CS 120: Intro to Computer Programming II

In-Class Activity - 02 References - Day 3

Activity 1 - Turn in this one

Predict, **without running it,** what this code will execute. A reference diagram will probably help you!

```
y = [100]
x = y
z = x[0]
a = y[0]
y[0] = 444
print(x)
print(y)
print(z)
print(a)
print()
y = -1
print(x)
print(y)
print(z)
print(a)
```

Now, execute the code. Were your predictions correct?

Solution: The central insight, from the first part of the code, is that `x,y` refer to the same array. Therefore, changing `y[0]` necessarily affects `x` in the same way.

On the other hand, `z,a` were both set by reading a reference **out** of the array. Thus, `z,a` are unaffected when we alter that element of the array.

Thus, the first four lines of output are:

```
[444]
[444]
100
100
```

In the second section, we first change `y` to a single integer, althugh `x` still points at the array. Thus, the second set of printouts is:

```
[444]
-1
100
100
```

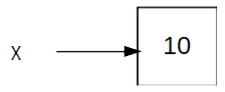
Activity 2 - Turn in this one

Execute the following code. Then, draw the reference diagram for `x`, after the function has returned.

While you are only required to turn in the reference diagram for the final state, make sure to discuss this in detail with your group!

```
def myfun(v):
    return [ v[0] ]

x = myfun( [10,20] )
```

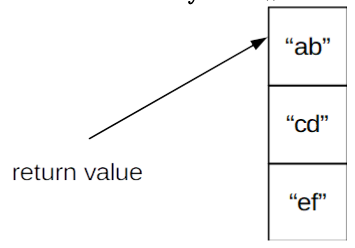
Solution:



(activity continues on the next page)

Activity 3 - Turn in this one

Write a function `myfun1()` which takes no parameters, and returns the following array:

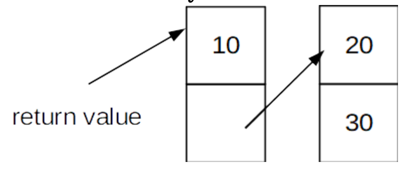


Solution:

```
def myfun1():
    return [ "ab", "cd", "ef" ]
```

Activity 4 - Turn in this one

Write a function `myfun2()` which takes no parameters, and returns the following array:



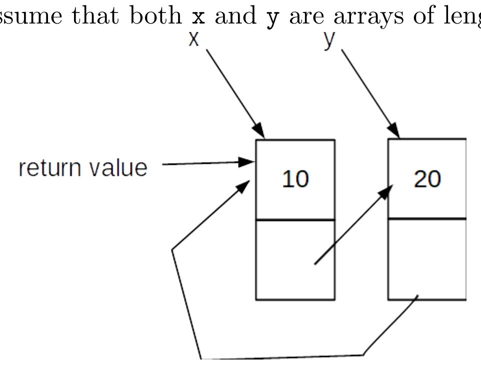
Solution:

```
def myfun2():
    return [ 10, [20,30] ]
```

Activity 5 - Optional

OPTIONAL. Complete this if you have time, and turn it in. If you don't have time, you may report to your TA that you ran out of time.

Write a function `myfun3(x,y)` that will return a value whose reference diagram is shown below. You can assume that both `x` and `y` are arrays of length 2.



Solution:

```
def myfun3(x,y):
    x[1] = y
    y[1] = x
    return x
```

(activity continues on the next page)

Activity 6 - Optional

OPTIONAL. Complete this if you have time, and turn it in. If you don't have time, you may report to your TA that you ran out of time.

Expain, in your own words, the difference between the `==` operator and the `is` operator.

Solution: `==` compares the **contents** of two objects; if they represent the same values, then this will return `True` .

`is` checks to see if the two expressions **are the same object.** Even if they have the same contents, if they are stored in different objects, `is` will return `False` .

Activity 7 - Optional

OPTIONAL. Complete this if you have time, and turn it in. If you don't have time, you may report to your TA that you ran out of time.

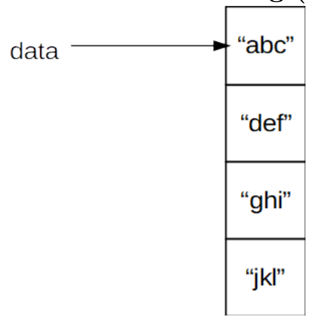
I've told you that parameters passed to a function - and the value returned from it - are basically like assignment statements. Do you believe me that this is true???

Execute the following code. At each point where I ask you (including inside the function!) draw the complete data structure diagram. Then, predict what will be printed out by the code.

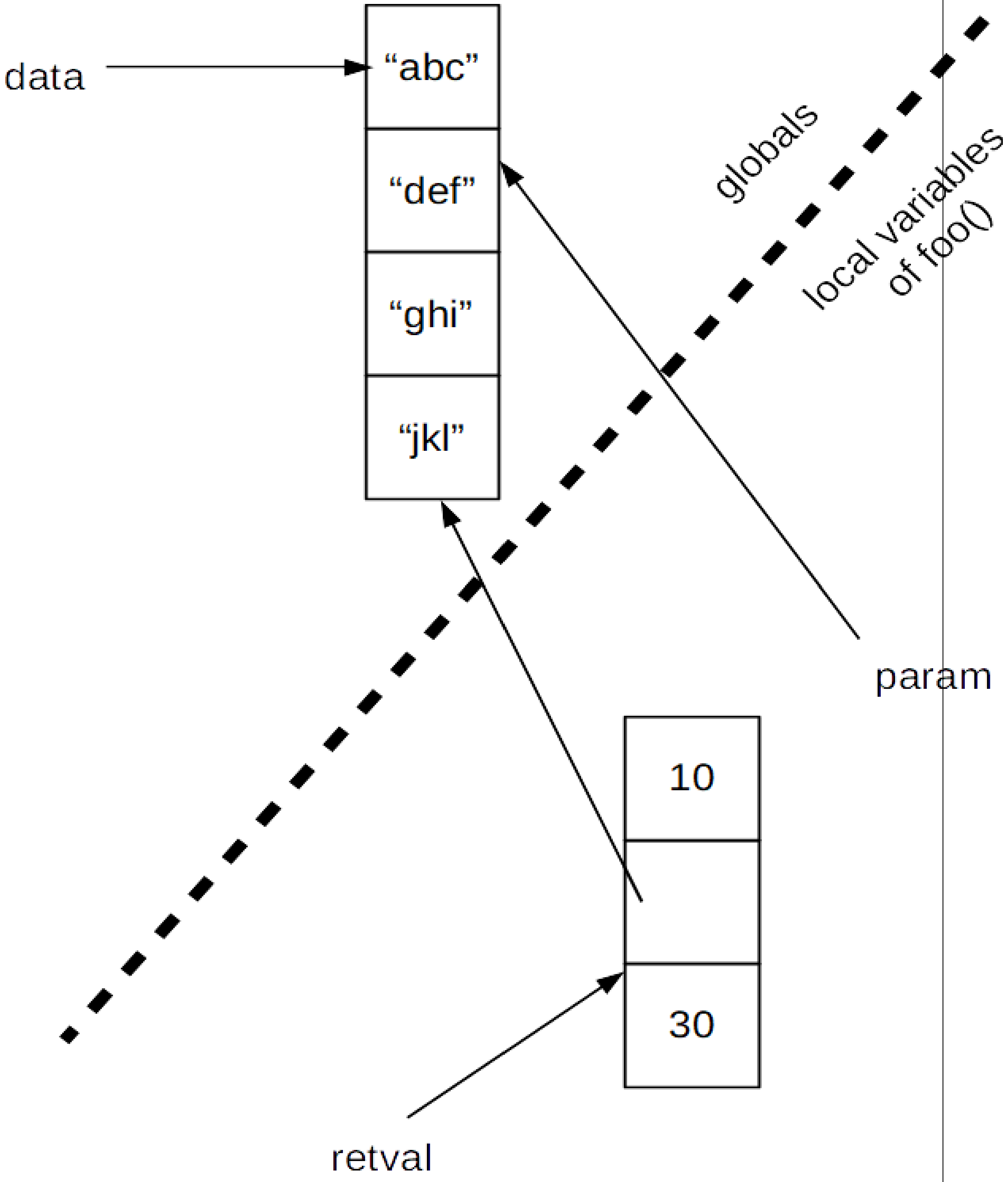
Finally, have someone in the group execute the code to double-check.

```
def modifier(param):
    retval = [10, param, 30]
    # DRAW HERE

    return retval


data = ["abc", "def", "ghi", "jkl"]
# DRAW HERE

changed = modifier(data)
# DRAW HERE

print(changed)
```

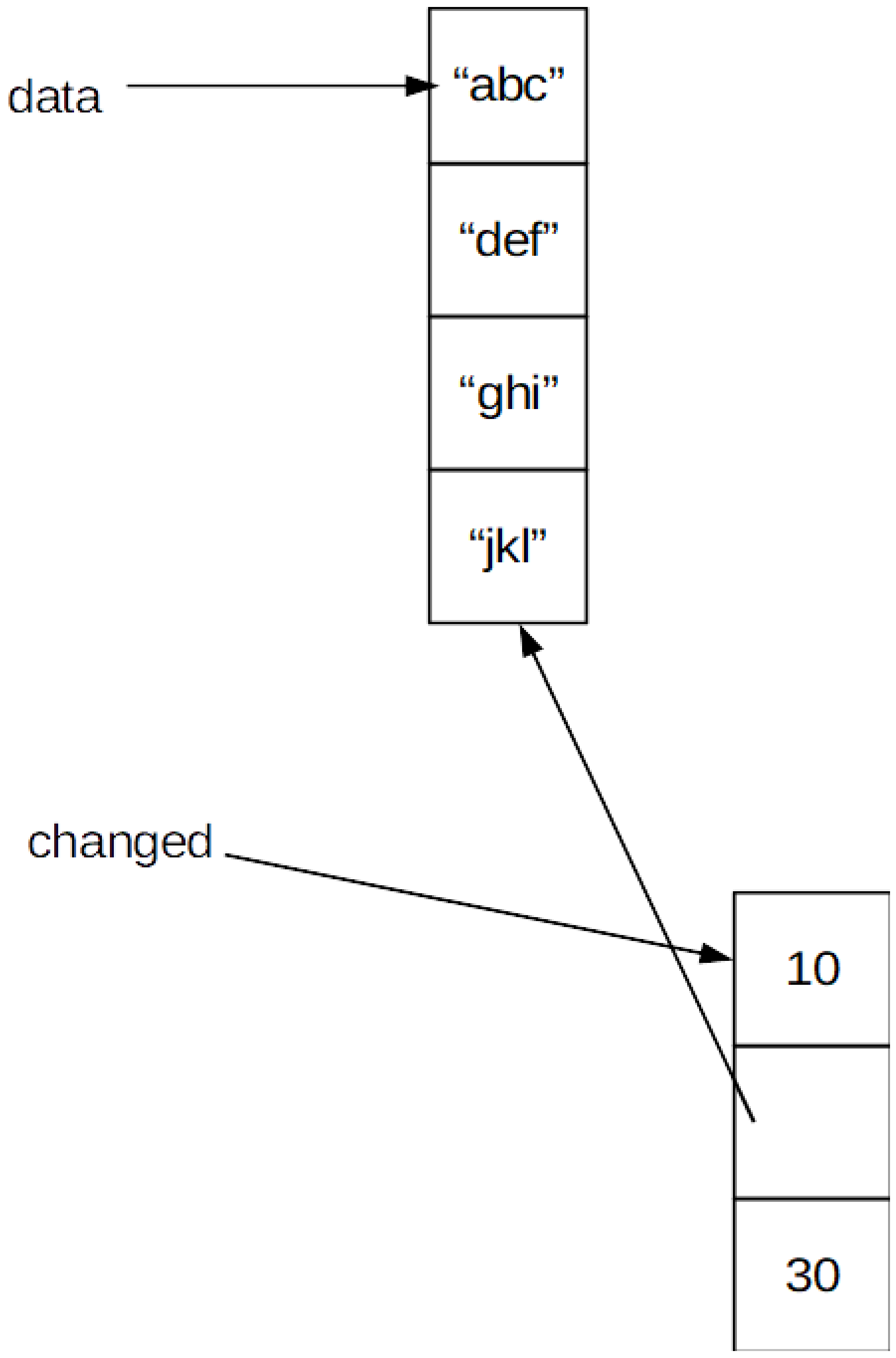
Solution: First drawing (before the function call!)



Second drawing (inside the function call)



Third drawing (after the function call)



Therefore, Python will print

```
[ 10, ["abc", "def", "ghi", "jkl"], 20]
```