canucks Raptors BlueJays		 String
Person age 0 20 29 105 200		 Natural[0,110]

Yellow highlighted portions are omitted from handout given to students and are filled in during lecture.

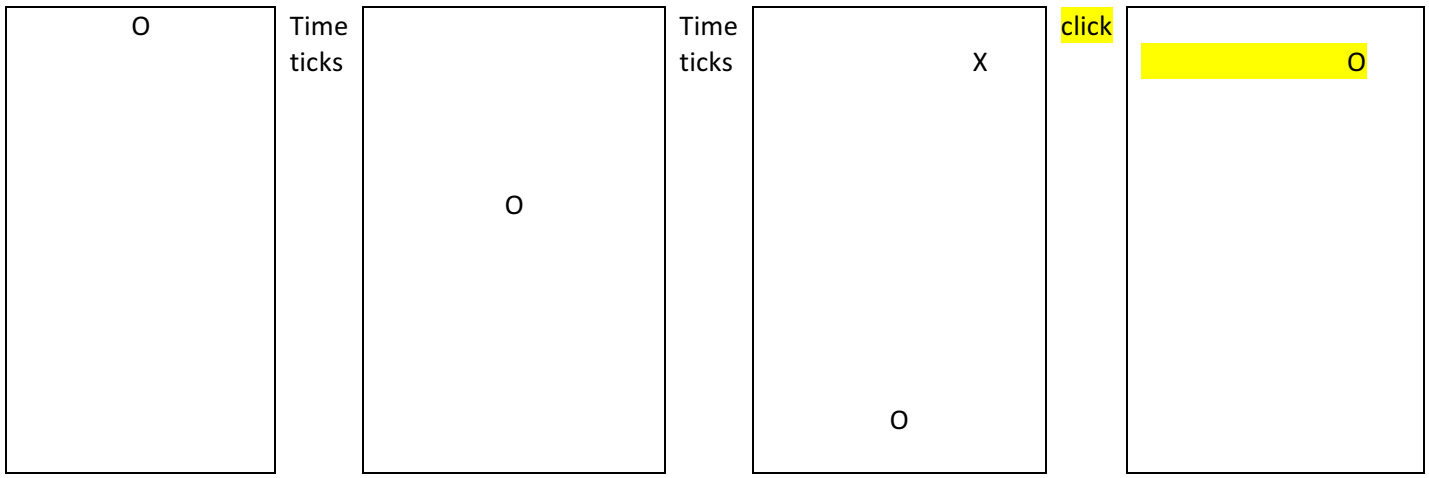
Domain Analysis – dropping spider



Constant Information	Changing Information	Big Bang options
MTS Spider image Width screen Height screen Ctr x Speed Top Bottom	Y coordinate of spider	On tick To draw

Yellow highlighted portions are omitted from handout given to students and are filled in during lecture.

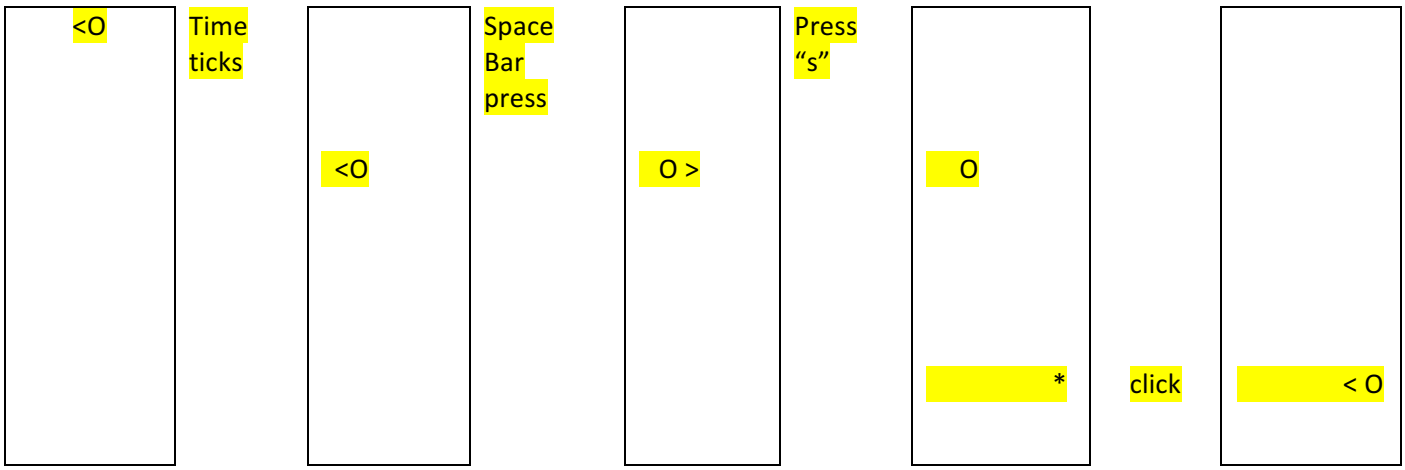
Domain Analysis – dropping spider continued…



Constant Information	Changing Information	Big Bang options
MTS Spider image Width screen Height screen Ctr x Speed Top Bottom	Y coordinate of spider X coordinate of spider	On tick To draw On mouse

Yellow highlighted portions are omitted from handout given to students and are filled in during lecture.

Domain Analysis – butterfly change direction…



Constant Information	Changing Information	Big Bang options
MTS Butterfly image Width screen Height screen Ctr x Ctr y Speed	Y coordinate of butterfly X coordinate of butterfly Direction of butterfly	On tick To draw On mouse On key

Yellow highlighted portions are omitted from handout given to students and are filled in during lecture.

ListOfNatural v Natural (atomic) v Natural (self-ref)

ListOfNatural Data Definition	Natural Data Definition
An arbitrary number of values	**A single value that specifies how many times to repeat something**

`;; ListOfNatural is one of:`
`;; - empty`
`;; - (cons Natural ListOfNatural)`
`;; interp. a list of natural numbers`
`(define LON-MT empty)`
`(define LON-1 (cons 3 empty))`
`(define LON-2 (cons 3 (cons 4 empty)))`
	`;; Natural is one of:`
	`;; - 0`
	`;; - (add1 Natural)`
	`;; interp. a natural number`
	`(define N0 0)  ;0`
	`(define N1 (add1 N0))  ;1`
	`(define N2 (add1 N1))  ;2`

`(define (fn-for-lon lon)`
`  (cond [(empty? lon) (...)]`
`  [else`
`  (... (first lon)`
`  (fn-for-lon (rest lon n)))]))`
	6
	`(define (fn-for-natural n)`
	`  (cond [(zero? n) (...)]`
	`  [else`
	`  (... n`
	`  (fn-for-natural (sub1 n)))]))`

Produce a list one element longer, use cons:
 	
`(cons 6 LON-2)`
 	
`(cons 6 (cons 3 (cons 4 empty)))`
	1
	Produce a natural one bigger, use add1:
	
	`(add1 N2) ==> 3`
	
	`(add1 2) ==> 3`

Produce a list one element shorter, use rest:
 	
`(rest LON-2)`
`  ==> (cons 4 empty)`
 	
`(rest (cons 3 (cons 4 empty)))`
`  ==> (cons 4 empty)`
	2
	Produce a natural one smaller, use sub1:
	
	`(sub1 N2) ==> 1`
	
	`(sub1 2) ==> 1`

Access the first element in the list, use first:
 	
`(first LON-2) ==> 3`
 	
`(first (cons 3 (cons 4 empty))) ==> 3`
	3
	What is the first natural encountered in the countdown?
	
	`n`

To operate on the remaining elements in the list, use NR: `(recursive-call (rest LON-2))`	**4** To operate on the remaining naturals, use NR: `(recursive-call (sub1 n))`
Base-case (when the recursion stops) is: when the list is empty… `(empty? lon)`	**5** Base-case (when the recursion stops) is: when the Natural is 0… `(zero? n)`

Yellow highlighted portions are omitted from handout given to students and are filled in during lecture.

Functions

Use them…	Define them…	
CALL by name, passing the expected value(s)	**Already defined for us BSL built-in/primitive functions/operators**	**Define our own using HtDF recipe**

`(+ 4 5)`
`(* (+ 4 3) 5)`
`(substring "Hi" 0 1)`
 	
`(bulb "red")`
 	
`(pos? -1)`
`(pos? (- 5 7))`
	+, * , /, -
	string-append
	substring
	positive?
	
	More in helpdesk…
		`(define (bulb c)`
		`(circle 20 "solid" c))`
		
		`(define (pos? n)`
		`(> n 0))`
		
		`(define (sum-to-n n)`
		`(cond [(zero? n) 0]`
		`[else`
		`(+ n`
		`(sum-to-n (sub1 n)))]))`

Data Types

Use them…	Define them…	
In the signature of our function	**BSL built-in/primitive types**	**Define our own using HtDD recipe**

`;; Natural -> Natural`
`;; produce 2 * n`
`(define (double n)`
`(* n 2))`
 	
`;; String Number -> String`
 	
`;; TeamName -> Boolean`
 	
`;; Age -> Boolean`
 	
`(define S1`
`(make-student "Jim"`
`3.5))`
 	
`(student-name S1)`
`;evaluates to "Jim"`
 	
`(student-id S1)`
`;evaluates to 3.5`
 	
`;; Student -> Number`
`;; produce gpa of s as %`
 	
`;; Natural -> Natural`
`;; produce sum o number`
`;; 0 to n inclusive`
	Number
	Integer
	Natural
	String
	Boolean
	More in helpdesk…
		`;; TeamName is String`
		
		`;; Age is Natural[0,110]`
		
		`(define-struct student (name gpa))`
		`;; Student is (make-student String Number)`
		
		`;; Natural is one of:`
		`;; - 0`
		`;; - (add1 Natural)`
		`;; interp. a natural number`

Yellow highlighted portions are omitted from handout given to students and are filled in during lecture.

Abstract Functions – add them to concept model

Functions		
Use them…	**Define them…**	
CALL **by name, passing the expected value(s)**	**Already defined for us** **BSL built-in/primitive functions/operators**	**Define our own** **using HtDF recipe**

`(+ 4 5)`
`(* (+ 4 3) 5)`
`(substring "Hi" 0 1)`
 	
`(bulb "red")`
 	
`(pos? -1)`
`(pos? (- 5 7))`
 	
`(map sqr (list 2 3 4))`
`⇨ (list 4 9 16)`
 	
`(foldr + 0`
`(list 2 3 4))`
`⇨ 9`
	+, *, /, -
	string-append
	substring
	positive?
	
	More in helpdesk…
	
	map
	filter
	foldr
	build-list
	andmap
	ormap
		`(define (bulb c)`
		`(circle 20 "solid" c))`
		
		`(define (pos? n)`
		`(> n 0))`
		
		`created from examples:`
		`map-2`
		`filter-2`
		
		created from template:
		fold
		fold-unit