>>> True and 13? True-- Not quite. Try again! --? 13-- OK! -->>> False or 0? 0-- OK! -->>> not 10? False-- OK! -->>> not None? True-- OK! -----------------------------------------------------------------------The Truth Will Prevail > Suite 2 > Case 1(cases remaining: 3)What would Python display? If you get stuck, try it out in the Pythoninterpreter!>>> True and 1 / 0 # If this errors, just type Error.? Error-- OK! -->>> True or 1 / 0 # If this errors, just type Error.? True-- OK! -->>> -1 and 1 > 0 # If this errors, just type Error.? True-- OK! -->>> -1 or 5? True-- Not quite. Try again! --? -1-- OK! -->>> (1 + 1) and 1 # If this errors, just type Error. If this is blank, just type Nothing.? 1-- OK! -----------------------------------------------------------------------The Truth Will Prevail > Suite 2 > Case 2(cases remaining: 2)What would Python display? If you get stuck, try it out in the Pythoninterpreter!>>> print(3) or ""(line 1)? 3(line 2)? False-- Not quite. Try again! --(line 1)? 3(line 2)? Nothing-- Not quite. Try again! --(line 1)? 3(line 2)? -- Not quite. Try again! --(line 1)? 3(line 2)? ''-- OK! -----------------------------------------------------------------------The Truth Will Prevail > Suite 3 > Case 1(cases remaining: 1)What would Python display? If you get stuck, try it out in the Pythoninterpreter!>>> def f(x):... if x == 0:... return "I am zero!"... elif x > 0:... return "Positive!"... else:... return "">>> 0 or f(1)? "Positive!" -- OK! -->>> f(0) or f(-1)? True-- Not quite. Try again! --? "I am zero!"-- OK! -->>> f(0) and f(-1)? ""-- OK! -----------------------------------------------------------------------OK! All cases for The Truth Will Prevail unlocked.Cannot backup when running ok with --local.(venv) PS E:\CS61A\Lab\lab02> py ok -q hof-wwpd -u --local

=====================================================================

Assignment: Lab 2

OK, version v1.18.1

=====================================================================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Unlocking tests

At each "? ", type what you would expect the output to be.

Type exit() to quit

---------------------------------------------------------------------

Higher Order Functions > Suite 1 > Case 1

(cases remaining: 1)

What would Python display? If you get stuck, try it out in the Python

interpreter!

>>> # If Python displays <function...>, type Function, if it errors type Error, if it displays nothing type Nothing

>>> def cake():

... print('beets')

... def pie():

... print('sweets')

... return 'cake'

... return pie

>>> chocolate = cake()

? beets

-- OK! --

>>> chocolate

? sweets

-- Not quite. Try again! --

? Error

-- Not quite. Try again! --

? Function

-- OK! --

>>> chocolate()

(line 1)? beets

-- Not quite. Try again! --

(line 1)? sweets

(line 2)? 'cake'

-- OK! --

>>> more\_chocolate, more\_cake = chocolate(), cake

?

-- Not quite. Try again! --

? sweets

-- OK! --

>>> more\_chocolate

? 'cake'

-- OK! --

>>> # Reminder: cake, more\_cake, and chocolate were defined/assigned in the code above!

>>> # It might be helpful to refer to their definitions on the assignment website so you don't have to scroll as much!

>>> def snake(x, y):

... if cake == more\_cake:

... return chocolate

... else:

... return x + y

>>> snake(10, 20)

? Function

-- OK! --

>>> snake(10, 20)()

(line 1)? 'cake'

-- Not quite. Try again! --

(line 1)? sweets

(line 2)? 'cake'

-- OK! --

>>> cake = 'cake'

>>> snake(10, 20)

? 30

-- OK! --

---------------------------------------------------------------------

OK! All cases for Higher Order Functions unlocked.

Cannot backup when running ok with --local.

(venv) PS E:\CS61A\Lab\lab02> py ok -q short-circuit -u --local

=====================================================================

Assignment: Lab 2

OK, version v1.18.1

=====================================================================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Unlocking tests

At each "? ", type what you would expect the output to be.

Type exit() to quit

---------------------------------------------------------------------

The Truth Will Prevail > Suite 1 > Case 1

(cases remaining: 4)

-- Already unlocked --

---------------------------------------------------------------------

The Truth Will Prevail > Suite 2 > Case 1

(cases remaining: 3)

-- Already unlocked --

---------------------------------------------------------------------

The Truth Will Prevail > Suite 2 > Case 2

(cases remaining: 2)

-- Already unlocked --

---------------------------------------------------------------------

The Truth Will Prevail > Suite 3 > Case 1

(cases remaining: 1)

-- Already unlocked --

---------------------------------------------------------------------

OK! All cases for The Truth Will Prevail unlocked.

Cannot backup when running ok with --local.

(venv) PS E:\CS61A\Lab\lab02> py ok -q lambda -u --local

=====================================================================

Assignment: Lab 2

OK, version v1.18.1

=====================================================================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Unlocking tests

At each "? ", type what you would expect the output to be.

Type exit() to quit

---------------------------------------------------------------------

Lambda the Free > Suite 1 > Case 1

(cases remaining: 5)

Q: Which of the following statements describes a difference between a def statement and a lambda expression?

Choose the number of the correct choice:

0) A def statement can only have one line in its body.

1) A lambda expression cannot have more than two parameters.

2) A lambda expression does not automatically bind the function object that it returns to an intrinsic name.

3) A lambda expression cannot return another function.

? 2

-- OK! --

---------------------------------------------------------------------

Lambda the Free > Suite 1 > Case 2

(cases remaining: 4)

Q: How many parameters does the following lambda expression have?

lambda a, b: c + d

Choose the number of the correct choice:

0) Not enough information

1) two

2) three

3) one

? 1

-- OK! --

---------------------------------------------------------------------

(cases remaining: 3)

Q: When is the return expression of a lambda expression executed?

Choose the number of the correct choice:

0) When the lambda expression is evaluated.

1) When you pass the lambda expression into another function.

2) When the function returned by the lambda expression is called.

3) When you assign the lambda expression to a name.

? 2

-- OK! --

---------------------------------------------------------------------

Lambda the Free > Suite 2 > Case 1

(cases remaining: 2)

What would Python display? If you get stuck, try it out in the Python

interpreter!

>>> # If Python displays <function...>, type Function, if it errors type Error, if it displays nothing type Nothing

>>> lambda x: x # A lambda expression with one parameter x

? Function

-- OK! --

>>> a = lambda x: x # Assigning a lambda function to the name a

>>> a(5)

? 5

-- OK! --

>>> (lambda: 3)() # Using a lambda expression as an operator in a call exp.

? 3

-- OK! --

>>> b = lambda x, y: lambda: x + y # Lambdas can return other lambdas!

>>> c = b(88, 43)

>>> c

? 151

-- Not quite. Try again! --

? 131

-- Not quite. Try again! --

? Function

-- OK! --

>>> c()

? 131

-- OK! --

>>> d = lambda f: f(4) # They can have functions as arguments as well

>>> def square(x):

... return x \* x

>>> d(square)

? 16

-- OK! --

---------------------------------------------------------------------

Lambda the Free > Suite 2 > Case 2

(cases remaining: 1)

What would Python display? If you get stuck, try it out in the Python

interpreter!

>>> # Try drawing an environment diagram if you get stuck!

>>> higher\_order\_lambda = lambda f: lambda x: f(x)

>>> g = lambda x: x \* x

>>> higher\_order\_lambda(2)(g) # Which argument belongs to which function call?

? 4

-- Not quite. Try again! --

? Error

-- OK! --

>>> higher\_order\_lambda(g)(2)

? 4

-- OK! --

>>> call\_thrice = lambda f: lambda x: f(f(f(x)))

>>> call\_thrice(lambda y: y + 1)(0)

? 3

-- OK! --

>>> print\_lambda = lambda z: print(z) # When is the return expression of a lambda expression executed?

>>> print\_lambda

? Function

-- OK! --

>>> one\_thousand = print\_lambda(1000)

? 1000

-- OK! --

>>> one\_thousand # What did the call to print\_lambda return?

? 1000

-- Not quite. Try again! --

? None

-- Not quite. Try again! --

? Nothing

-- OK! --

---------------------------------------------------------------------

OK! All cases for Lambda the Free unlocked.