1 Execute Python expressions

Execute each of the following expressions.

- If it executes without an error, then:
  - `value` state the value that it evaluates to

- If it suffers an error during evaluation:
  - `error` describe the error (in one phrase — a brief explanation in your own words)
  - `frame` state the name of the current environment frame: “global” or a function name
  - `operator` state the operator that caused the error
  - `arguments` state the values to which the operator was being applied

Your answer will contain either part “value”, or parts “error”, “frame”, “operator”, and “arguments”.

1. \((4.0 + 5) \times 6\)
   - `value 54.0`
   - `error`
   - `frame`
   - `operator`
   - `arguments`

2. `len(str(5 \times 2)) + "10"`
   - `value`
   - `error cannot add an integer and a string`
   - `frame global`
   - `operator +`
   - `arguments 2, "10"`
Execute Python statements

Execute each of the following code snippets, just as if they were written in a program. (Each one is a sequence of statements.) Your answer to each question contains up to 5 parts.

output  write any output that it prints (before any error). This part might be blank. If it does not print any output.

If there is an error during execution:

error  describe the error (in one phrase — a brief explanation in your own words)

frame  state the name of the current environment frame: “global” or a function name

operator  state the operator that caused the error

arguments  state the values to which the operator was being applied

Your answer will contain either only part “output”, or parts “output”, “error”, “frame”, “operator”, and “arguments”.

3. a = [1, 2]
b = []
b = b + a
b = b + a
a.append(3)
print b

output [1, 2, 1, 2]
error
frame
operator
arguments

4. a = [1, 2]
b = []
b.append(a)
b.append(a)
a.append(3)
print b

output [[1, 2, 3], [1, 2, 3]]
error
frame
operator
arguments
5. \texttt{lst = [1, 2, 3]}
   \texttt{myvar = lst[0]}
   \texttt{lst[0] = 18}
   \texttt{print lst}
   \texttt{print myvar}

   \textbf{output} \texttt{[18, 2, 3]}
   \texttt{1}

   \textbf{error}
   \textbf{frame}
   \textbf{operator}
   \textbf{arguments}

6. \texttt{myvar = 18}
   \texttt{lst = [myvar, 2, 3]}
   \texttt{myvar = 22}
   \texttt{print lst}
   \texttt{print myvar}

   \textbf{output} \texttt{[18, 2, 3]}
   \texttt{22}

   \textbf{error}
   \textbf{frame}
   \textbf{operator}
   \textbf{arguments}
7. `plane = ("Passengers", "Luggage")`
   `plane[1] = "Snakes"
   print plane

   output
   error cannot change a tuple

   frame global
   operator tuple element assignment
   arguments ("Passengers", "Luggage") , "Snakes"

8. `numb3rs = ([1, 2, 3], [4, 5, 6])`
   `numb3rs[0][2] = 0
   print numb3rs

   output ([1, 2, 0], [4, 5, 6])
   error
   frame
   operator
   arguments

Note that an immutable object can reference mutable objects. The immutable object’s elements are always the same Python object, but aren’t always the same abstract value.
9. for x in [1, 2]:
    for y in [3, x]:
        print x, y

    **output**
    1 3
    1 1
    2 3
    2 2

**error**

**frame**

**operator**

**arguments**

10. bar = 1
    def foo():
        bar = 2
        foo = 3
        return bar
    foo()
    print foo()
    print bar

    **output**
    2
    1

    **error**

    **frame**

    **operator**

    **arguments**
11. \( \text{total} = 0 \)

```python
def sum1(n):
    total = 0
    for i in range(n):
        total = total + i
    return total

def sum2(n):
    total = 0
    for i in range(n):
        total = total + i
    print total

print sum1(5)
print total
print sum2(5)
print total
```

**output**

```
10
0
10
None
0
```

**error**

**frame**

**operator**

**arguments**
12. def f1(n):
    print "A"
    return 2

    print "B"

    def f2():
        return 1
        print "C"

    print f1(f2())

    output B
    A
    2

    error
    frame
    operator
    arguments

13. x = 22
    def reset_x():
        x = 0
        reset_x()
    print x

    output 22
    error
    frame
    operator
    arguments

14. a = [42, 39, 123]
b = sorted(a, reverse=True)
a[0] = b
b[0] = 7
b.sort()
print a

    output [[7, 39, 42], 39, 123]
    error
    frame
    operator
    arguments