Exercise: Classes and object exploration

In this reading, you will explore the behaviour of functions, objects and classes in Python and how the flow of execution of different program statements works to enable better understanding.
You will perform minor modifications in the given code to observe how it changes the output.

First, set up a file called **`class_explore.py`** that contains the following piece of code. Alternatively, you can use the interactive environment here.

```python
class A:
    def __init__(self, c):
        print("---------Inside class A----------")
        self.c = c
    print("Print inside A.")

    def alpha(self):
        c = self.c + 1
        return c

print(dir(A))
print("Instantiating A..")
# a = A(1)
# print(a.alpha())

class B:
    def __init__(self, a):
        print("---------Inside class B----------")
        self.a = a

#    print(a.alpha())
    d = 5
    print(d)
#    print(a)

print("Instantiating B..")
# b = B(a)
# print(a)
```

Run

Reset

```
Hello World!
Print inside A.
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__',
Instantiating A..
5
Instantiating B..
```

Now, modify the code as per the instructions below and observe the changes.

Step 1: Run the code and observe its output. Take note of every line in the output and how it is different from the output you expected.
Algorithmically we can view the program consisting of the following:

`1. Class definition of A`

`1.1 Constructor for A`

`1.2 Definition of local method alpha()`

`2. Instantiating object a over class A`

`3. Calling method alpha() over object of class A`

`4. Class definition of B`

`5. Constructor for B`

`5.1 Calling method alpha() over object of class A`

`5.2 Instantiating object a over class B *.`

`Additional print statements distributed through the code.`

Step 2: Comment out lines #13, 14, 21, 24, 27 and 28. Run the code again.

The output is:

```
Hello World!
Print inside A.
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '
Instantiating A..
5
Instantiating B..
```

Even if you have commented out the creation of instances for both classes A and B, the output still shows “Print inside A” and “Print inside B” and also the value of variable d, which is 5. How is that possible?
It's because statements inside a class body get executed irrespective of the instance creation. You will also see how the print statement “Inside class A”, which is inside the constructor, is not executed because it's inside a function.
The value of d=5 being printed demonstrates that the namespace and scope of the variable is determined by the interpreter before you create any instance of the class or call any function inside it. If you observe the list you get by calling the dir() function, you'll note that the last entry is the alpha() function added to the namespace of A.

Step 3: Now remove the comment for lines 21 and 24.

If you run the code at this point, it will throw an error, "NameError: name 'a' is not defined". Take note of how you have passed the object a to the constructor of class B and the code still worked fine earlier. Only when you tried to ‘use’ object a, did you get an error because it has not been instantiated. In other words, Python still does not know what ‘a’ means. The same will happen if you remove commenting next to line 28.

To make the code work, now remove the # in front of line 14 and run it again.

The output is:

```
Hello World!
Print inside A.
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '
Instantiating A..
---------Inside class A----------
2
5
<submission.A object at 0x7fcab3ef6940>
Instantiating B..
```

Step 4: Remove the commenting for line 27 and 28.

The variable c of class A is modified over object a inside class B. Even though the instance of class B is still not created, the value of a.c is still getting updated, even outside the class, as evidenced by the final line in the output which shows the output is 2.

Step 5: Finally remove all the remaining comments and run the code one more time.

Here are a few observations.

When you try and print the ‘object’ of class A as in lines 21 and 28, you get the address of the object instead of the contents.
Note how the address of object a is the same both inside class B and in the global scope of the program. It remains the same irrespective of from where it is called.
The alpha() function is called twice in the program, but you still get the output as 2 every time and not 3. That's because the value is updated only temporarily and not assigned to anything.

Revise items about classes, function calls and scope in case of confusion.