




CS10
The Beauty and Joy of Computing

Lecture #17
Higher Order Functions

2014-11-03


SELF-DRIVING CARS



UC Berkeley EECS
Lecturer
Gerald Friedland

PRO

- Fewer accidents – 90% of accidents caused by human error
- Efficient travel since can create convoys
- Huge efficiency gains if you can work + drive



CON

- Who gets sued when there's an accident?
- Handing control back to driver takes ~5 sec
- Very expensive
- Could be dangerous if they can't handle case

www.technologyreview.com/featuredstory/520431/driverless-cars-are-further-away-than-you-think/

Why use functions? (review)

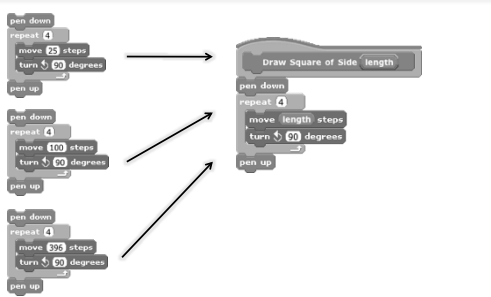
```

pen down
repeat 4
  move 25 steps
  turn 90 degrees
pen up

pen down
repeat 4
  move 100 steps
  turn 90 degrees
pen up

pen down
repeat 4
  move 375 steps
  turn 90 degrees
pen up

```



The power of Abstraction!

Midland
UC Berkeley CS10 "The Beauty and Joy of Computing" : HOF (2)

Peer Instruction

I understand functions.

- Strongly disagree
- Disagree
- Neutral
- Agree
- Strongly agree

Midland
UC Berkeley CS10 "The Beauty and Joy of Computing" : HOF (3)

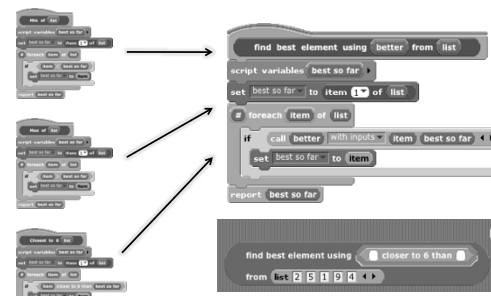
But how general can we be?

```

find best element using 'better' from list
script variables: best so far
set best so far to item 1 of list
foreach item of list
  if call better with inputs item best so far
    set best so far to item
report best so far

find best element using 'closer to 6 than'
from list 2 5 1 0 4

```

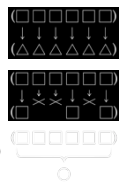


The power of even more Abstraction!

Midland
UC Berkeley CS10 "The Beauty and Joy of Computing" : HOF (4)

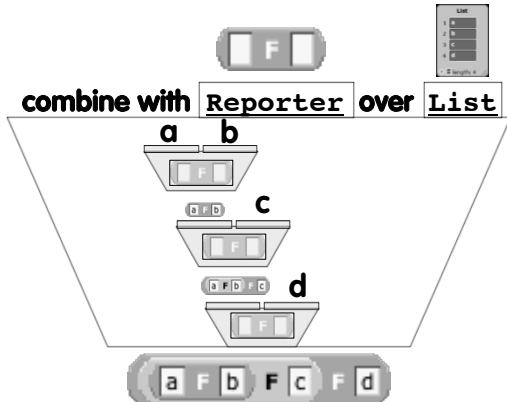
Today

- Functions as Data
- Higher-Order Functions
- Useful HOFs (you can build your own!)
 - map Reporter over List
 - Report a new list, every element E of List becoming Reporter(E)
 - keep items such that Predicate from List
 - Report a new list, keeping only elements E of List if Predicate(E)
 - combine with Reporter over List
 - Combine all the elements of List with Reporter(E)
 - This is also known as "reduce"
- Acronym example
 - keep → map → combine



Midland
UC Berkeley CS10 "The Beauty and Joy of Computing" : HOF (5)

combine with Reporter over List



Peer Instruction


I understand higher-order functions.

- Strongly disagree
- Disagree
- Neutral
- Agree
- Strongly agree

UC Berkeley CS10 "The Beauty and Joy of Computing" : HOF (7)

Let's Play Board Games...

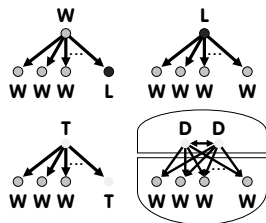
- No chance, such as dice or shuffled cards
- Both players have complete information
 - No hidden information, as in Stratego & Magic
- Two players (Left & Right) usually alternate moves
 - Repeat & skip moves ok
 - Simultaneous moves not ok
- The game can end in a pattern, capture, by the absence of moves, or ...



UC Berkeley CS10 "The Beauty and Joy of Computing" : HOF (8)

A Strong Solution visits every position


- For every position
 - Assuming alternating play
 - Value ... (for player whose turn it is)
 - Winning (☑) losing child
 - Losing (All children winning)
 - Tieing (☑) losing child, but ☑ tieing child
 - Drawing (can't force a win or be forced to lose)
 - Remoteness
 - How long before game ends?



UC Berkeley CS10 "The Beauty and Joy of Computing" : HOF (9)

Strong Solving Example: 1,2,...,10

- Rules (on your turn):
 - Running total = 0
 - Add 1 or 2 to running total
- Goal: Be the FIRST to get to 10
- Example:
 - Ana: "2 to make it 2"
 - Bob: "1 to make it 3"
 - Ana: "2 to make it 5"
 - Bob: "2 to make it 7" → photo
 - Ana: "1 to make it 8"
 - Bob: "2 to make it 10" I WIN!



7 ducks (out of 10)

UC Berkeley CS10 "The Beauty and Joy of Computing" : HOF (10)

Let's write code to determine value!

- 0 = Win
- 1 = Lose
- 2 = Win
- 3 = Win
- 4 = Lose
- 5 = Win
- 6 = Win
- 7 = Lose
- 8 = Win
- 9 = Win
- 10 = Lose

- P = Position
- M = Move
- We only need 3 blocks to define a game
 - Do Move M on Position P
 - a new Position
 - Generate Moves from Position P
 - list of Moves
 - Primitive Value of Position P
 - {win, lose, tie, undecided}
- Let's write Value of Position P

UC Berkeley CS10 "The Beauty and Joy of Computing" : HOF (11)

Answer

```

Value of Position: position
if not Primitive Value: position = CONSTANT Undecided
report Primitive Value: position
else
  script variables: children, child values
  set children to map Do Move M on Position: position over Generate Moves from Position: position
  set child values to map Value of Position over children
  if child values contains CONSTANT Lose
    report CONSTANT Win
  else
    if child values contains CONSTANT Tie
      report CONSTANT Tie
    else
      report CONSTANT Lose
  
```

UC Berkeley CS10 "The Beauty and Joy of Computing" : HOF (12)

bjc



bjc

Typical function machines use levers

- 

