Q&A Session for Programming Languages Lecture 12

Session Number: 1203115422
Date: 2020-10-16
Starting time: 14:27

ANON - 14:27
Q: When studying for the last exam, I noticed that a few powerpoints that were uploaded included annotations for in-lecture problems or exercises that we went over in class but most did not. Is it possible that we can continue to include these annotations?
Priority: N/A
	Ana L. Milanova - 14:37
	A: Missing answers in notes is done on purpose, to encourage the class to attend lecture :).

ANON - 14:37
Q: what does R5RS stand for?
Priority: N/A
	Ana L. Milanova - 14:38
	A: Revised Report 5 of the Algorithmic Language Scheme.

ANON - 14:40
Q: At RPI so far, we've learned imperative paradigm, but in this course we've used logical paradigm and now functional paradigm?
Priority: N/A
	Ana L. Milanova - 14:41
	A: Yes, in earlier courses we have used languages that are classified as "imperative". In this class we learn logic programming and functional programming.
	Ana L. Milanova - 14:43
	A: Languages nowadays are increasingly "multiparadigm", so traditionally "imperative" languages more and more include features that traditionally have been associated with the functional programming paradigm.

ANON - 14:40
Q: Is this like Assembly OpCodes?
Priority: N/A
	Ana L. Milanova - 14:43
	A: I missed the slide that you are referring to, as I was typing an answer to another question. What slide was that on?

ANON - 14:44

Q: Re: Assembly Question
Priority: N/A

ANON - 14:44
Q: the slide about the syntax with assigning value
Priority: N/A
Ana L. Milanova - 14:46
A: If I understand the question right, I might be misinterpreting. Yes, these assignments get naturally translated into assembly, using loads and stores and arithmetic opcodes.

ANON - 14:45
Q: does visible state refer to how the values of global variables can change based on when/where a function is called? aka f(5) now could return a different result than f(5) later. is there anything more than that for this concept?
Priority: N/A
Ana L. Milanova - 14:49
A: visible state can be more general than just global variables. But global variables is the most natural example when explaining side effects.

ANON - 14:48
Q: would an int be considered a first-class value since you can pass in an int and return an int?
Priority: N/A
Ana L. Milanova - 14:49
A: Yes, an int variable, or an int constant is a first-class value.

ANON - 14:48
Q: So if i understood what was explained, since we called f(5) the first time, the program will save the last result and when f(5) is called again, its takes that last result and computes?
Priority: N/A
Ana L. Milanova - 14:52
A: Yes, the call to f(5) depends on visible memory state. If memory state changed between two invocations of f(5), then the result of f(5) may change in the second invocation.

ANON - 14:50
Q: when a functional programming language uses reduction semantics, does each reduction/execution step count as a "state" similar to state-transitional langauges?
Priority: N/A

Ana L. Milanova - 16:37
A: In the reduction semantics each execution step reduces the program from one term to another. You can think of each term as being the current "state" of execution. But we have no notion of "memory state" which is a mapping of variables to values.
Ana L. Milanova - 16:37
A: as we have in imperative programming, where memory state changes after a step of execution.

ANON - 14:51
Q: Wait, why aren't functions in C first class values? Can't we do something like:
Priority: N/A
Steven Haussmann - 14:52
A: For context: followup question showed a function's result being passed to another function. This isn't indicative of functions being first-class; it just means that you can use the return value of one function when calling another.
Steven Haussmann - 14:53
A: For functions to be "first-class", you would need to be able to pass and return them (which you nominally can, via function pointers). You also need to be able to create them. That's where C's functions fail to meet the mark.

ANON - 14:51
Q: as a followup, would a dynamically allocated vector on the heap be considered a first-class value? im not sure if the vector itself is passed in or the memory pointers are passed in
Priority: N/A

ANON - 14:51
Q: int foo() { ... return 1; }
Priority: N/A

ANON - 14:51
Q: bar(foo(), ...);
Priority: N/A

ANON - 14:55
Q: is there such thing as a pure functional langauge in application or is it more of a theoretical concept?
Priority: N/A
Steven Haussmann - 14:57
A: Yes, there are a number of purely functional languages. Haskell is good example.

ANON - 14:55
Q: When you say we can't create functions in C, does that mean like we can't create smth like an anonymous function on the fly?
Priority: N/A
Steven Haussmann - 14:55
A: Yes. You can't create a new function wherever you want at runtime.

ANON - 14:58
Q: I cannot hear Professor Milanova
Priority: N/A
Ana L. Milanova - 14:59
A: Yes, I'm sorry. My internet connection has been very flaky for the past two days. I got dropped just when I started talking. I'm back now but don't know for how long.

ANON - 14:59
Q: Which variant of racket should we download--Regular or CS?
Priority: N/A
Ana L. Milanova - 15:01
A: I was assuming the standard Racket distribution. Should be Regular?

ANON - 14:59
Q: Why is Lisp considered a high-level programming language?
Priority: N/A
Steven Haussmann - 15:00
A: Almost anything that's compiled down to some machine-executable form is considered "high-level"

ANON - 15:00
Q: What is prefix form exactly?
Priority: N/A
Steven Haussmann - 15:00
A: All expressions are written strictly in prefix form. The function comes first, followed by its arguments.

ANON - 15:00
Q: Could you clarify what is meant by visible and non-visible states?
Priority: N/A
Steven Haussmann - 15:01
A: Visible state is something that can affect computation. Non-visible state cannot -- even "pure" languages will have non-

visible state

ANON - 15:00
Q: Do functional languages evaluate from inner-most to outer-most OR from outer-most to inner-most? since we've seen examples of both in the slides
Priority: N/A
        Steven Haussmann - 15:02
        A: It depends entirely on the language. Nominally, for programs that don't get into infinite loops, it shouldn't really matter (but many programs can get into those...)

ANON - 15:02
Q: Aaah, so this is why HW3 had the Boolean xpressions as "and and a b c" = a and b and c
Priority: N/A
        Ana L. Milanova - 15:02
        A: And the test had S-expressions too :).

ANON - 15:04
Q: what does ({S-expr}) mean?
Priority: N/A
        Ana L. Milanova - 15:05
        A: 0 or more S-epxr in the parentheses.

ANON - 15:07
Q: So I could not find some of the lectures on mediasite, is that just me, or is there something I missed?
Priority: N/A
        Ana L. Milanova - 15:09
        A: It could be the UI. Try looking at the first page, or changing to Most Recent. As far as I know all lectures are there and they should be visible, but if you can't find them, just let us know on the proglang list.
        Steven Haussmann - 15:10
        A: The sorting is a bit off, because it's doing lexicographic sorting: 11 comes before 2

ANON - 15:08
Q: Why the curly braces?
Priority: N/A

ANON - 15:08
Q: Why the curly braces in ({S-expr})?

Priority: N/A
	Ana L. Milanova - 15:10
	A: This is EBNF (Extended Backus Naur Form) notation. It means 0 or more occurrences of the term enclosed in curly braces. In this case this is 0-or more S-expressions.

ANON - 15:09
Q: if we do cons of a and b, does the list a still exist on its own?
Priority: N/A
	Steven Haussmann - 15:10
	A: There are no side effects, so yes, the arguments will not be affected in any way.
	Ana L. Milanova - 15:11
	A: Yes, we are creating a list here. But it is an improper list.

ANON - 15:11
Q: So if we did cons of c and (a b) then its (c a b)? where car is c and cdr is (a b)?
Priority: N/A
	Steven Haussmann - 15:12
	A: Correct. Remember that the list syntax is just syntactic sugar for a bunch of cons operations.

ANON - 15:14
Q: Could we do more practice examples of car, cdr, and cons during the Q/A break?
Priority: N/A
	Ana L. Milanova - 15:15
	A: We will have some more coming in in lecture, but we can do even more if there is demand.

ANON - 15:15
Q: why do we need the quotes before '(a b c), if we did not have the ', what would it evaluate to?
Priority: N/A
	Steven Haussmann - 15:16
	A: It would try to evaluate (a b c) as a function named a being invoked with arguments b and c

---

ANON - 15:16
Q: So what would (car (a b c)) yield?
Priority: N/A
	Ana L. Milanova - 15:17
	A: (car (a b c)) will throw an error because it will try to

evaluate (a b c) and will find that "a" is undefined.
        Ana L. Milanova - 15:18
        A: But if you meant (car '(a b c)) that will yield "a".

ANON - 15:16
Q: If ' prevents an expression from being evaluated, why do we need it when evaluating list functions?
Priority: N/A
        Steven Haussmann - 15:17
        A: It prevents the S-expression that follows it from being interpreted as a function application
---

ANON - 15:16
Q: isn't (a b c) just a list and not a function? so how could it evaluate anything at all?
Priority: N/A
        Steven Haussmann - 15:17
        A: Remember that everything in Scheme is just an S-expression. (a b c) is an invocation of function "a" with arguments "b" and "c"

ANON - 15:17
Q: whats the difference between car '(abc) and car(abc)?
Priority: N/A
        Steven Haussmann - 15:17
        A: (car `(a b c)) will produce the head of the list of a, b, and c (which is a)
        Steven Haussmann - 15:18
        A: (car (a b c)) will produce the head of the result of evaluating (a b c). If there is no function named a, then this will be an error.
        Ana L. Milanova - 16:45
        A: To add, remember prefix syntax. We write (car '(a b c)), "car" is the function we are applying, '(a b c) is the argument. We CAN'T WRITE car('(a b c)) the way we do in other languages.

ANON - 15:18
Q: can you review what caddr means?
Priority: N/A
        Steven Haussmann - 15:18
        A: It's a sequence of car and cdr operations, read from right to left. So, it will get the tail of the list, then the tail of that, and then the head of that.
        Steven Haussmann - 15:19
        A: (caddr `(1 2 3 4 5)) is 3

ANON - 15:19
Q: so cons is an anti-quote?
Priority: N/A
	Steven Haussmann - 15:19
	A: No, cons just produces a cons cell with the arguments it's given.
	Steven Haussmann - 15:19
	A: I believe Milanova is just excluding the quote marks from the results.

ANON - 15:19
Q: if we did cdr of a list with a single element, will it return an empty list?
Priority: N/A
	Ana L. Milanova - 15:19
	A: Correct.

ANON - 15:21
Q: So the ' basically acts like a string in imperative languages? The interpreter takes everying following the ' as one argument.
Priority: N/A
	Ana L. Milanova - 15:22
	A: You can think of it this way I guess. The quote prevents evaluation, tells the interpreter this is a list of data.
	Steven Haussmann - 15:23
	A: It just means that the s-expression that follows should not be evaluated. The datum that's quoted can be anything -- a number, a string, a list, etc.

ANON - 15:21
Q: since the lanauge is fully-parenthesized, what would happen if we forgot to put parenthesis?
Priority: N/A
	Steven Haussmann - 15:22
	A: If you forget the outer-most parens, you'll just get each argument listed out by the interpreter. Otherwise, you'll probably get an error, because a function will be given too many/too few arguments.

ANON - 15:21
Q: What would happen if it was (zero? '(- 3 3)) ?
Priority: N/A
	Steven Haussmann - 15:24
	A: Error. It wants a number, not a list.

ANON - 15:22

Q: I am still confused why the result came to be (c d) in slide 17, i guess i am missing where the extra paranthesis come from
Priority: N/A
	Ana L. Milanova - 15:31
	A: (caddr '((a) b (c d))) is the same as (car (cdr (cdr '((a) b (c d))))), start evaluating from the inner cdr towards the outer car.

ANON - 15:23
Q: Does the cdr of () throw an error?
Priority: N/A
	Steven Haussmann - 15:23
	A: Yes. car and cdr both expect a list; `() is not a proper list.
	Ana L. Milanova - 15:31
	A: Yes, what Steven says. This will be an error because the empty list is an "atom", something we cannot take the car or the cdr of.

ANON - 15:23
Q: how is scheme dynamic? I don't remember there being any types declared at any point
Priority: N/A
	Steven Haussmann - 15:26
	A: Variables do not have types associated with them. Type information is contained in the objects they store.

ANON - 15:23
Q: I meant the cdr of '() ^^
Priority: N/A
	Ana L. Milanova - 15:32
	A: Will be the same result. As it is an "atom", we cannot take the car or the cdr of it.

ANON - 15:23
Q: for dynamic typing, what does typing mean?
Priority: N/A
	Steven Haussmann - 15:30
	A: "typing" refers to how type information is stored. A statically typed language requires types to be given for its variables.

ANON - 15:23
Q: So what is the difference between list? and pair?
Priority: N/A

Steven Haussmann - 15:24
A: list? tests if something is a proper list. pair? just checks if it has a head and tail

---

ANON - 15:24
Q: Why does (cons `(a b) `(c d)) yield ((a b) c d) and not ((a b) (c d))?
Priority: N/A
Steven Haussmann - 15:24
A: Because it produces a list whose head is the first argument and whose tail is the second argument.
Steven Haussmann - 15:25
A: The head is (a b); the tail is (c d) -- thus, you get a list of three things: (a b), c, d

---

ANON - 15:25
Q: so is (abc) in slide 15 different than the (abc) in slide 17?
Priority: N/A
Steven Haussmann - 15:26
A: I don't see (a b c) in slide 15 -- are you looking at (a (b c))? That's distinct, yes, since it's a list of a and (b c)

ANON - 15:26
Q: what's the difference between (cons 'a '(b)) and (cons 'a 'b)? I ran the latter and got (a . b). I read in the book that's an improper list, but I noticed you didn't mention improper lists in the lecture.
Priority: N/A
Steven Haussmann - 15:27
A: It was mentioned at one point -- but yes, the former is proper and the latter is improper.
Steven Haussmann - 15:27
A: The rule remains the same: a proper list's tail is either a proper list or the null list. (cons `a `b) produces a cons cell whose tail is not a proper list; thus, it is not a proper list.

ANON - 15:27
Q: Why does cdr('()) throw an error? Isn't '() a Proper List, namely the NULL List?
Priority: N/A
Steven Haussmann - 15:28
A: That's the one exception, yes -- whilst `() is a proper list, it also has no tail, as it's empty.

ANON - 15:27
Q: So is pair? () true?

Priority: N/A
	Steven Haussmann - 15:29
	A: (pair? `()) is false. You can't get the head or tail of a null list.

ANON - 15:28
Q: what is the result of (cons `() `()) ?
Priority: N/A
	Steven Haussmann - 15:29
	A: You will get a list containing a single element; that single element is the null list.
	Steven Haussmann - 15:29
	A: The head of the result is `() (so it contains a null list), and the tail of the result is also `() (so it is a proper list)
	Ana L. Milanova - 15:33
	A: (()). Try those in the interpreter, it's a lot of fun!

ANON - 15:30
Q: can you overrload / override functions in scheme?
Priority: N/A
	Steven Haussmann - 15:31
	A: It wouldn't really make sense, since types are dynamic. You can have multiple functions with the same name but differently-structured arguments, though

---

ANON - 15:31
Q: is true represented as #t and false represented as #f or is that like a comment?
Priority: N/A
	Steven Haussmann - 15:32
	A: Those are the exact representations, yes. The interpreter will balk at you if you try things other than #t or #f

---

ANON - 15:33
Q: why does the s-expr have to evaluate to a boolean value? can't the s-expr be a number or string, how do those evaluate here?
Priority: N/A
	Steven Haussmann - 15:34
	A: We're writing a function that decides if something is zero. It wouldn't really make sense to return other values. We could if we wanted to, of course.
	Steven Haussmann - 15:34
	A: Oh, sorry!
	Steven Haussmann - 15:35
	A: I misread your question. You're asking about if. Yes, the condition must evaluate to a boolean.

ANON - 15:34
Q: The function definitions with ? after their name evaluate to a boolean value, while those without the ? evaluate other values?
Priority: N/A
Ana L. Milanova - 15:35
A: Yes, ? means that the function is a predicate. It evaluates to true or false. Like Prolog predicates.
Steven Haussmann - 15:36
A: Note that this isn't enforced by the language. It's simply a convention.

ANON - 15:35
Q: is this evaluation similar to imperative langauges where in the case one of the conditions evaluates to false, there is no need to evaluate the other conditions?
Priority: N/A
Steven Haussmann - 15:36
A: I believe so, yes. This is only meaningful if there are side-effects, of course. You can test this by putting print statements in both branches and seeing how only one thing is printed.

ANON - 15:37
Q: i'm having a difficult time conceptually understanding how if s-expr have a simple definition, how they can be used everywhere in scheme?
Priority: N/A
Ana L. Milanova - 15:38
A: The S-expr is just the syntax. You can see that everything that we've written so far has this form, i.e., is an S-expr. But the S-epxressions have rich semantics, we can write and evaluate complex programs using this syntax.

ANON - 15:38
Q: In the Video, Professor Milanova said pair means you can take both the car and cdr? Does cdr imply you have a car?
Priority: N/A
Steven Haussmann - 15:38
A: Yes, I can't think of any situation where you'd have only a head or only a tail.
Ana L. Milanova - 15:41
A: Yes, I believe this is true. A pair is a "cons cell", which is made of a car-pointer and a cdr-pointer. A cons cell", by construction has both a car and a cdr.

ANON - 15:41
Q: should the trace just be one line evaluating if a is pair? why is everything else necessary?
Priority: N/A
Steven Haussmann - 15:42
A: The interpreter has to interpret each part of the line, looking up names and evaluating parts as needed. Several functions are being called, too.
Ana L. Milanova - 15:44
A: This nicely follows up on your previous question! The one line is the syntactic form, "(not (pair? object))". The interpreter "interprets" this S-expression (assigns meaning to it), which entails evaluating all those nested functions.

ANON - 15:43
Q: so does lambda (n) (* n n) just mean n = n * n?
Priority: N/A
Steven Haussmann - 15:43
A: It creates a function that takes an argument (named n) and produces the result of multiplying n with n
Steven Haussmann - 15:43
A: For example, ((lambda (n) (* n n)) 3) produces 9

ANON - 15:44
Q: what would happen if we do (pari? (a)) instead of (pair? `(a))? which means not passing (a) as an argument
Priority: N/A
Ana L. Milanova - 15:45
A: Then the interpreter tries to evaluate (a), and will complain that "a" is undefined. (Assuming that you don't have a definition of an "a" function in scope.)

ANON - 15:45
Q: are lambda functions considered anonymous?
Priority: N/A
Steven Haussmann - 15:46
A: "anonymous function" is a reasonable term, yes.

ANON - 15:45
Q: can you show this after this video? I can see absolutely none of that
Priority: N/A
Ana L. Milanova - 15:46
A: Sure, we can do that.
Ana L. Milanova - 15:47
A: It might still turn blurry but I'll try!

ANON - 15:46
Q: We can't really see what is in the console/window...
Priority: N/A
        Ana L. Milanova - 15:49
        A: Yes, will do.

ANON - 15:48
Q: after this video ends, could you draw out what's happening on slide 29? it would be helpful to visualize it
Priority: N/A
        Ana L. Milanova - 15:49
        A: Yes, will do.

ANON - 15:53
Q: how come there is no #t for the zchk if number? evalues to true?
Priority: N/A
        Steven Haussmann - 15:54
        A: It just returns (zero? x), which will be #t if it's zero and #f if not

---

ANON - 15:56
Q: What's the difference between cond and if in practice? I guess I'm asking which is more useful / common... seems like cond
Priority: N/A
        Steven Haussmann - 15:56
        A: cond lets you write many cases; if is just an if/else

---

ANON - 15:57
Q: For syntax, instead of inputting '(1 2) for len(), we put (1 2), Scheme will try to evaluate 1 as a function?
Priority: N/A
        Steven Haussmann - 15:58
        A: Yes, it will attempt to evaluate (1 2); however, since 1 is not a procedure, it will certainly give an error.

ANON - 15:58
Q: for slide 30, could you draw out what the recursive stack would look like?
Priority: N/A
        Ana L. Milanova - 15:59
        A: Ok, we'll do.

ANON - 16:03
Q: I'm curious if there's a general sense of when test grades and hw grades will be releasing?
Priority: N/A
Ana L. Milanova - 16:05
A: We are working hard on tests right now and aiming to have grades by Tuesday. HW3 will likely be 10 days from now as all graders are busy grading Exam.

ANON - 16:07
Q: does shallow recursion mean we're basically O(n) but deep recursion means we're O(n^something)? probably not exactly but the general idea
Priority: N/A
Steven Haussmann - 16:08
A: It means that we're just recursing over each element in the list, rather than diving into the elements themselves
Steven Haussmann - 16:09
A: But yes, a shallow-recursive function will probably run in linear time w.r.t. the length of the list, whilst a deep-recursive one will depend on what's in the list

---

ANON - 16:10
Q: why is atomcount deep recursive? is it because the input could potentially contain lists of lists?
Priority: N/A
Steven Haussmann - 16:10
A: Yes. It needs to sift through every list contained in the list.

ANON - 16:11
Q: if we know the input contains no nested lists, would it be guaranteed to be shallow recursion?
Priority: N/A
Steven Haussmann - 16:11
A: You would effectively have shallow recursion, yes, but it'd just be a coincidence.

ANON - 16:12
Q: For deep recursive runtime, I think it would be something like O(n), where n is the number of cons calls needed to create the structure provided; is this thinking correct?
Priority: N/A
Steven Haussmann - 16:13
A: I believe that's about right, yes. You'll need (n-1) cons calls to connect n things together.
Steven Haussmann - 16:13

A: sounds like something from Graph Theory :)

ANON - 16:12
Q: If you swapped the order of the Bases Cases in atomcount/fun, would this not just result in a Potential Exception if the Input is NULL?
Priority: N/A

Steven Haussmann - 16:16

A: Our atom? predicate just uses pair?, and pair? correctly handles the null list, so it should be fine.

ANON - 16:12
Q: i.e. the number of branchings
Priority: N/A

ANON - 16:15
Q: does that say coud?
Priority: N/A

ANON - 16:15
Q: If we append to a list in the Recursive Case, why does atom need to make it a list if the Input is a List?
Priority: N/A

ANON - 16:15
Q: is what's in blue a comment?
Priority: N/A

ANON - 16:16
Q: what is the differenct befor cons and append?
Priority: N/A

Steven Haussmann - 16:17

A: append will concatenate two lists. (append `(1 2) `(3 4)) is `(1 2 3 4)

Steven Haussmann - 16:17

A: cons will produce a list whose head is the first argument and whose tail is the second argument. (cons `(1 2) `(3 4)) is `((1 2) 3 4)

ANON - 16:16
Q: between∗
Priority: N/A

ANON - 16:17
Q: Does append exist in Scheme as well? I thought we only had cons?
Priority: N/A
	Steven Haussmann - 16:18
	A: there is an append function, yes

ANON - 16:17
Q: doesn't cons also concatenate two lists? what's the difference between what appends does and what cons does?
Priority: N/A
	Steven Haussmann - 16:19
	A: to be specific, cons just makes a pair. It constructs a cons cell pointing to its two arguments
	Steven Haussmann - 16:20
	A: append will produce a list containing all of the elements of its first argument, followed by all of the elements of its second argument
	Steven Haussmann - 16:20
	A: cons will produce a pair. The first thing in the pair (the head) is the first argument; the second thing in the pair (the tail) is the second argument.

ANON - 16:19
Q: When I opened DrRacket on my windows pc, it was running DrRacket in a terminal. Is this expected, or should I try to redownload ?
Priority: N/A
	Steven Haussmann - 16:19
	A: I get a GUI if I run DrRacket. If I just type "Racket" into the start menu, I get a terminal. Check that you did the former.

ANON - 16:19
Q: whats the difference between (define (square n) (*n n)) and (define square (lambda (n) (* n n)))?
Priority: N/A
	Steven Haussmann - 16:22
	A: functionally, there isn't a difference. The latter is just creating a lambda function and then immediately assigning it to that variable, rather than directly declaring a function wtih that name. There may be some finer distinctions under the hood.

ANON - 16:19
Q: and what is the difference between a pair and a list? is a list just a type of pair?
Priority: N/A
	Steven Haussmann - 16:21
	A: A list is either the null list or a pair whose second

element is also a list.
        Steven Haussmann - 16:24
        A: It's identical to how we defined improper and proper lists in Prolog.
        Ana L. Milanova - 16:31
        A: Yes, what Steven says. One key difference is that '() is not a pair but it is a list. Another difference is that an improper list is not a list, i.e., (list? (cons 'a 'b)) is #f, but an improper list is a pair.

ANON - 16:20
Q: You can also write the calls after the function.
Priority: N/A

ANON - 16:24
Q: so how do we deal with the special case for exam1 like network connection
Priority: N/A

ANON - 16:25
Q: so lambda is like how it works in python, just a condensed version of a function?
Priority: N/A
        Steven Haussmann - 16:25
        A: Specifically, it's a function that doesn't have a name assigned to it. You get it as a value.
        Steven Haussmann - 16:25
        A: You can, of course, immediately assign it to a variable.

ANON - 16:25
Q: If we append to a list in the Recursive Case, why does atom need to make it a list if the Input is a List? Dos append take Lists as arguments? I'm wondering why we could not have the flatten Base Case 2 as just ((atom? x) x)
Priority: N/A
        Steven Haussmann - 16:26
        A: append takes two lists as arguments, yes

ANON - 16:26
Q: Will there be office hours today?
Priority: N/A
        Ana L. Milanova - 16:33
        A: Yes.

ANON - 16:27
Q: should we write the program in drracket or another editor like sublime?
Priority: N/A

Steven Haussmann - 16:28

A: DrRacket is nice because it includes both an editor and an interpreter in one window. A good text editor would probably work plenty well, though (I use VSCode)

ANON - 16:31
Q: So is lambda a special keyword name for lambda functions? Or was the name of that lambda function from Lecture just lambda?
Priority: N/A

Ana L. Milanova - 16:34

A: "lambda" is a keyword, not a function name. It denotes a function value. We use the lambda key word to write anonymous functions, they come up very often in functional programming.

Ana L. Milanova - 16:35

A: We'll cover the Lambda Calculus in a week or so, that's where the keyword and its meaning come from.

ANON - 16:36
Q: ok thank you.
Priority: N/A

ANON - 16:37
Q: So on slide 4 you mentioned switch two conditions null? and atom?. Could you please explain what would happen if thery are switched?
Priority: N/A

Ana L. Milanova - ?

A: Each base case '() will contribute 1 to the atom count because '() is an atom, i.e., not a pair.

ANON - 16:39
Q: sllide 34 I made a typo...
Priority: N/A