Exercise 1

6/6 points (graded)

**ESTIMATED TIME TO COMPLETE: 12 minutes**  
**Note that you will have to answer all questions before you can click the Check button.**

**Part 1: Function Types**

For each of the following functions, specify the type of its **return**. You can assume each function is called with an appropriate argument, as specified by its docstring.

If the output can be either an int or a float, select num, which isn't a real Python type, but which we'll use to indicate that either basic numeric type is legal.

In fact, in Python, booleans True and False can be operated on as if they were the integers 1 and 0; but it is ugly and confusing to take advantage of this fact, and we will resolutely pretend that it isn't true.

[What are those lines under the function definitions?](https://courses.edx.org/xblock/block-v1:MITx+6.00.1x+1T2021+type@vertical+block@2f571a585a7b491a9537d6091d517ead?show_title=0&show_bookmark_button=0&recheck_access=1&view=student_view&format=Finger%20Exercises)

def a(x):

'''

x: int or float.

'''

return x + 1

Indicate the type of the output that the function a will yield .

def b(x):

'''

x: int or float.

'''

return x + 1.0

Indicate the type of the output that the function b will yield. 

def c(x, y):

'''

x: int or float.

y: int or float.

'''

return x + y

Indicate the type of the output that the function c will yield. 

def d(x, y):

'''

x: Can be int or float.

y: Can be int or float.

'''

return x > y

Indicate the type of the output that the function d will yield. 

def e(x, y, z):

'''

x: Can be int or float.

y: Can be int or float.

z: Can be int or float.

'''

return x >= y and x <= z

Indicate the type of the output that the function e will yield. 

def f(x, y):

'''

x: int or float.

y: int or float

'''

x + y - 2

Indicate the type of the output that the function f will yield. 

Any function lacking a ‘return’ statement will return None by default when it has finished running the code. Sometimes this is done deliberately; other times it is a mistake (as it appears to be in this example).

Exercise 2

14/14 points (graded)

**ESTIMATED TIME TO COMPLETE: 12 minutes**  
**Note that you will have to answer all questions before you can click the Check button.**

**Transcript**

You have the following function definitions:

def a(x):

'''

x: int or float.

'''

return x + 1

def b(x):

'''

x: int or float.

'''

return x + 1.0

def c(x, y):

'''

x: int or float.

y: int or float.

'''

return x + y

def d(x, y):

'''

x: Can be of any type.

y: Can be of any type.

'''

return x > y

def e(x, y, z):

'''

x: Can be of any type.

y: Can be of any type.

z: Can be of any type.

'''

return x >= y and x <= z

def f(x, y):

'''

x: int or float.

y: int or float

'''

x + y - 2

Below is a transcript of a session with the Python shell. Provide the type and value of the expressions being evaluated. If evaluating an expression would cause an error, select NoneType and write 'error' in the box. If the value of an expression is a function, select function as the type and write 'function' in the box.

1. a(6)





1. a(-5.3)





1. a(a(a(6)))





1. c(a(1), b(1))





1. d('apple', 11.1)





1. e(a(3), b(4), c(3, 4))





1. f





Exercise 3

8/8 points (graded)

**ESTIMATED TIME TO COMPLETE: 8 minutes**  
**Note that you will have to answer all questions before you can click the Check button.**

Below is a transcript of a session with the Python shell. Provide the type and value of the expressions being evaluated. If evaluating an expression would cause an error, select NoneType and write 'error' in the box. If the result is a function, select function and write 'function' in the box. As always, try to do this problem by hand before turning to your interpreter.

Assume the following definitions have been made:

def a(x, y, z):

if x:

return y

else:

return z

def b(q, r):

return a(q>r, q, r)

1. a(False, 2, 3)





1. b(3, 2)





1. a(3>2, a, b)





1. b(a, b)





Exercise 4

4/4 points (graded)

**ESTIMATED TIME TO COMPLETE: 6 minutes**

Below is a transcript of a session with the Python shell. Provide the type and value of the expressions being evaluated. If evaluating an expression would cause an error, select NoneType and write 'error' in the box. If the result is a function, select function and write 'function' in the box.

To get the most out of this problem, try to figure out the answers by reading the code, not running it. Run the code in your interpreter only after you've checked your answers a few times.

**Hint:** If you are confused, you may find it helpful to draw out an environment diagram similar to what was presented in lecture.

>>> a = 10

>>> def f(x):

return x + a

>>> a = 3

>>> f(1)





>>> x = 12

>>> def g(x):

x = x + 1

def h(y):

return x + y

return h(6)

>>> g(x)





Exercise 5

4/4 points (graded)

**ESTIMATED TIME TO COMPLETE: 10 minutes**

Enter the value of the expressions below.

To get the most out of this problem, try to figure out the answers by reading the code, not running it. Run the code only after you've used up a few of your checks.

**Hint:** If you are confused, you may find it helpful to draw out an environment diagram similar to what was presented in lecture.

1. def foo(x, y = 5):
2. def bar(x):
3. return x + 1
4. return bar(y \* 2)
6. foo(3)



1. def foo(x, y = 5):
2. def bar(x):
3. return x + 1
4. return bar(y \* 2)
6. foo(3, 0)



1. def foo (x):
2. def bar (z, x = 0):
3. return z + x
4. return bar(3, x)
5. foo(2)



1. def foo (x):
2. def bar (z, x = 0):
3. return z + x
4. return bar(3)
5. foo(5)



### Exercise 6

30 points possible (graded)

**ESTIMATED TIME TO COMPLETE: 20 minutes**

As we'll see in subsequent lectures, everything in Python is an *object*. Objects are special because we can associate special functions, referred to as *object methods*, with the object. In this problem you'll be working with string objects, and their built-in methods.

A complete description of the methods available to string objects can be found in the [Python library reference on string methods](http://docs.python.org/library/stdtypes.html#string-methods).

In this exercise, we want you to get some experience in using methods as functions. The convention for object methods is to use the "dot" notation, so that if s is a string, evaluating s.upper will return the actual function, and evaluating s.upper() will cause the function itself to be evaluated (in this case it returns a new string, since strings are immutable) with every character now in upper case. An example of this follows:

>>> s = 'abc'

>>> s.capitalize

<built-in method capitalize of str object at 0x104c35878>

>>> s.capitalize()

'Abc'

For each of the expressions in this problem, specify its type and value. If it generates an error, select type 'NoneType' and put the word 'error' in the box for the value. If it would be a function, select type 'function' and put the word 'function' in the box for the value.

Assume we've made the following assignments:

> str1 = 'exterminate!'

> str2 = 'number one - the larch'

Assume that the expressions are evaluated in the order shown - that is, each problem part is evaluated directly after the previous problem part(s).

1. str1.upper





1. str1.upper()





*Note:* str1 variable is not assigned new string value returned by .upper() object method.

1. str1





1. str1.isupper()





1. str1.islower()





1. str2 = str2.capitalize()  
   str2





*Note:* str2 = …

1. str2.swapcase()





*Note:* str2 variable is not assigned new string value returned by .swapcase() object method.

1. str1.index('e')





1. str2.index('n')





1. str2.find('n')





1. str2.index('!')





1. str2.find('!')





*Note:* Be sure to make note of the difference between the find and index string methods...

1. str1.count('e')





1. str1 = str1.replace('e', '\*')  
   str1





1. str2.replace('one', 'seven')



