


(Task 1 of 15) In this tutorial, we will learn *more* about definitions.

0

(Task 2 of 15) What is the result of running this program?

Lispy | 

Lispy [Run ]	<pre>(defvar x 1) (deffun (f y) (deffun (g) (defvar z 2) (+ x y z)) (g)) (+ (f 3) 4)</pre>	Python	<pre>x = 1 def f(y): def g(): z = 2 return x + y + z return g() print(f(3) + 4)</pre>
--	--	--------	---

10

2


You predicted the output correctly 🎉🎉🎉🎉


3

This program binds `x` to `1` and `f` to a function and then evaluates `(+ (f 3) 4)`. The value of `(+ (f 3) 4)` is the value of `(+ (g) 4)`, which is the value of `(+ (+ x y z) 4)`, which is the value of `(+ (+ 1 3 2) 4)`, which is `10`.

Click [here](#) to run this program in the Stacker.

(Task 3 of 15) What is the result of running this program?

Lispy | 

Lispy [Run ]	<pre>(defvar x 1) (deffun (f) x) (deffun (g) (defvar x 2) (f)) (g)</pre>	Python	<pre>x = 1 def f(): return x def g(): x = 2 return f() print(g())</pre>
--	--	--------	---

1

5

Please briefly explain why you think the answer is `1`.

6

The function `g` is returning a function defined in the global namespace, so it will not inherit the shadowed definition of `x`, and will return the `x` appropriate to the global scope, which is

1

7

You predicted the output correctly 🎉🎉🎉🎉

8

(g) evaluates to (f), which evaluates to 1 because f and x are both defined in the same block, and f does *not* get the x defined inside g.

Click [here](#) to run this program in the Stacker.

(Task 4 of 15) What is the result of running this program?

Lispy | 🍷

Lispy [Run ▶]	Python
(defvar x 1)	x = 1
(defun (f)	def f():
(defvar y 2)	y = 2
(defun (g)	def g():
(+ x y))	return x + y
(g))	return g()
(f)	print(f())

3

10

You predicted the output correctly 🎉🎉🎉🎉

11

(f) returns the value of (g). The value of (g) is the value of (+ x y), which is (+ 1 2), which is 3.

Click [here](#) to run this program in the Stacker.

(Task 5 of 15) What is the result of running this program?

Lispy | 🍷

Lispy [Run ▶]	Python
(defvar x 1)	x = 1
(defun (f y)	def f(y):
(defvar x 2)	x = 2
(+ x y))	return x + y
	print(f(0) + x)
(+ (f 0) x)	

3

13

You predicted the output correctly 🎉🎉🎉🎉

14

This program binds x to 1 and f to a function in the top-level block. It binds x to 2 in the

body of `f`. So, `(f 0)` evaluates to `(+ 2 0)`, which is `2`, and `(+ (f 0) x)` evaluates to `(+ 2 1)`, which is `3`.

Click [here](#) to run this program in the Stacker.

(Task 6 of 15) What is the result of running this program?

Lispy | 

```

Lispy [Run] Scala 3
(deffun (f x) (defvar y 1) (+ x y))
              val y = 1
              x + y
              println(f(2) + y)
(+ (f 2) y)

```

error

16

You predicted the output correctly 🎉🎉🎉

17

`(+ (f 2) y)` is evaluated in the top-level block, where `y` is not defined. So, this program errors.

Click [here](#) to run this program in the Stacker.

(Task 7 of 15) When we see a variable reference (e.g., the `x` in `(+ x 1)`), how do we find its value, if any?

18

We examine the current hash table that corresponds to the current environment, which is extended from the hash tables corresponding to the environment(s) above the current one.

19

(Task 8 of 15) Variable references follow the hierarchical structure of blocks.

20

If the variable is defined in the current block, we use that declaration.

Otherwise, we look up the block in which the current block appears, and so on recursively. (Specifically, if the current block is a function body, the next block will be the block in which the function definition is; if the current block is the top-level block, the next block will be the **primordial block**.)

If the current block is already the primordial block and we still haven't found a corresponding declaration, the variable reference errors.

The primordial block is a non-visible block enclosing the top-level block. This block

defines values and functions that are provided by the language itself.

Any feedback regarding these statements? Feel free to skip this question.

21

(You skipped the question.)

22

(Task 9 of 15) Now please scroll back and select 1-3 programs that make the point that

23

■ If the variable is declared in the current block, we use that declaration.

You don't need to select *all* such programs.

(You selected 3 programs)

24

Okay. How do these programs ([1](#),[12](#),[15](#)) support the point?

25

In each program we define a local variable, and then use it, sometimes shadowing the variable defined in a higher scope

26

(Task 10 of 15) Please select 1-3 programs that make the point that

27

■ Otherwise, we look up the block in which the current block appears, and so on recursively.

You don't need to select *all* such programs.

(You selected 1 programs)

28

Okay. How does this program ([12](#)) support the point?

29

Inside the function scope we use the defined variable, but outside the function scope we use the variable defined in the global scope. May have miss selected the right program

30

(Task 11 of 15) Please select 1-3 programs that make the point that

31

■ If the current block is already the primordial block and we still haven't found a corresponding declaration, the variable reference errors.

You don't need to select <i>all</i> such programs.	
(You selected 1 programs)	32
Okay. How does this program (15) support the point?	33
This program errors as it does not have a defined y in the global scope	34

(Task 12 of 15) The scope of a variable is the region of a program where we can refer to the variable. Technically, it includes the block in which the variable is defined (including sub-blocks) <i>except</i> the sub-blocks where the same name is re-defined. When the exception happens, that is, when the same name is defined in a sub-block, we say that the variable in the sub-block shadows the variable in the outer block.	35
We say a variable reference (e.g., "the x in (+ x 1)") is in the scope of a declaration (e.g., "the x in (defvar x 23)") if and only if the former refers to the latter.	
Any feedback regarding these statements? Feel free to skip this question.	36
(You skipped the question.)	37

(Task 13 of 15) Please select all statements that apply to the following program:	38
<pre>(defvar x 45) (defun (f) (defun (g) (+ x 1)) (defvar x 67) (+ x 2)) (f) (+ x 3)</pre>	
<input type="checkbox"/> The scope of the top-level `x` includes the whole program. <input checked="" type="checkbox"/> The scope of the top-level `x` includes `(+ x 3)`. <input checked="" type="checkbox"/> The top-level `x` shadows the `x` defined in `f`. <input type="checkbox"/> The `x` in `(+ x 1)` is in the scope of the `x` defined in `f`. <input checked="" type="checkbox"/> The `x` in `(+ x 2)` is in the scope of the `x` defined in `f`. <input type="checkbox"/> The `x` in `(+ x 3)` is in the scope of the `x` defined in `f`.	39
	40

You should also have chosen The `x` in `(+ x 1)` is in the scope of the `x` defined in `f`.

You should NOT have chosen The top-level `x` shadows the `x` defined in `f`.

The scope of the top-level `x` includes the whole program *except* the body of `f`. The scope of the `x` defined in `f` is the body of `f`.

The `x` defined in `f` shadows the top-level `x` rather than the other way around.

(Task 14 of 15) Here is a program that confused many students

Lispy | 



```

Lispy [Run] Scala 3
(defvar x 1)      val x = 1
(deffun (foo)    def foo =
  x)              x
(deffun (bar)    def bar =
  (defvar x 2)   val x = 2
  (foo))         foo
                println(bar)

(bar)

```

Please

1. Run this program in the stacker by clicking the green run button above;
2. The stacker would show how this program produces its result(s);
3. Keep clicking  until you reach a configuration that you find particularly helpful;
4. Click  to get a link to your configuration;
5. Submit your link below;

<https://www.cs.unb.ca/~bremner/teaching/cs4613/stacker/?>

42

`syntax=Lispy&randomSeed=smol-`

`tutor&hole=%E2%80%A2&nNext=3&program=%0A%28defvar+x+1%29%0A%28deffun+%28foo%29%0A++x%29%0A%28deffun+%28bar%29%0A++%28defvar+x+2%29%0A++%28foo%29%29%0A%0A%28bar%29%0A&readOnlyMode=`

(Task 15 of 15) Please write a couple of sentences to explain how your configuration explains the result(s) of the program.

43

Since we are in an environment that extends top level, but does not define `x`, so we use the definition of `x` from the top level

44

Let's review what we have learned in this tutorial.

Variable references follow the hierarchical structure of blocks.

If the variable is defined in the current block, we use that declaration.

Otherwise, we look up the block in which the current block appears, and so on recursively. (Specifically, if the current block is a function body, the next block will be the block in which the function definition is; if the current block is the top-level block, the next block will be the **primordial block**.)

If the current block is already the primordial block and we still haven't found a corresponding declaration, the variable reference errors.

The primordial block is a non-visible block enclosing the top-level block. This block defines values and functions that are provided by the language itself.

The **scope** of a variable is the region of a program where we can refer to the variable. Technically, it includes the block in which the variable is defined (including sub-blocks) *except* the sub-blocks where the same name is re-defined. When the exception happens, that is, when the same name is defined in a sub-block, we say that the variable in the sub-block **shadows** the variable in the outer block.

We say a variable reference (e.g., "the `x` in `(+ x 1)`") is **in the scope of** a declaration (e.g., "the `x` in `(defvar x 23)`") if and only if the former refers to the latter.

You have finished this tutorial 🎉🎉🎉

Please the finished tutorial to a PDF file so you can review the content in the future. **Your instructor (if any) might require you to submit the PDF.**

Start time: 1737144594458