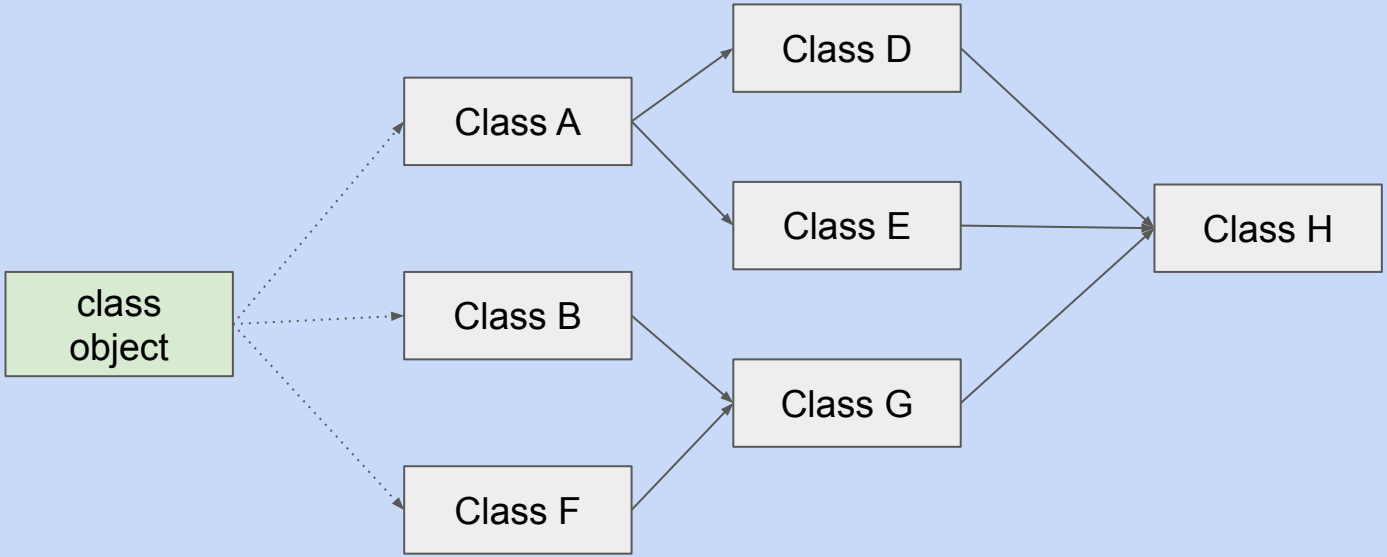
Python training - lab 13

OOP - multiple inheritance



OOP - multiple inheritance

```
class A:

    def generic(self):
        print("AAAA")


class B:

    def generic(self):
        print("BBBB")


class C(A, B):

    def generic2(self):
        B.generic(self)


c = C()
c.generic()
c.generic2()
print(C.mro())  # Method Resolution Order(MRO)
```

OOP - composition

```
class A:
    pass


class B:
    pass


class C:
    def __init__(self, atr1, atr2):
        self.atr1 = atr1
        self.atr2 = atr2

a = A()
b = B()

c = C(a, b)
```

OOP - __repr__ and __str__

```
class A:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __repr__(self)
        return f'A({self.x}, {self.y})'

    def __str__(self)
        return f'This is object A({self.x}, {self.y})'


a = A(4, 5)
print(a)
```

OOP - destructor

```
class A:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __del__(self)
        """
        This is the destructor.
        Gets called when the objects is destroyed
        """
        pass


a = A(4, 7)
a = 1234
```

OOP - operator overloading

```
class A:
   def __init__(self, x):
      self.x = x

   def __lt__(self, other):
      return self.x < other.x


a1 = A(4)
a2 = A(7)

if a1 < a2:  # normally error here
   print('a1 lower than a2')
else:
   print('a2 lower than a1')
```