Lispy |

(Task 1 of 9) In this tutorial, we will learn about **lambda expressions**, which are expressions that create functions.

The following program illustrates how to create a function.

Lispy [Run]

```
((lambda (n)
   (+ n 1))
 2)
```

Python

```
print((lambda n: n + 1)(2))
```

This program program produces `3`. The top-level block contains one expression, a function call. The only (actual) parameter of the function call is `2`. The function of the function call is created by

Lispy [Run]

```
(lambda (n)
  (+ n 1))
```

Python

```
print(lambda n: n + 1)
```

This function takes only one parameter `n`, and returns (the value of) `(+ n 1)`. So, the result of the whole program is the value of `(+ 2 1)`, which is `3`.

Any feedback regarding these statements? Feel free to skip this question. 1

*(You skipped the question.)* 2

---

Lispy |

(Task 2 of 9) What is the result of running this program?

Lispy [Run]

```
(deffun (f x)
  (lambda (y) (+ x y)))
(defvar x 0)
((f 2) 1)
```

JavaScript

```
function f(x) {
  return function (y) {
    return x + y;
  };
}
let x = 0;
console.log(f(2)(1));
```

3 4

Lispy |

You predicted the output correctly 🎉🎉🎉

This program is essentially the same as

Lispy [Run]

```
(deffun (f x)
  (deffun (fun y)
    (+ x y))
  fun)
```

Scala 3

```
def f(x : Int) =
  def fun(y : Int) =
    x + y
  fun
```

```
(defvar x 0)          val x = 0
((f 2) 1)             println(f(2)(1))
```

Click here to run this program in the Stacker.

---

Lispy |

(Task 3 of 9) What is the result of running this program?

Lispy [Run]

```
(deffun (bar y)
  (lambda (x)
    (+ x y)))
(defvar f (bar 2))
(defvar g (bar 4))
(f 2)
(g 2)
```

JavaScript

```
function bar(y) {
  return function (x) {
    return x + y;
  };
}
let f = bar(2);
let g = bar(4);
console.log(f(2));
console.log(g(2));
```

7

`4 6`

8

You predicted the output correctly 🎉🎉🎉

The value of `(bar 2)` is a lambda function defined in an environment where `y` is bound to `2`. The value of `(bar 4)` is *another* lambda function defined in an environment where `y` is bound to `4`. The two lambda functions are *different* values. So, the value of `(f 2)` is `12`, while the value of `(g 2)` is `52`.

Click here to run this program in the Stacker.

---

Lispy |

(Task 4 of 9) What is the result of running this program?

Lispy [Run]

```
(deffun (foobar)
  (defvar n 0)
  (lambda ()
    (set! n (+ n 1))
    n))
(defvar f (foobar))
(defvar g (foobar))

(f)
(f)
(g)
```

Pseudo

```
fun foobar():
  let n = 0
  return lam ():
    n = n + 1
    return n
  end
end
let f = foobar()
let g = foobar()
print(f())
print(f())
print(g())
```

10

`1 2 1`

```
1 2 1
```

11

You predicted the output correctly 🎉🎉🎉

Every time `foobar` is called, it creates a *new* environment that binds `n`. So, `f` and `g` have different bindings for the `n` variable. When `f` is called the first time, it mutates its binding for the `n` variable. So, the second call to `f` produces `2` rather than `1`. `g` has its own binding for the `n` variable, which still binds `n` to `0`. So, `(g)` produces `1` rather than `3`.

Click here to run this program in the Stacker.

Lispy |

(Task 5 of 9) What is the result of running this program?

Lispy [Run]

```
(defvar x 1)
(deffun (f)
  (lambda (y)
    (+ x y)))
(defvar g (f))
(set! x 2)
(g 0)
```

Pseudo

```
let x = 1
fun f():
  return lam (y):
    return x + y
  end
end
let g = f()
x = 2
print(g(0))
```

13

`2`

14

You predicted the output correctly 🎉🎉🎉

`x` is bound to `1`. `g` is bound to the lambda function. `(set! x 2)` binds `x` to `2`. So, the value of `(g 0)` is the value of `(+ 2 0)`, which is `2`.

Click here to run this program in the Stacker.

15

(Task 6 of 9) `(deffun (f x y z) body)` is a shorthand for `(defvar f (lambda (x y z) body))`.

16

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

17

*(You skipped the question.)*

Lispy |

(Task 7 of 9) Please rewrite this function definition with as a variable definition that binds

(Task 7 of 9) Please rewrite this function definition with as a variable definition that binds lambda function.

Lispy [Run]

```
(deffun (f x) (+ x 1))
```

Scala 3

```
def f(x : Int) =
  x + 1
```

(defvar f (lambda (x) (+ x 1)))

19

The correct answer is `(defvar f (lambda (x) (+ x 1)))`.

The following are common wrong answers:

- `(deffun f (lambda (x) (+ x 1)))`, which didn't replace the definition keyword
- `(defvar f (lambda (f x) (+ x 1)))`, which makes `f` a function that takes two parameters rather than one

20

---

Lispy |

(Task 8 of 9) Here is a program that confused many students

Lispy [Run]

```
(deffun (foo n)
  (deffun (bar)
    (set! n (+ n 1))
    n)
  bar)
(defvar f (foo 0))
(defvar g (foo 0))

(f)
(f)
(g)
```

Pseudo

```
fun foo(n):
  fun bar():
    n = n + 1
    return n
  end
  return bar
end
let f = foo(0)
let g = foo(0)
print(f())
print(f())
print(g())
```

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 ⏭ Next until you reach a configuration that you find particularly helpful;
4. Click 🔗 Share This Configuration to get a link to your configuration;
5. Submit your link below;

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

22

syntax=Lispy&randomSeed=smol-tutor&hole=%E2%80%A2&nNext=18&program=%0A%28deffun+%28foo+n%29%0A++%28deffun+%28bar%29%0A++++%28set%21+n+%28%2B+n+1%29%29%0A+++++n%29%0A++bar%29%0A%28defvar+f+%28foo+0%29%29%0A%28defvar+g+%28foo+0%29%29%0A%0A%28f%29%0A%28f%29%0A%28g%29%0A&readOnlyMode=

(Task 9 of 9) Please write a couple of sentences to explain how your configuration explains the result(s) of the program. 23

We can see that now that the execution context of operating on the environment of f is finished, we operate in the execution environment of g, setting n in the environment defined in the defvar g 24

Lispy |

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

In this tutorial, we will learn about **lambda expressions**, which are expressions that create functions.

The following program illustrates how to create a function.

Lispy [Run]

```
((lambda (n)
   (+ n 1))
 2)
```

Pseudo

```
print((lam (n):
  return n + 1
end)(2))
```

This program program produces `3`. The top-level block contains one expression, a function call. The only (actual) parameter of the function call is `2`. The function of the function call is created by

Lispy [Run]

```
(lambda (n)
  (+ n 1))
```

Pseudo

```
print(lam (n):
  return n + 1
end)
```

This function takes only one parameter `n`, and returns (the value of) `(+ n 1)`. So, the result of the whole program is the value of `(+ 2 1)`, which is `3`.

`(deffun (f x y z) body)` is a shorthand for `(defvar f (lambda (x y z) body))`.

You have finished this tutorial 🎉🎉🎉

Please print 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: 1737765907079