Week 7
Course Review & Advanced Topics

Recursive Functions

Instructor: Asst. Prof. LIN Shang-Wei
Email: shang-wei.lin@ntu.edu.sg

Topic Outline

How to Write Recursive Functions

How to Trace Recursive Functions

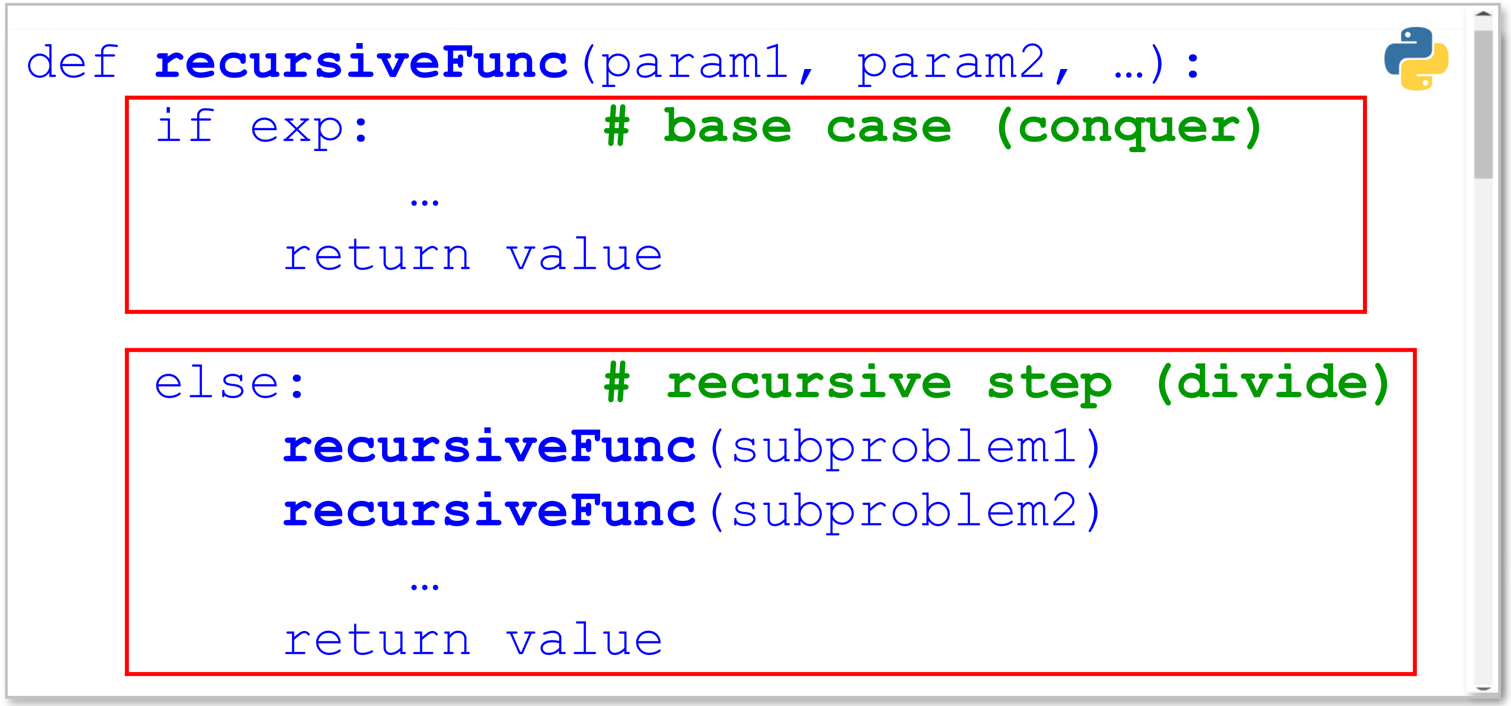
How to Write Recursive Functions

What Is A Recursive Function?

Recursive Function

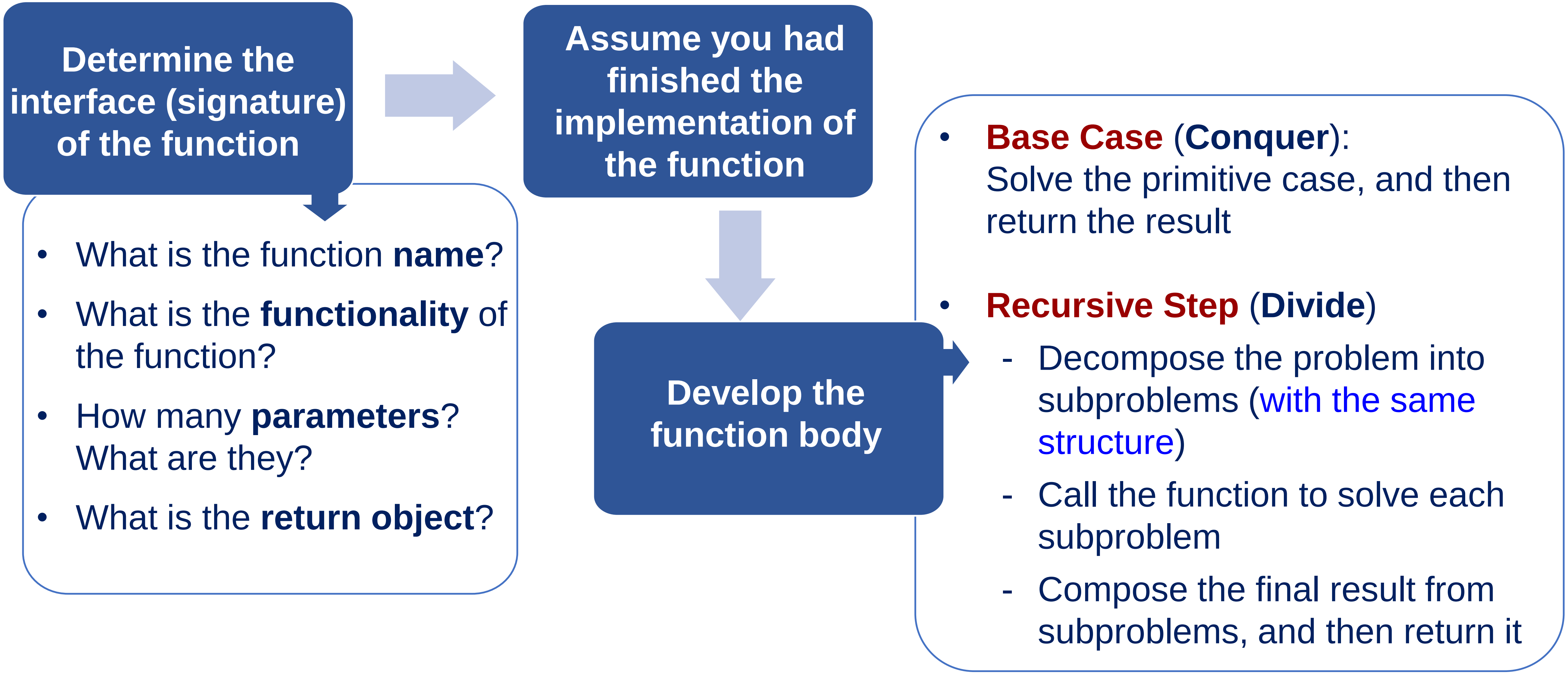
- A function that invokes itself
- naturally supports divide-and-conquer

General Form:



```
def recursiveFunc(param1, param2, …):
    if exp:           # base case (conquer)
            …
        return value

    else:             # recursive step (divide)
        recursiveFunc(subproblem1)
        recursiveFunc(subproblem2)
            …
        return value
```

How to Write a Recursive Function?



Determine the interface (signature) of the function

- What is the function **name**?
- What is the **functionality** of the function?
- How many **parameters**? What are they?
- What is the **return object**?

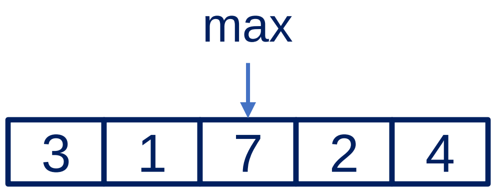
Assume you had finished the implementation of the function

Develop the function body

- **Base Case** (**Conquer**):
  Solve the primitive case, and then return the result
- **Recursive Step** (**Divide**)
  - Decompose the problem into subproblems (with the same structure)
  - Call the function to solve each subproblem
  - Compose the final result from subproblems, and then return it

Example

Develop a recursive function to find out the maximum element in a list



Step 1: Determine the name, functionality, and interface of the function

```python
def maxNum(aList):
```

Functionality: To find to the maximum element in a list

Input: `aList`, the target list of integers

Output: The maximum integer in `aList`

Step 2: Assume you had finished the implementation of the function

So that we can use `maxNum()` to solve subproblems!!

Example (cont.)

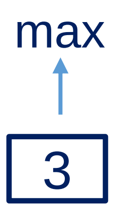
Step 3: Develop the function body of `maxNum()`

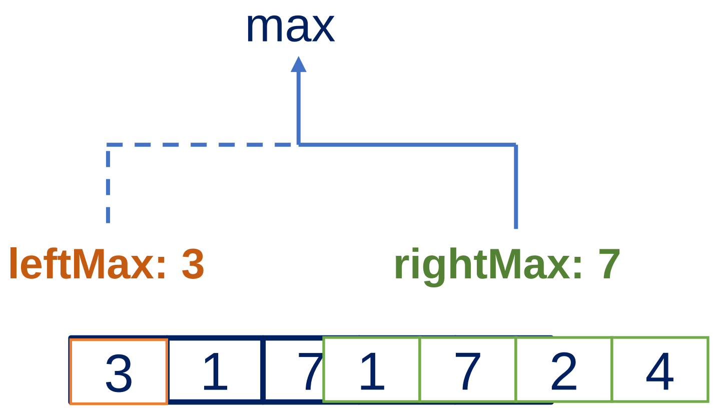
- Base Case (Conquer):

  If `aList` has only one element

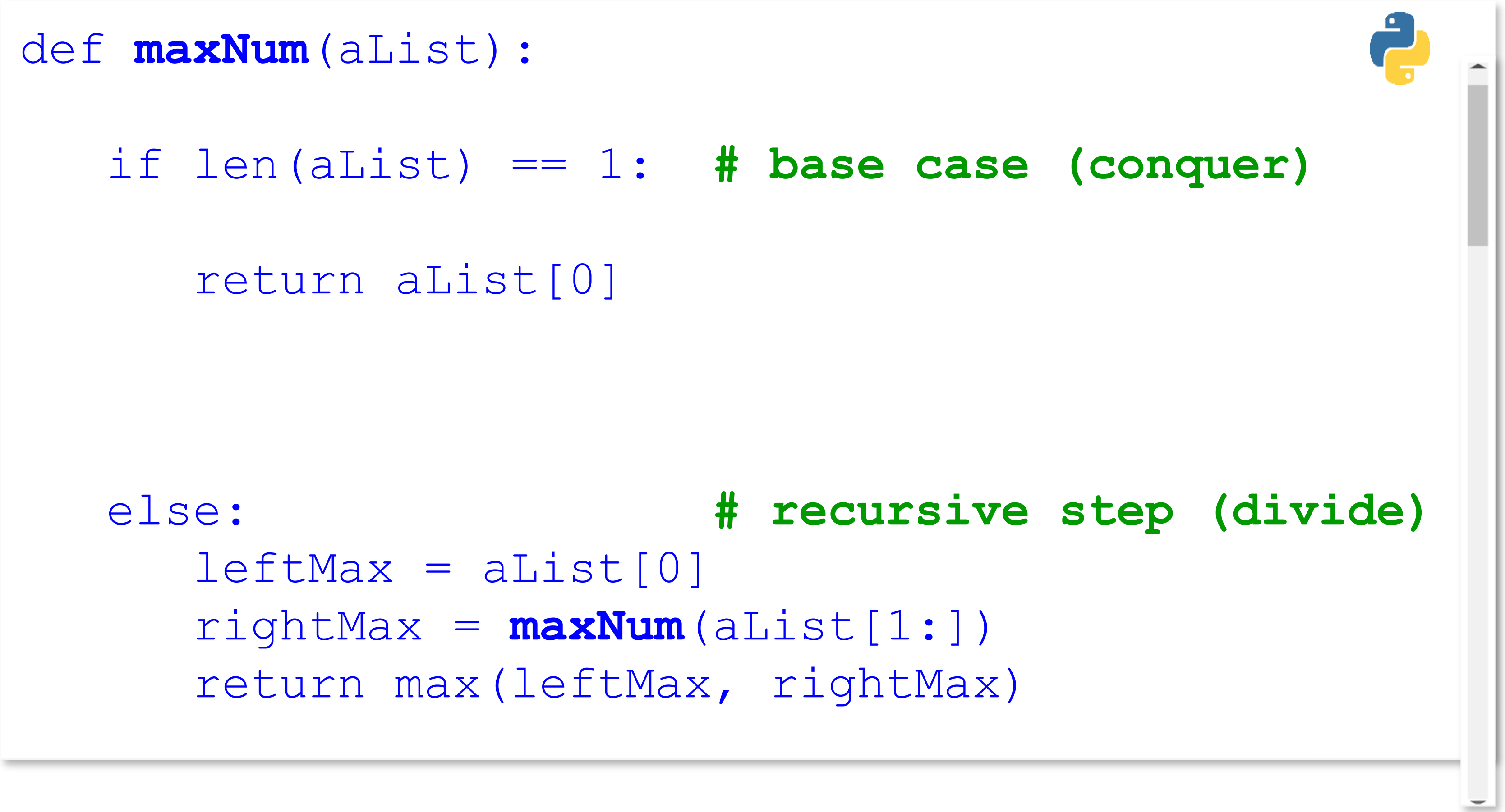
- Recursive Step (Divide):

  If `aList` has more than one element

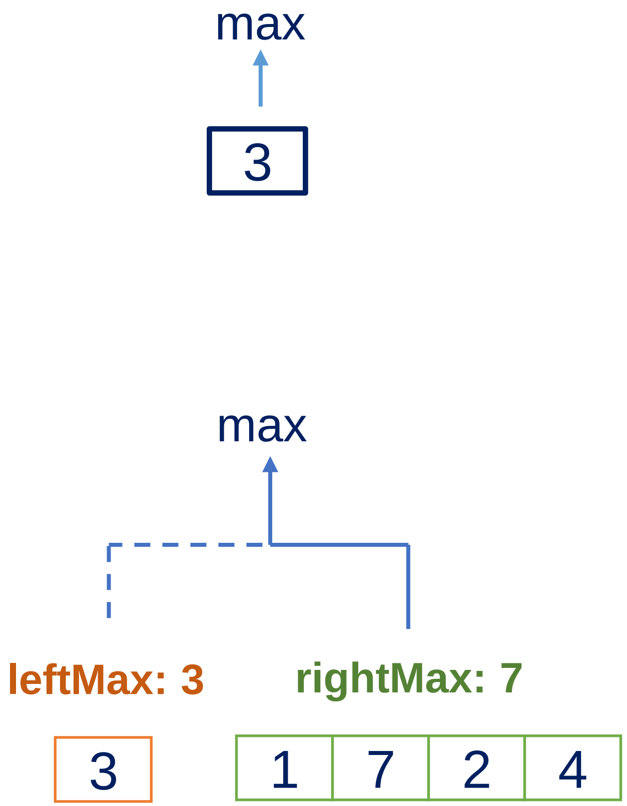




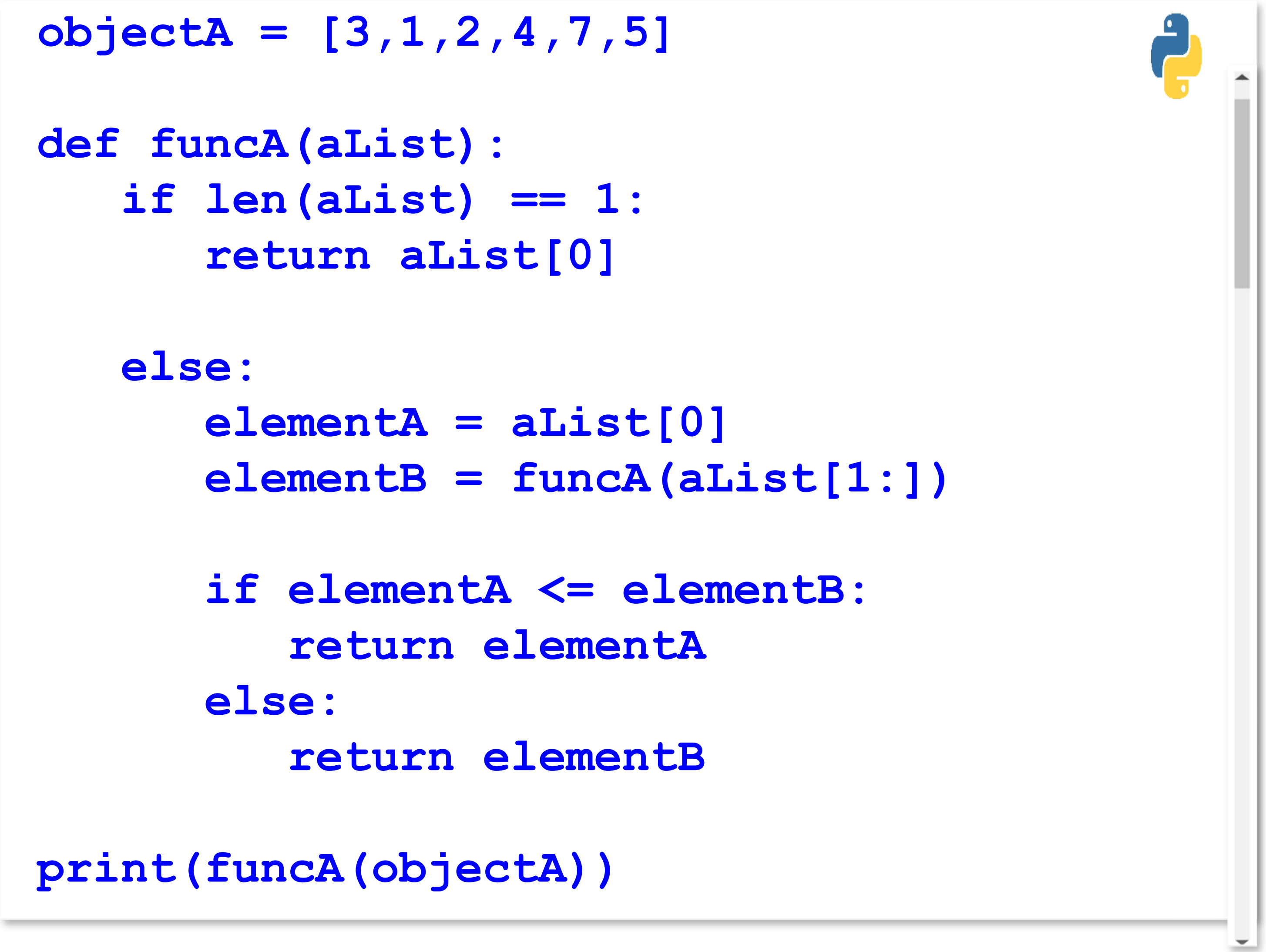
Example (cont.)



```
def maxNum(aList):

    if len(aList) == 1:   # base case (conquer)

        return aList[0]



    else:                 # recursive step (divide)
        leftMax = aList[0]
        rightMax = maxNum(aList[1:])
        return max(leftMax, rightMax)
```



Q1: What is the output of the following Python program?



```python
objectA = [3,1,2,4,7,5]

def funcA(aList):
   if len(aList) == 1:
      return aList[0]

   else:
      elementA = aList[0]
      elementB = funcA(aList[1:])

      if elementA <= elementB:
         return elementA
      else:
         return elementB

print(funcA(objectA))
```

A. 3

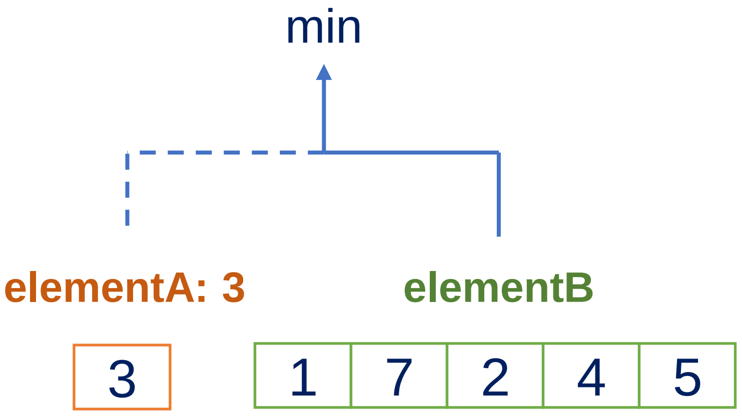
✓ B. 1

C. 2

D. 4

E. 7

F. 5



min

elementA: 3

elementB

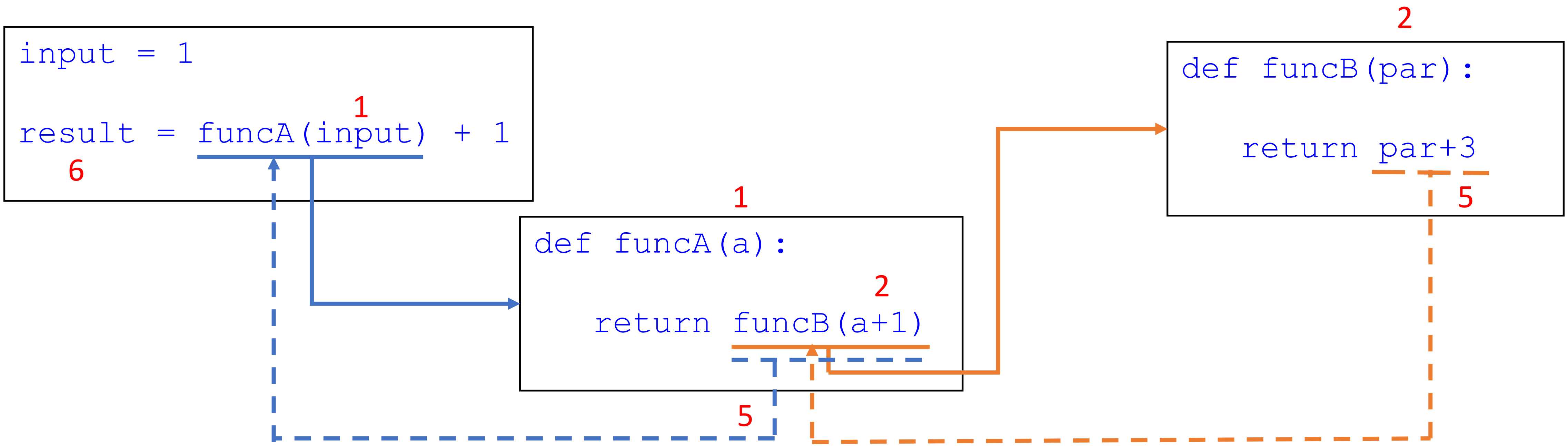
3

1	7	2	4	5

How to Trace A Recursive Function?

How to Trace A Recursive Function?

Try to draw the CALL GRAPH!!!

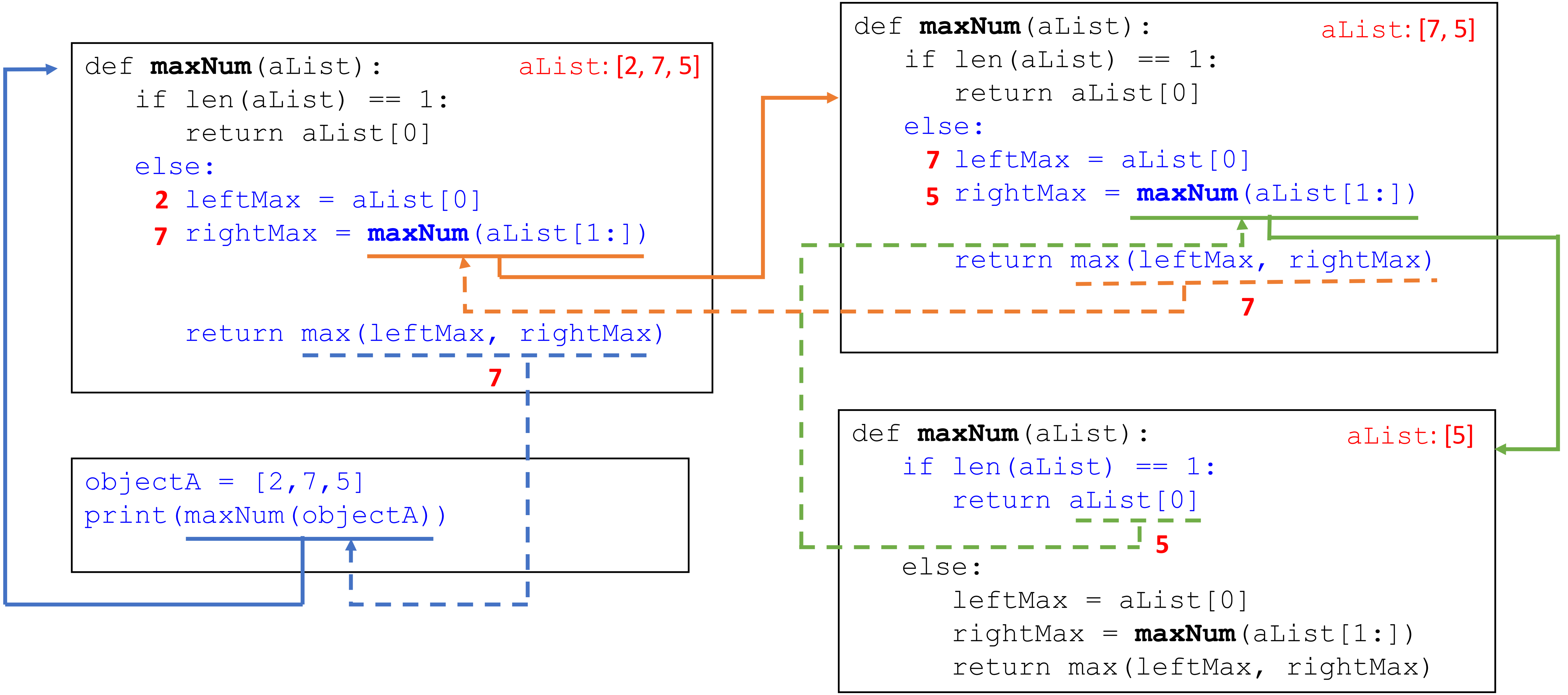


```
input = 1

result = funcA(input) + 1
```

```
def funcA(a):

    return funcB(a+1)
```

```
def funcB(par):

    return par+3
```

Example



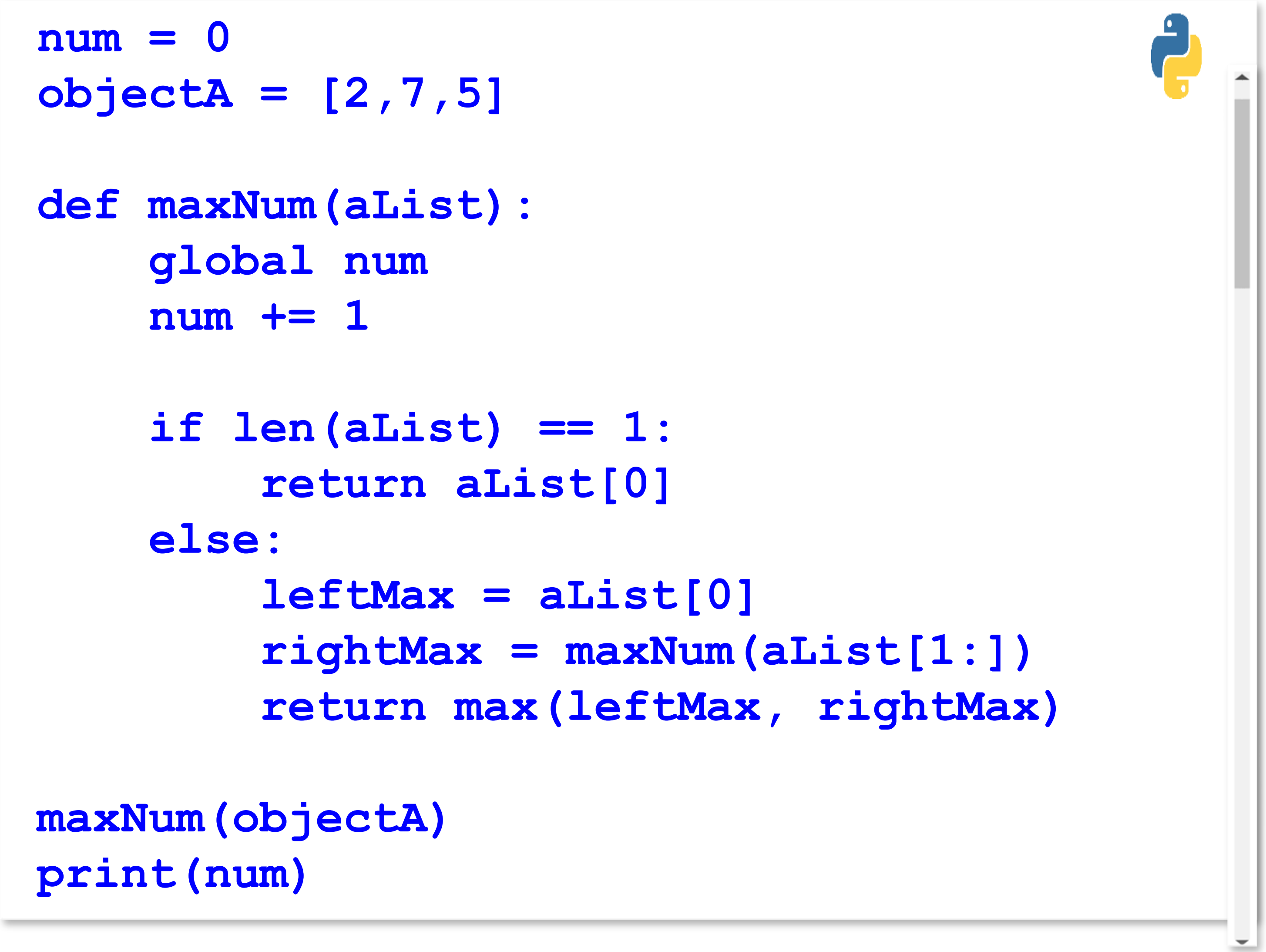
```
def maxNum(aList):                    aList: [2, 7, 5]
    if len(aList) == 1:
        return aList[0]
    else:
     2  leftMax = aList[0]
     7  rightMax = maxNum(aList[1:])


        return max(leftMax, rightMax)
                         7
```

```
def maxNum(aList):                    aList: [7, 5]
    if len(aList) == 1:
        return aList[0]
    else:
     7  leftMax = aList[0]
     5  rightMax = maxNum(aList[1:])

        return max(leftMax, rightMax)
                         7
```

```
def maxNum(aList):                    aList: [5]
    if len(aList) == 1:
        return aList[0]
                     5
    else:
        leftMax = aList[0]
        rightMax = maxNum(aList[1:])
        return max(leftMax, rightMax)
```

```
objectA = [2,7,5]
print(maxNum(objectA))
```

Q2: What is the output of the following Python program?



```
num = 0
objectA = [2,7,5]

def maxNum(aList):
    global num
    num += 1

    if len(aList) == 1:
        return aList[0]
    else:
        leftMax = aList[0]
        rightMax = maxNum(aList[1:])
        return max(leftMax, rightMax)

maxNum(objectA)
print(num)
```

A. 2

B. 7

C. 5

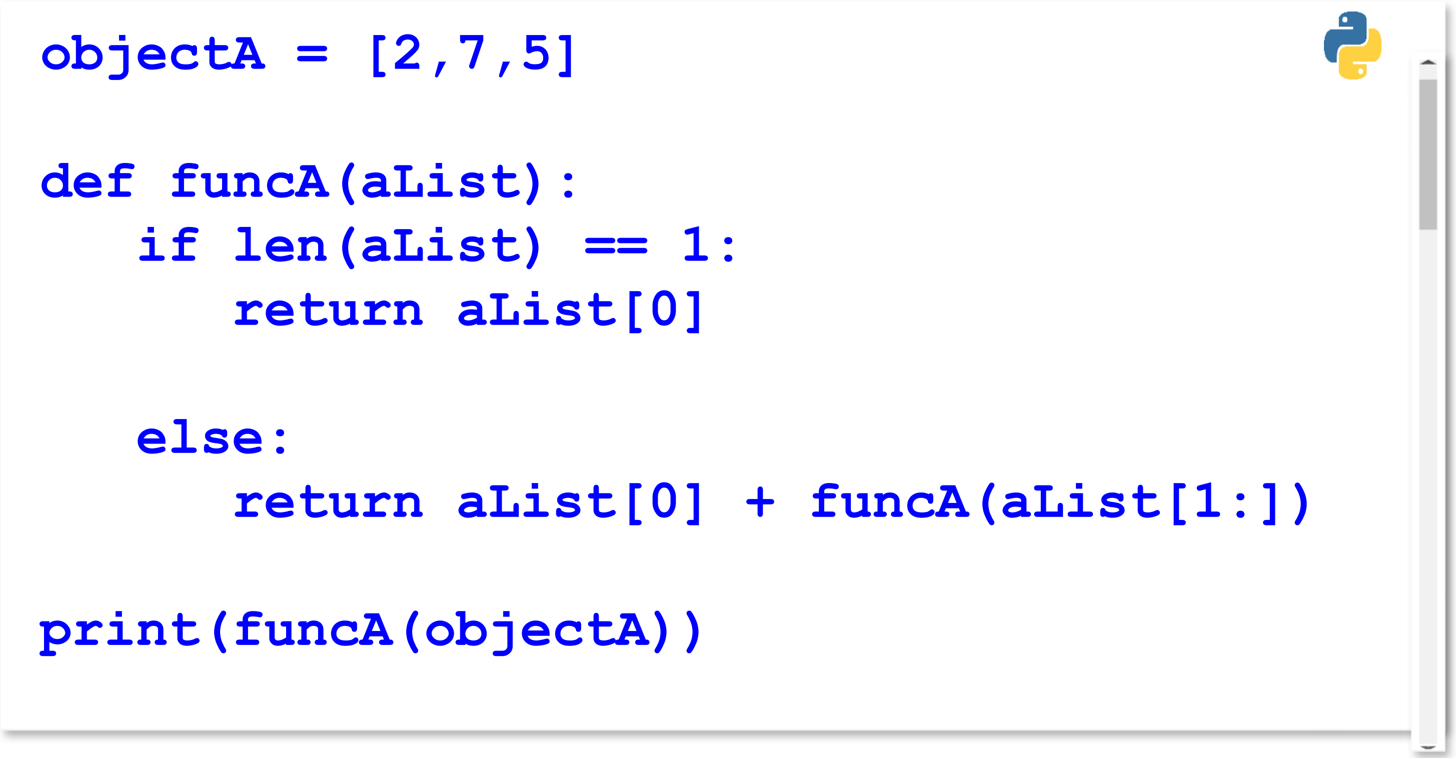
[✓ D. 3]

E. 9

F. 12

G. 14

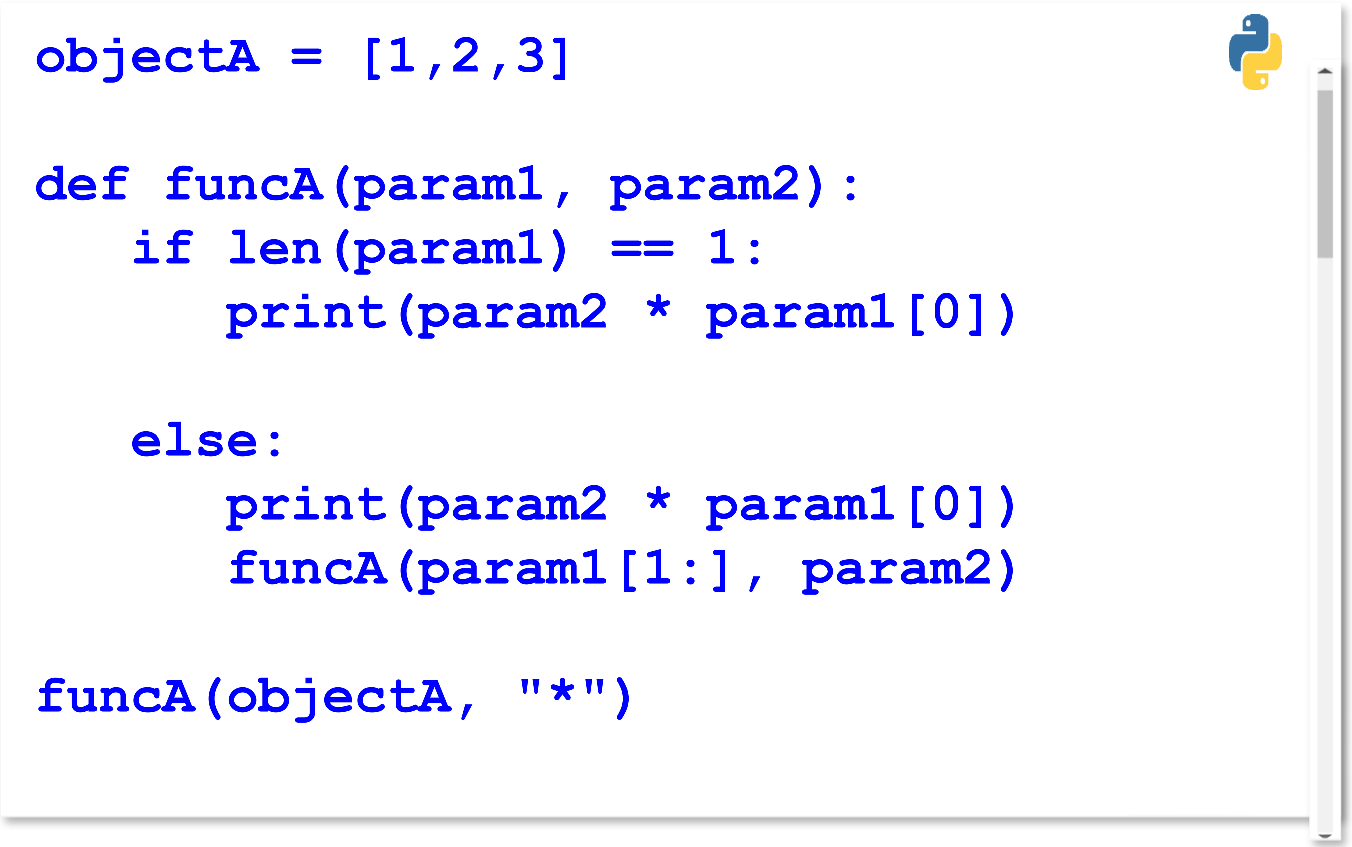
Q3: What is the output of the following Python program?



```python
objectA = [2,7,5]

def funcA(aList):
   if len(aList) == 1:
      return aList[0]

   else:
      return aList[0] + funcA(aList[1:])

print(funcA(objectA))
```

A. 2
B. 7
C. 5
D. 3
E. 9
F. 12
✓ G. 14

Q4: What is the output of the following Python program?



```
objectA = [1,2,3]

def funcA(param1, param2):
    if len(param1) == 1:
        print(param2 * param1[0])

    else:
        print(param2 * param1[0])
        funcA(param1[1:], param2)

funcA(objectA, "*")
```

```
A.*
  *
  *
```

```
B.**
  **
  **
```

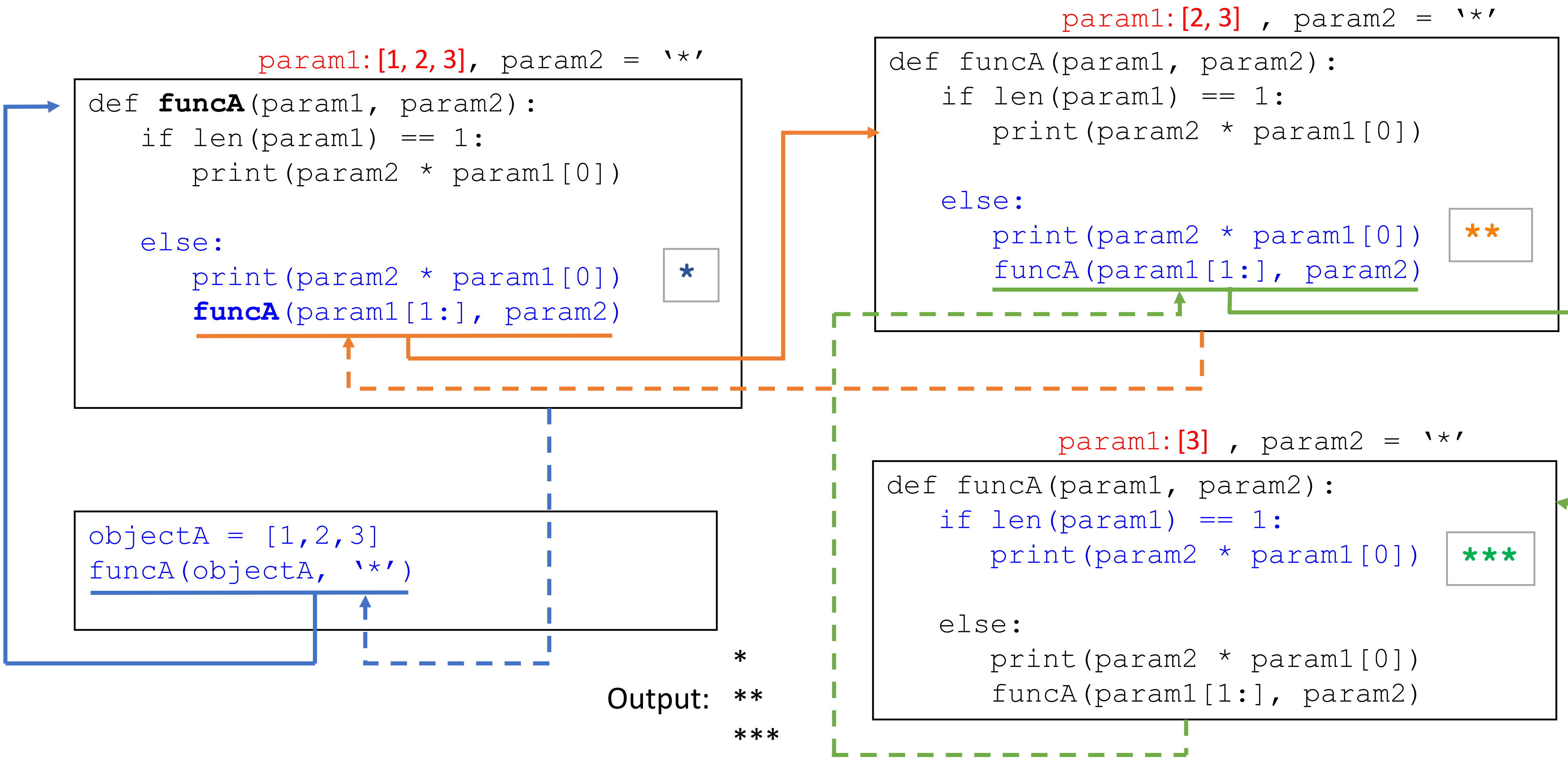
```
C.***
  ***
  ***
```

```
✓ D. *
     **
     ***
```

```
E.***
  **
  *
```

```
F.*
  ***
  **
```

Q4 Call Graph Explained



param1: [1, 2, 3], param2 = ‘*’

```
def funcA(param1, param2):
    if len(param1) == 1:
        print(param2 * param1[0])

    else:
        print(param2 * param1[0])
        funcA(param1[1:], param2)
```

param1: [2, 3] , param2 = ‘*’

```
def funcA(param1, param2):
    if len(param1) == 1:
        print(param2 * param1[0])

    else:
        print(param2 * param1[0])
        funcA(param1[1:], param2)
```

param1: [3] , param2 = ‘*’

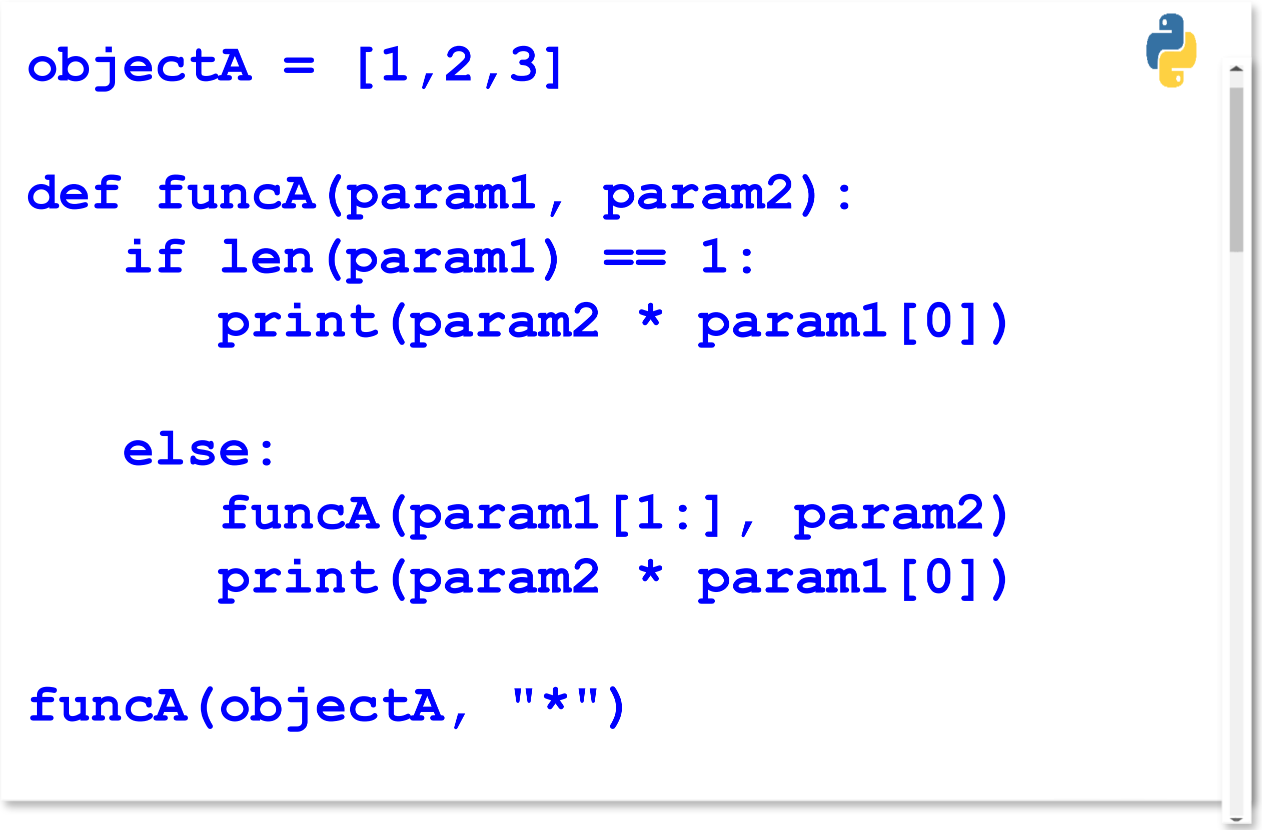
```
def funcA(param1, param2):
    if len(param1) == 1:
        print(param2 * param1[0])

    else:
        print(param2 * param1[0])
        funcA(param1[1:], param2)
```

```
objectA = [1,2,3]
funcA(objectA, ‘*’)
```

Output:

```
*
**
***
```

Q5: What is the output of the following Python program?



```
objectA = [1,2,3]

def funcA(param1, param2):
    if len(param1) == 1:
        print(param2 * param1[0])

    else:
        funcA(param1[1:], param2)
        print(param2 * param1[0])

funcA(objectA, "*")
```

```
A.*
  *
  *
```

```
B.**
  **
  **
```

```
C.***
  ***
  ***
```

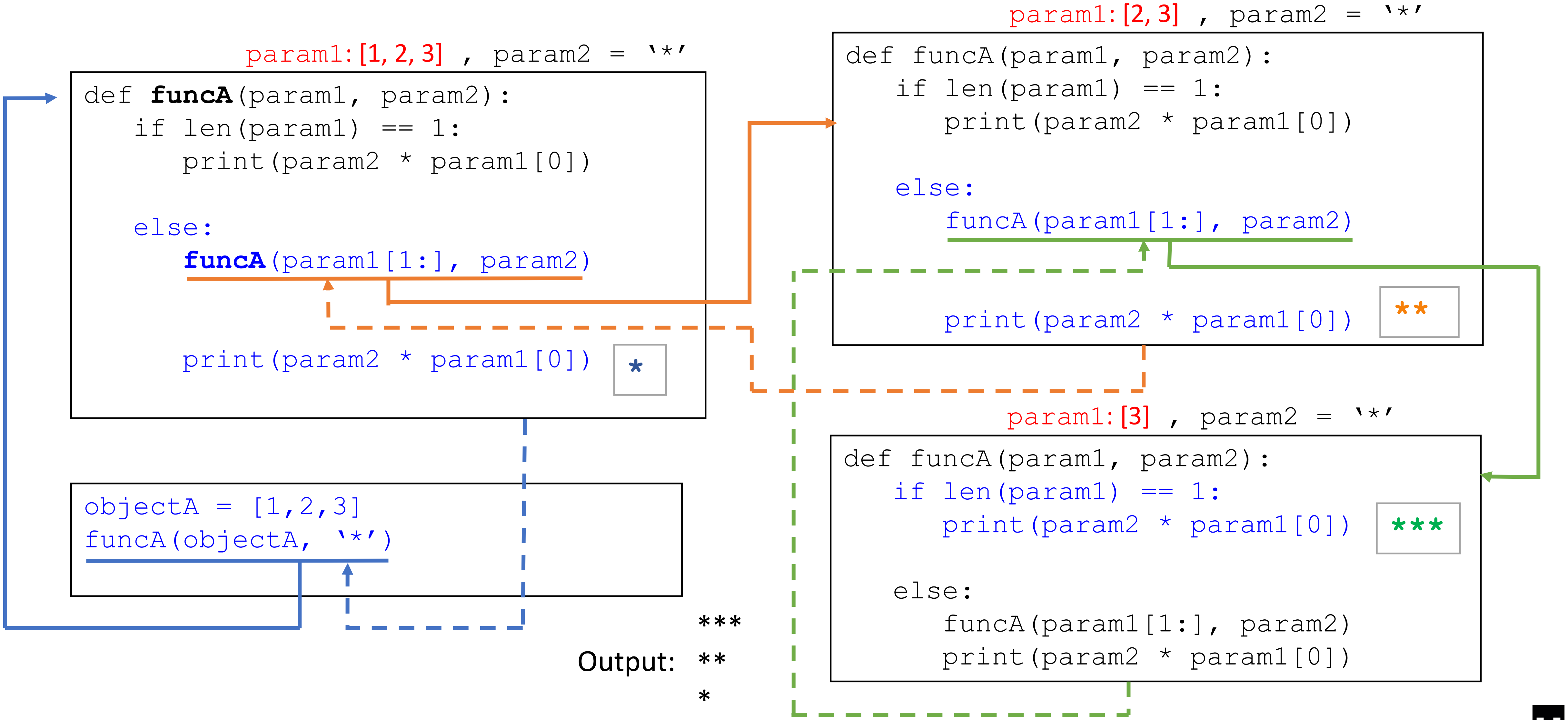
```
D. *
   **
   ***
```

✓

```
E.***
  **
  *
```

```
F.*
  ***
  **
```

Q5 Call Graph Explained



param1: [1, 2, 3] , param2 = ‘*’

```
def funcA(param1, param2):
    if len(param1) == 1:
        print(param2 * param1[0])

    else:
        funcA(param1[1:], param2)


        print(param2 * param1[0])
```

*

param1: [2, 3] , param2 = ‘*’

```
def funcA(param1, param2):
    if len(param1) == 1:
        print(param2 * param1[0])

    else:
        funcA(param1[1:], param2)


        print(param2 * param1[0])
```

**

param1: [3] , param2 = ‘*’

```
def funcA(param1, param2):
    if len(param1) == 1:
        print(param2 * param1[0])

    else:
        funcA(param1[1:], param2)
        print(param2 * param1[0])
```

***

```
objectA = [1,2,3]
funcA(objectA, ‘*’)
```

Output:

```
***
**
*
```