CS 120 (Spring 22): Introduction to Computer Programming II

Short Project #3 - Intro to References

due at 5pm, Thu 3 Feb 2022

REMEMBER: The `itertools` and `copy` libraries in Python are **banned.**

1 Overview

In this project, you will be practicing with reading and creating reference diagrams. First, we will give you a series of reference diagrams, and you will produce a function to build each one; some of the functions will have parameters, while others will not.

Second, we will give you a few snippets of code, and **you** will draw the reference diagram for each; in some cases, you will simply produce a reference diagram for the final state, but in others you will produce multiple diagrams, to represent various steps in the process.

2 Background - A File of Functions???

We all have seen that we can write a program inside a Python file; we define the `main()` function, and then we call it ourselves.

But another important skill for a programmer is the ability to write a function which will be utilized in somebody else's program. In this case, you define a file, and define functions inside of it, but you **do not** call the function yourself. Instead, a `main()` function, in some other file, calls your function when it wants to. (Of course, this means that the other file has to import your file.)

For the first part of this Short Project, you will be defining several functions in one file. Each of my testcases will import your file, and then call one or more of the functions. It will then examine things about the data structure that you return, and maybe print some things out.

IMPORTANT:
When designing a function to be called by other code, always be clear about what the caller expects you to do. Are you supposed to print something out? Fine - call `print()` as necessary (but don't print more than you are asked to do!) But if you are only asked to **return** a value, then return it, and let the caller figure out what to do with the value that you give them.

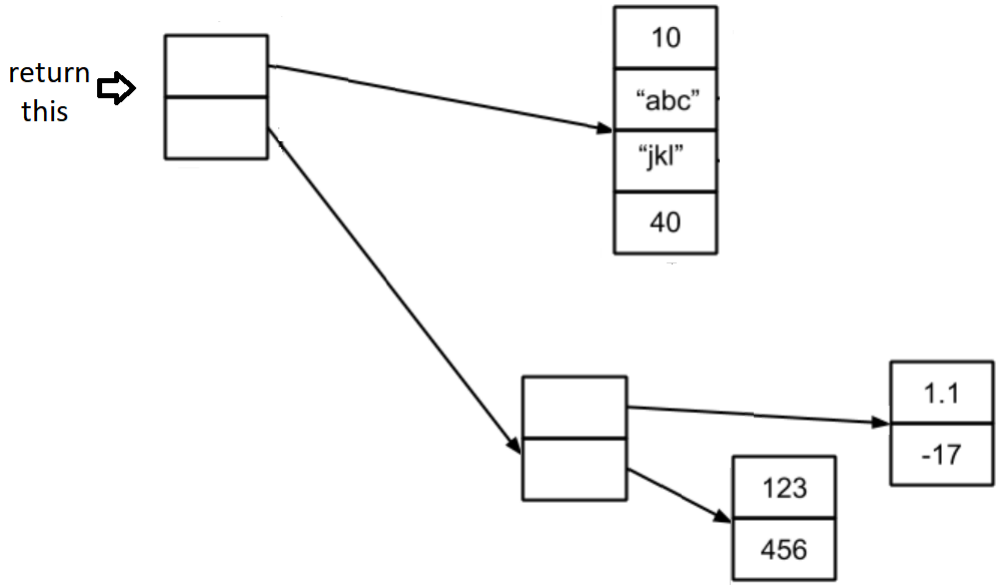
3 Part 1: Shape Functions

In `shapes.py` write a series of functions, each of which builds a data structure of some sort; we've drawn the picture for each data structure below. Some of

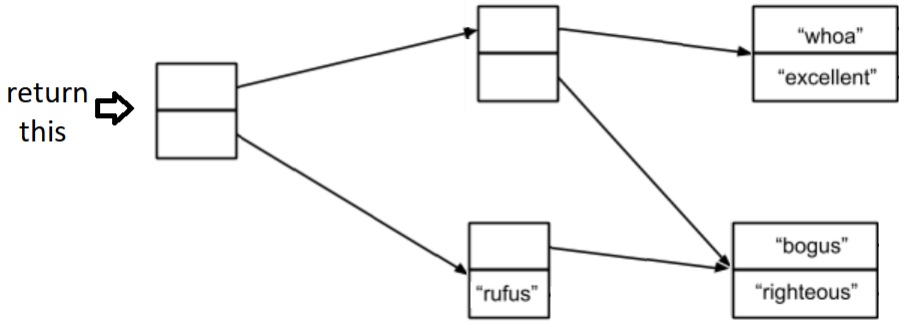
the functions take one or more parameters; in that case, drop the values into the data structure at the proper location. You should **not** do anything to these parameters (other than putting them in the proper locations); you know nothing about what they contain.

NOTE: All of the rectangles in these pictures represent Python lists (that is, arrays). Do not attempt to use tuples, classes, or any other type.

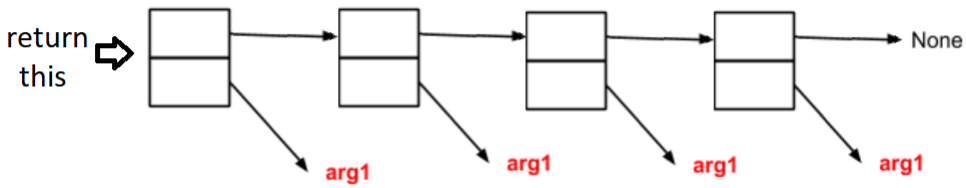
`shape_alpha()`



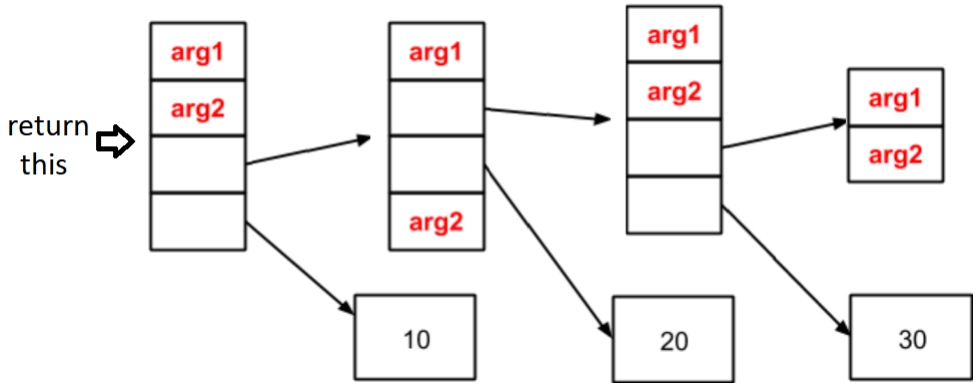
`shape_bravo()`



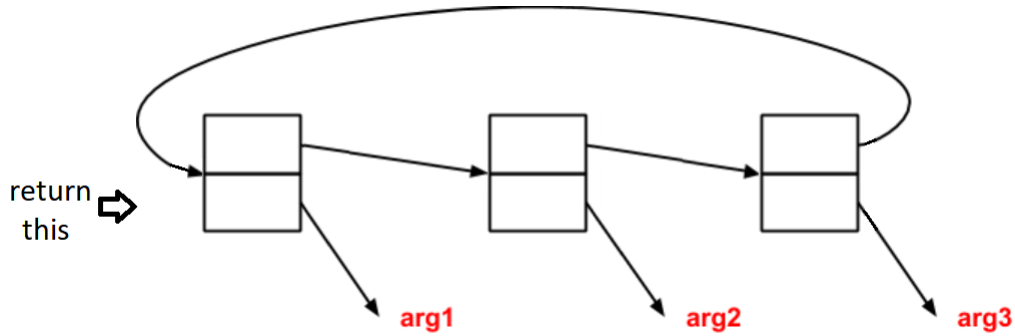
`shape_charlie(arg1)`



`shape_delta(arg1, arg2)`



`shape_echo(arg1, arg2, arg3)`



(spec continues on next page)

4 Part 2: Draw Your Own Reference Diagrams

For these diagrams, you may either submit a set of image files, or a single PDF that has all of them. Whatever you do, make sure to clearly label each one.

4.1 Diagram A

Draw a reference diagram for the **final state** of the variables, after this snippet of code runs.

```
data = [None, None, None, None, None, None]

for i in range(5):
    data[i] = [1*i, 11*i, 111*i]

data[-1] = data
```

4.2 Diagram B

Draw a reference diagram for the final state of the variables, after this snippet of code runs.

```
data    = [10, None, 20, ["foo", "bar", "baz"], None ]
data[1] = ["qwerty", "uiop", data[1] ]
data[4] = data[1]
```

4.3 Diagram(s) C

Draw multiple reference diagrams for this snippet of code; draw one for each step where you see the `DRAW HERE` comment. (Inside the loop, draw it once for each iteration of the loop.)

For this problem, **also write out what this code will print.**

```
all  = [10, None]
tail = all
# DRAW HERE

for i in range(4):
    tail[1] = [ i*10+20, None ]
    tail    = tail[1]
    # DRAW HERE

print(all)
```

(spec continues on next page)

5 Turning in Your Solution

You must turn in your code using GradeScope.

6 Acknowledgements

Thanks to Saumya Debray for many resources that I used and adapted for this class.