Announcements

- Homework
  - Homework 9 is out!
  - Homework 8 is due on Friday as well!
- Virtual CS Tea tomorrow!
  - Connection info posted on Campuswire
  - Just socializing and catching up this week
- CS/Data Science professor candidate talks upcoming
  - Teaching talk today 3-4
  - Research talk tomorrow 4:10-5:30
  - Zoom info on Campuswire
- Polling: `rembold-class.ddns.net`

Review Question

What would be the output of `print(D.x)` for the very contrived bit of code to the right?

A) 15

B) 13

C) 10

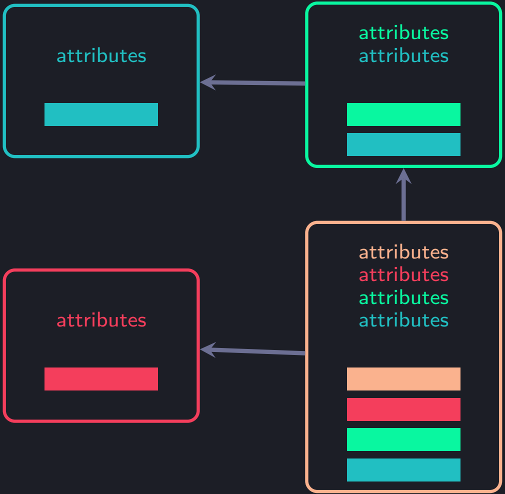
D) None

```
class ObjA:
    def __init__(self):
        self.x = 5

class ObjB(ObjA):
    def __init__(self):
        ObjA.__init__(self)
        self.x = 8

class ObjC(ObjB):
    def __init__(self):
        ObjA.__init__(self)
        self.y = self.x + 2

class ObjD(ObjC):
    def __init__(self):
        ObjC.__init__(self)
        ObjB.__init__(self)
        self.x += self.y

D = ObjD()
```

Multiple Inheritance

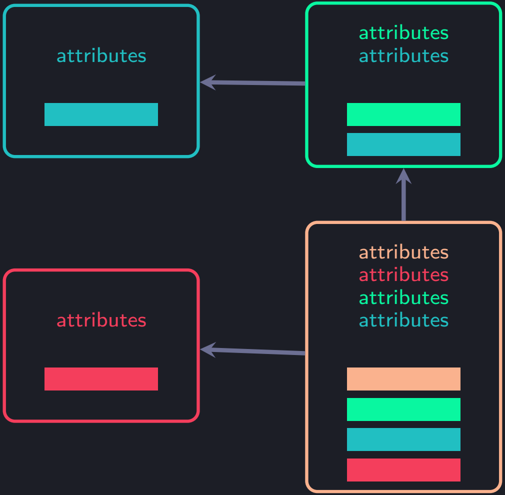
- A class is not limited to inheriting from only a single parent!
- Can provide multiple parents!
    - Ordering matters!
- Example: a class that forms the intersection of two other types

Pieces of Inheritance



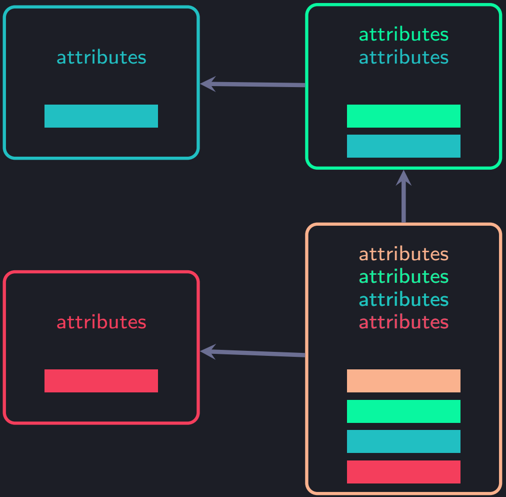
```
class Orange(Red, Green):
    def __init__(self):
        Green.__init__.self()
        Red.__init__.self()
        self.attributes
```

Pieces of Inheritance



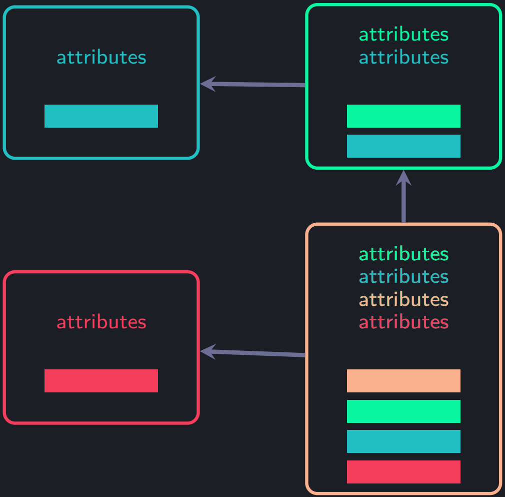
```
class Orange(Green, Red):
    def __init__(self):
        Green.__init__.self()
        Red.__init__.self()
        self.attributes
```

Pieces of Inheritance



```
class Orange(Green, Red):
    def __init__(self):
        Red.__init__.self()
        Green.__init__.self()
        self.attributes
```

Pieces of Inheritance

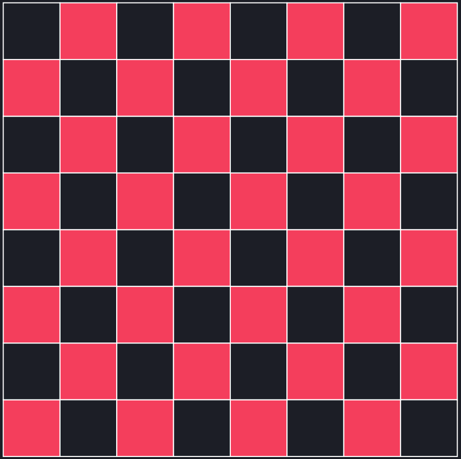


```
class Orange(Green, Red):
    def __init__(self):
        Red.__init__.self()
        self.attributes
        Green.__init__.self()
```

Thought Experiment

Suppose you wanted to program the background logic for playing a game of checkers. What different classes might you want to implement? What about any sub/super classes?

List them and their relationships (if any) out. We'll talk about them in a moment.



Variable Types

- So far we have been looking at instance variables.
  - A data attribute that is associated to a particular instance or object
- Much of the time, this is what we want when dealing with OOP ideas
- Sometimes though it is useful to create an attribute that is tied directly to the class itself, and not to a specific object.
  - Called a class variable.

Creating a class variable

- No usage of `self`, this variable applies to all!
- Define just like you would a normal variable in a function
- Usually defined at the top before the methods

```python
class MyBestClass:
    a_class_variable = 'hello!'

    def __init__(self):
        self.a_instance_variable = 'Hi!'
```

Accessing a class variable

- Can access directly by referencing the class:
  - `print(MyBestClass.a_class_variable)`
  - Requires no instance to actually exist
- Can access from an instance just like any other instance attribute

```
A = MyBestClass()
print(A.a_class_variable)
```

- If changing or setting a class variable, must use the `ClassName.Class_Variable` notation!
  - Using `self` or the object will create an instance variable instead!

Accessing a class variable

- Can access directly by referencing the class:
  - `print(MyBestClass.a_class_variable)`
  - Requires no instance to actually exist
- Can access from an instance just like any other instance attribute

```
A = MyBestClass()
print(A.a_class_variable)
```

- If changing or setting a class variable, must use the `ClassName.Class_Variable` notation!
  - Using `self` or the object will create an instance variable instead!

```
MyBestClass.class_var = 4
```

~~`A.class_var = 4`~~

When to Use

- Do all members or objects of your class share a common trait?
    - Think about if it would make sense for one member of that class to have that trait altered.
    - If no, might be a good candidate for a class variable
- Are there specific constants or known values that all instances of your class will rely on?
- Do you need some sort of global counter to keep track of the amount of instances you have created?