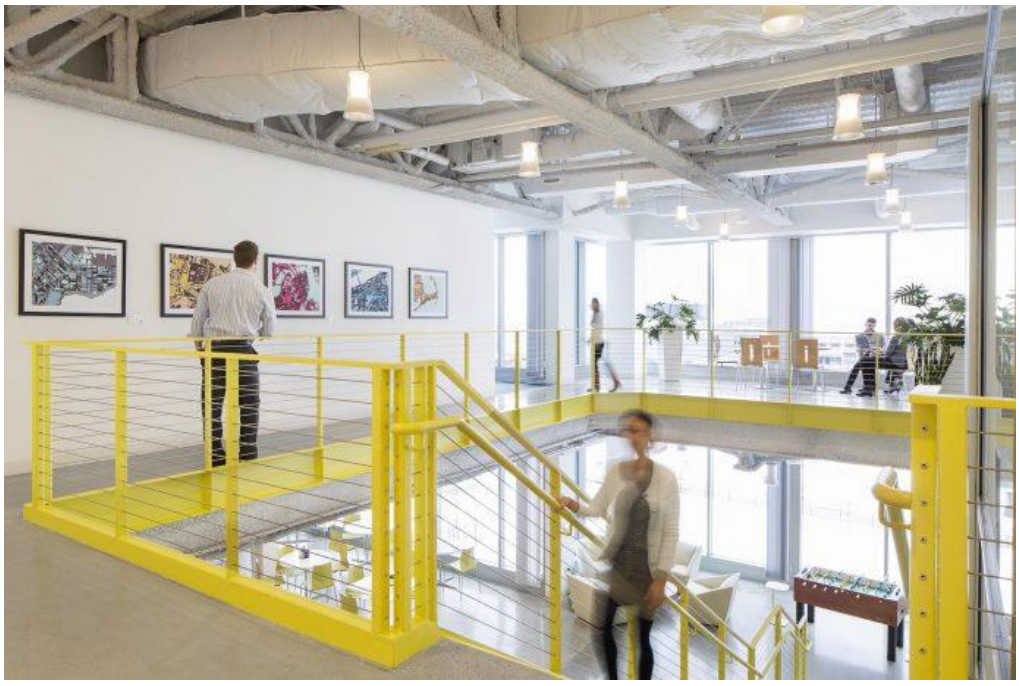
UPDATING STAFF RECORDS USING OBJECT ORIENTED PROGRAMMING

WITH PYTHON

DR. ALVIN ANG

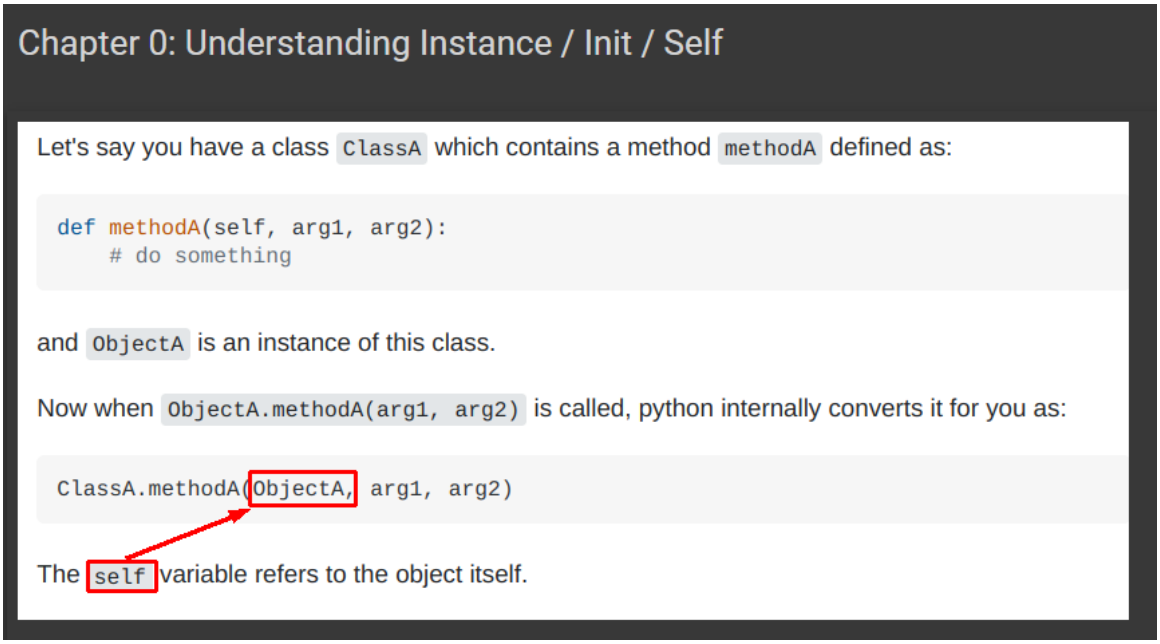


CONTENTS

I. OOP IS UNPOPULAR

- No, object-oriented programming (OOP) is not dead.
- But it is significantly less ubiquitous than it used to be.
- https://medium.com/machine-words/the-rise-and-fall-of-object-oriented-programming-d67078f970e2#:~:text=No%2C%20object%2Doriented%20programming%20

https://www.alvinang.sg/s/Updating_HR_Records_using_Object_Oriented_Programming_by_Dr_Alvin_Ang_Ch_0_Understanding_Instance_In.ipynb



https://stackoverflow.com/questions/2709821/what-is-the-self-parameter-in-class-methods

A. CREATING THE PARENT CLASS

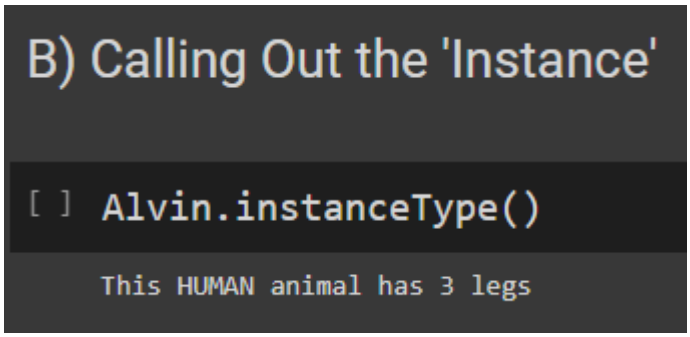
```
class Animal:

    color = 'white'
    legs = 4

#color and legs are Class Variables

    def __init__(self,color,legs):
        self.color = color
        self.legs = legs

# self.color and self.legs are Instance Variables

    def instanceType(self):
        print('This {} animal has {} legs'.format(self.color,self.legs))

#notice that the 'self.' points back to the 'Instance'

    def classType(self):
        print('This {} animal has {} legs'.format(Animal.color,Animal.legs))

#notice that the 'Animal.' points back to the 'Class'
```

B. AN OBJECT IS AN INSTANCE OF A CLASS

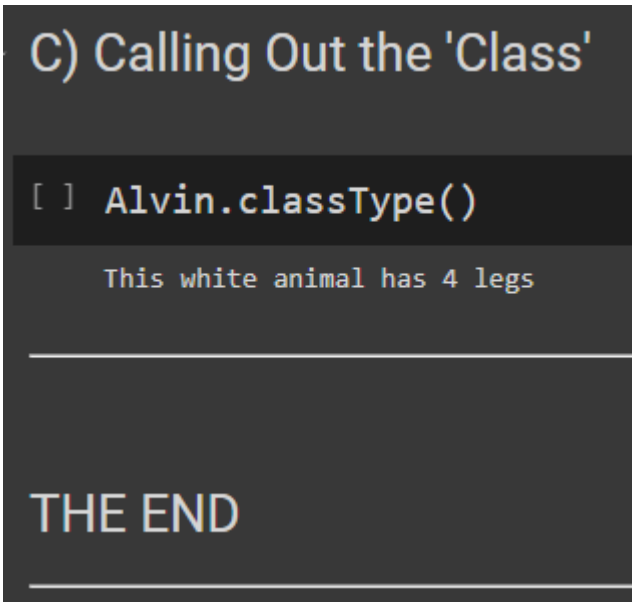
A) An Object is an Instance of a Class

```
[ ] Alvin = Animal('HUMAN', 3)

   #'Alvin' is now an Object = an Instance (child) of the Class 'Animal' (parent)
```

C. CALLING OUT THE 'INSTANCE'



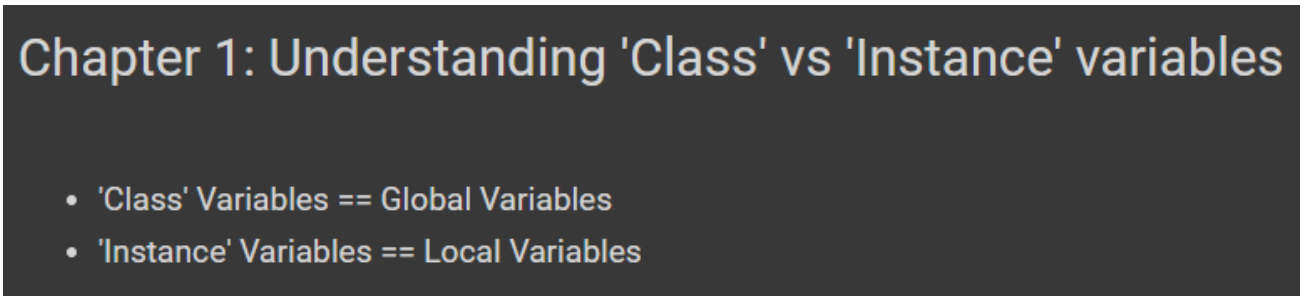
D. CALLING OUT THE 'CLASS'



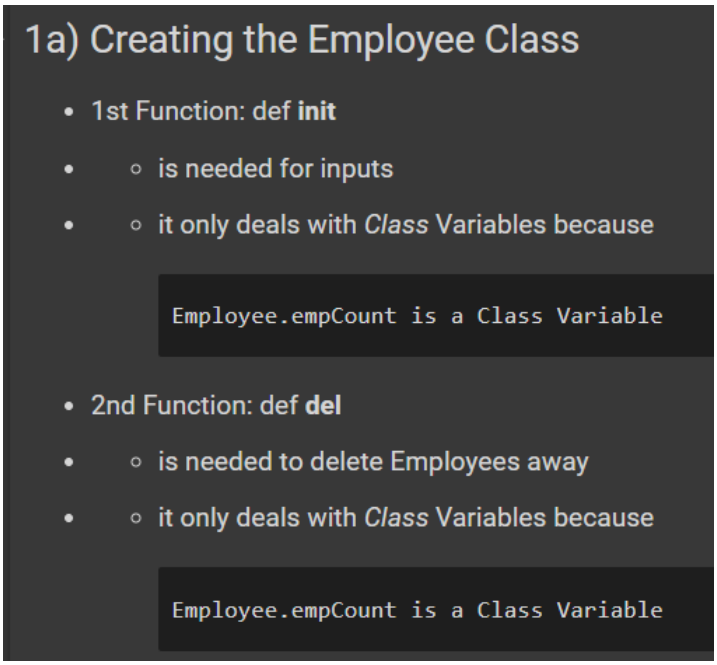
III. CHATPER 1: USING ONLY 1 PARENT CLASS (FULL TIME EMPLOYEES ONLY)

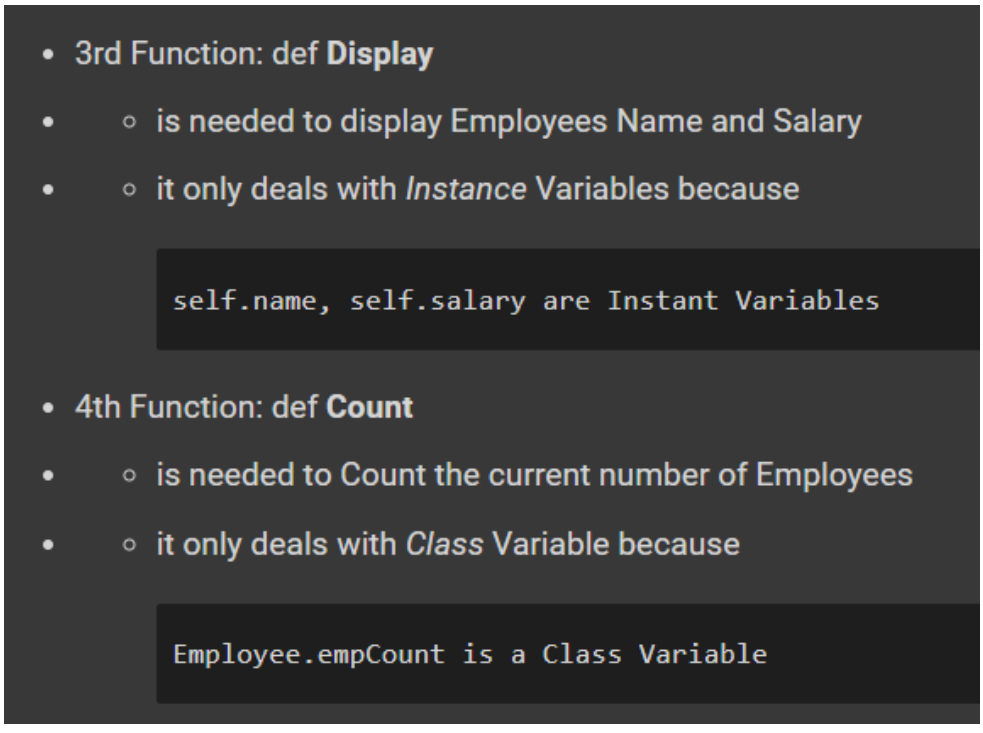
https://www.alvinang.sg/s/Updating_HR_Records_using_Object_Oriented_Programming_by_Dr_Alvin_Ang_Ch_1_Using_only_1_Parent_Class.ipynb

Understanding The Concept of 'Class' vs 'Instance' Variables



A. CREATING THE EMPLOYEE CLASS

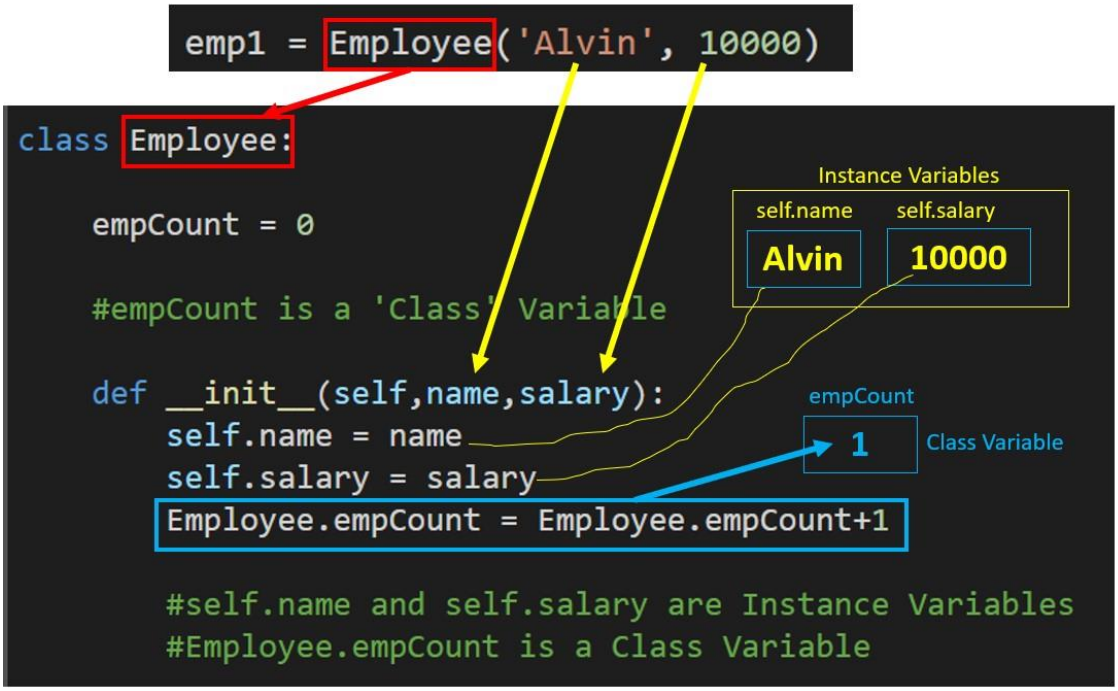




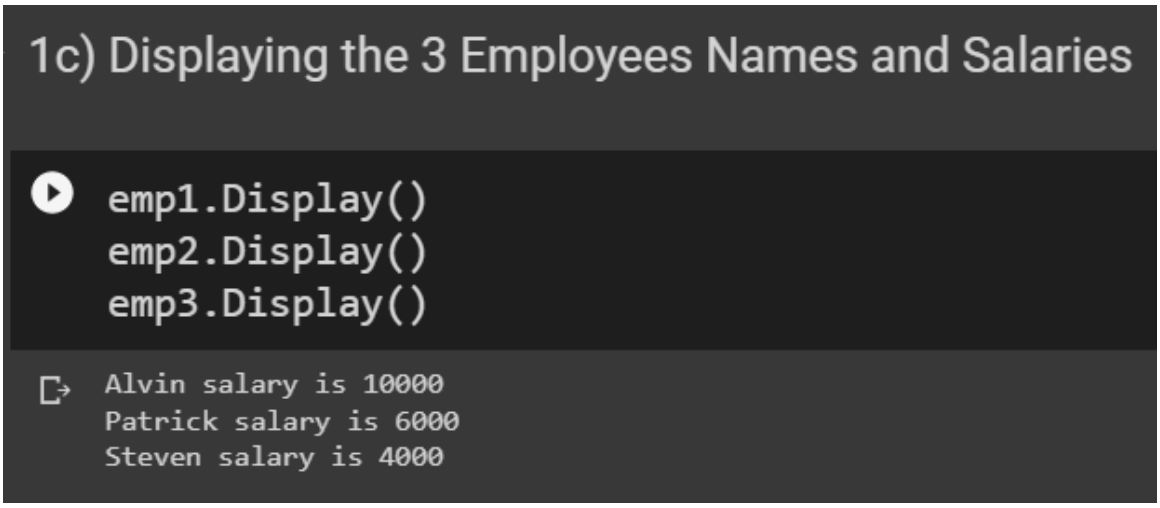
```
class Employee:

    empCount = 0                                        #Class Variable

#-------------------------------------------------------------------------------
#1st function: Init
#-------------------------------------------------------------------------------
    def __init__(self,name,salary):
        self.name = name                                #Instance Variable
        self.salary = salary                            #Instance Variable
        Employee.empCount = Employee.empCount+1         #Class Variable

#-------------------------------------------------------------------------------
#2nd function: Del
#-------------------------------------------------------------------------------
    def __del__(self):
        Employee.empCount = Employee.empCount-1

#-------------------------------------------------------------------------------
#3rd function: Display
#-------------------------------------------------------------------------------
    def Display(self):
        print("{} salary is {}".format(self.name,self.salary))

        #self.name and self.salary are Instance Variables

#-------------------------------------------------------------------------------
#4th function: Count
#-------------------------------------------------------------------------------
    def Count(self):
        print("Number of Employee is {}".format(Employee.empCount))

        #Employee.empCount is a Class Variable
```

B. INPUTTING 3 EMPLOYEES WITH THEIR SALARIES

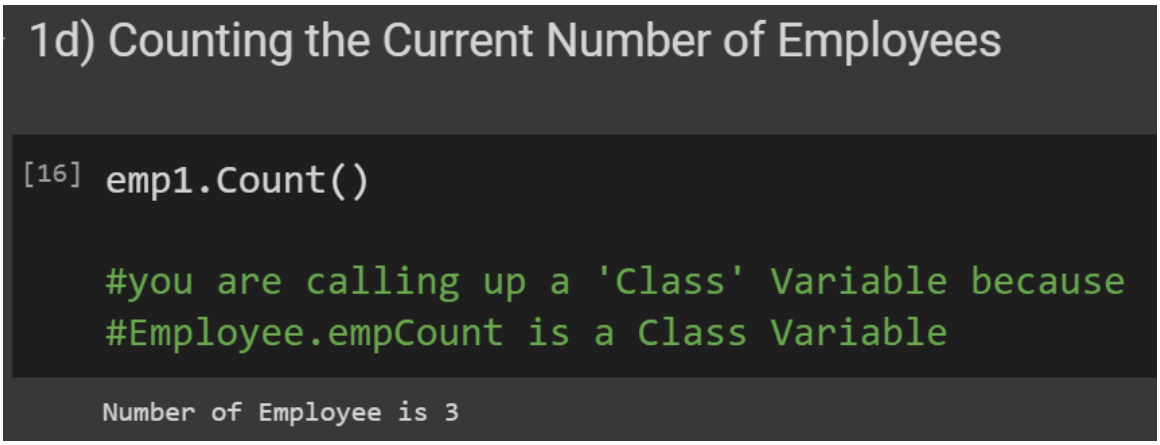
```
1b) Inputting 3 Employees with their Salaries

[14] emp1 = Employee('Alvin', 10000)
     emp2 = Employee('Patrick',6000)
     emp3 = Employee('Steven', 4000)
```



C. DISPLAYING THE 3 EMPLOYEES NAMES AND SALARIES



D. COUNTING THE CURRENT NUMBER OF EMPLOYEES



E. DELETING AWAY EMPLOYEE 1

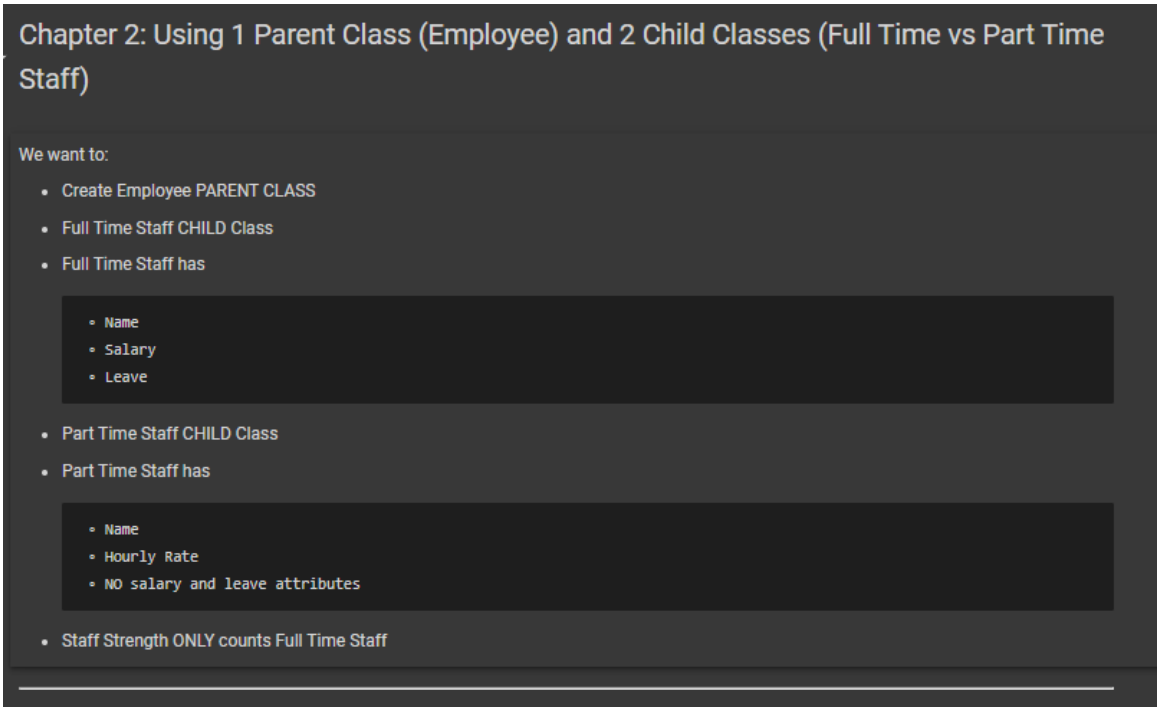
```
1e) Deleting away Employee 1

del emp1

#you are calling up a 'Class' Variable because
#Employee.empCount is a Class Variable
```

F. RECOUNTING THE CURRENT NUMBER OF EMPLOYEES

```
1f) Recounting the Current Number of Employees

[18] emp2.Count()

     Number of Employee is 2
```

IV. CHAPTER 2: USING 1 PARENT CLASS (EMPLOYEE) AND 2 CHILD CLASSES (FULL TIME VS PART TIME)

https://www.alvinang.sg/s/Updating_HR_Records_using_Object_Oriented_Programming_by_Dr_Alvin_Ang_Ch_2_Using_1_Parent_Class_Empl.ipynb



A. CREATING THE EMPLOYEE PARENT CLASS

2a) Creating the Employee PARENT Class

```
[1] class Employee:

        empCount = 0

    #-------------------------------------------------------------------------
    #1st function: Init
    #-------------------------------------------------------------------------

        def __init__(self,name,salary):
            self.name = name
            self.salary = salary
            Employee.empCount = Employee.empCount+1

    #-------------------------------------------------------------------------
    #2nd function: Del
    #-------------------------------------------------------------------------

        def __del__(self):
            Employee.empCount = Employee.empCount-1
            print("{} has left the company".format(self.name))

    #-------------------------------------------------------------------------
    #3rd function: Display Employee Count
    #-------------------------------------------------------------------------

        def dispEmployeeCount(self):
            print("Number of Employee is {}".format(Employee.empCount))
```

B. CREATING THE FT CHILD CLASS

2b) Creating the FT Child Class

```
class FullTimeStaff(Employee):

    #-------------------------------------------------------------------------------
    #1st function: Init
    #-------------------------------------------------------------------------------

        def __init__(self,name,salary,leave):
            super().__init__(name,salary)
            self.leave = leave

    #-------------------------------------------------------------------------------
    #2nd function: Display Full Time Employee Attributes
    #-------------------------------------------------------------------------------

        def dispFTEmployee(self):
            print("{} salary is {} and leave is {}".format(self.name,self.salary,self.leave))
```

- Note that the "Super Init" means we overwrite the Parent (Employee) class with values that will be input into the Child class here.

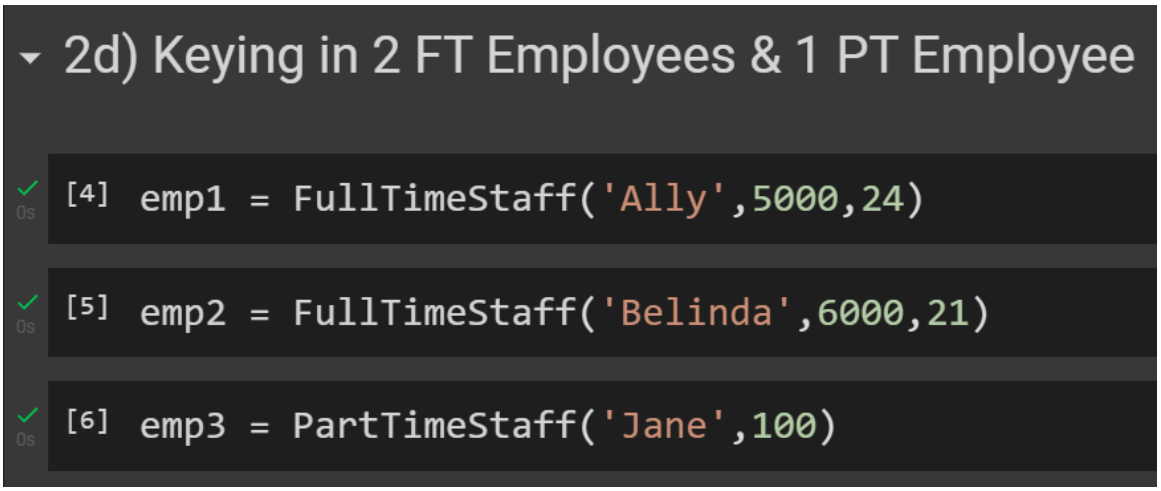
C. CREATING THE PT CHILD CLASS

2c) Creating the PT Child Class

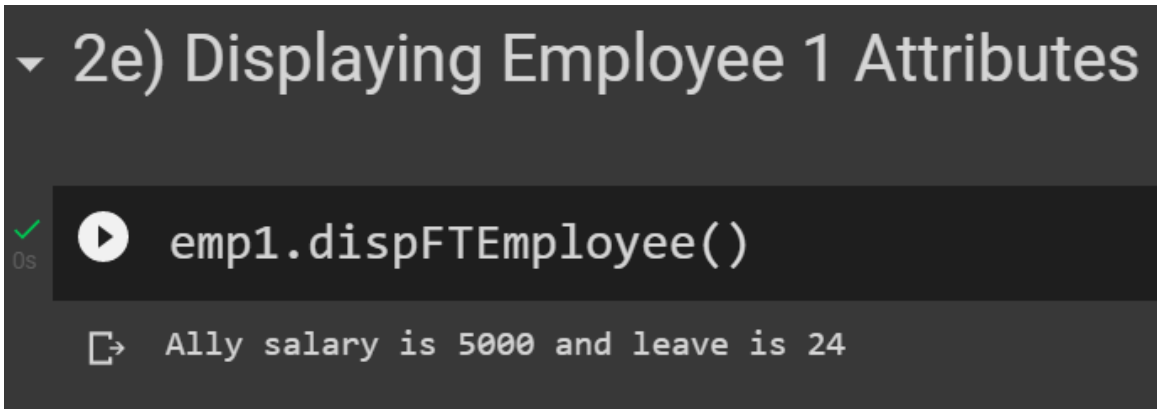
```
class PartTimeStaff(Employee):

    #-----------------------------------------------------------------------------
    #1st function: Init
    #-----------------------------------------------------------------------------

        def __init__(self,PTname,hrrate):
            #super().__init__(name,0)
            self.PTname = PTname
            self.hrrate = hrrate

    #-----------------------------------------------------------------------------
    #2nd function: Display Part Time Employee Attributies
    #-----------------------------------------------------------------------------

        def dispPTEmployee(self):
            print("{} hourly rate is {}".format(self.PTname,self.hrrate))
```

- Note that this PT Child class is almost entirely on its own.
- Even though it takes after the Parent Class, we delcare all local variables "PTname" and "hrrate" that has nothing to do with the Parent class.

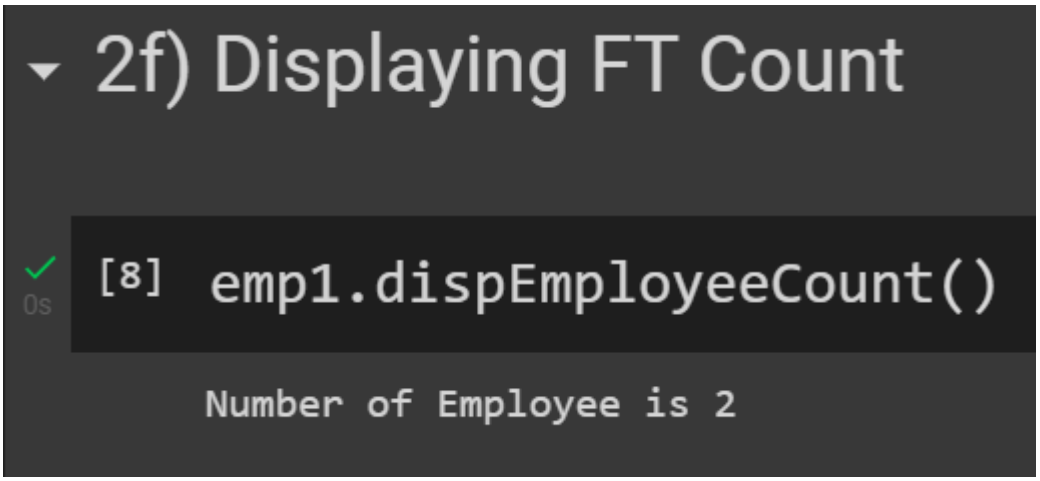
D. KEYING IN 2 FT EMPLOYEES AND 1 PT EMPLOYEE



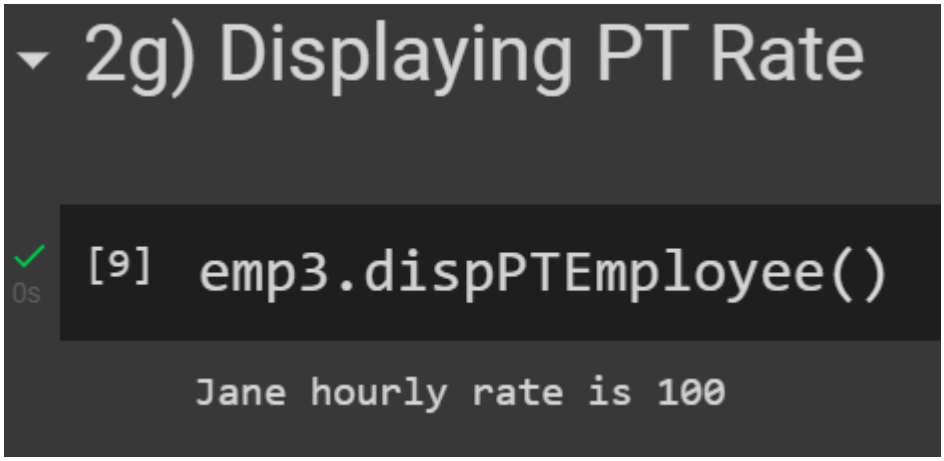
E. DISPLAYING EMPLOYEE 1 ATTRIBUTES



F. DISPLAYING FT COUNT

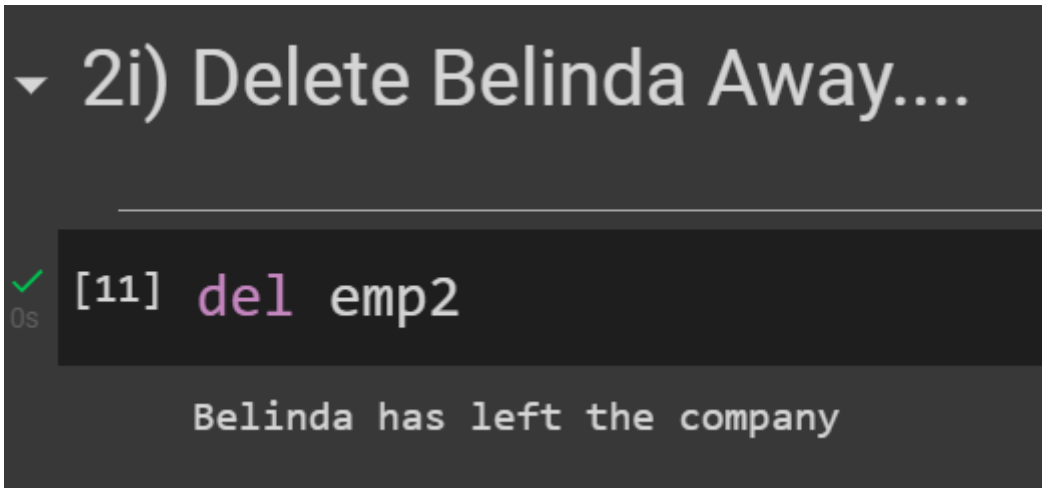


G. DISPLAYING PT RATE

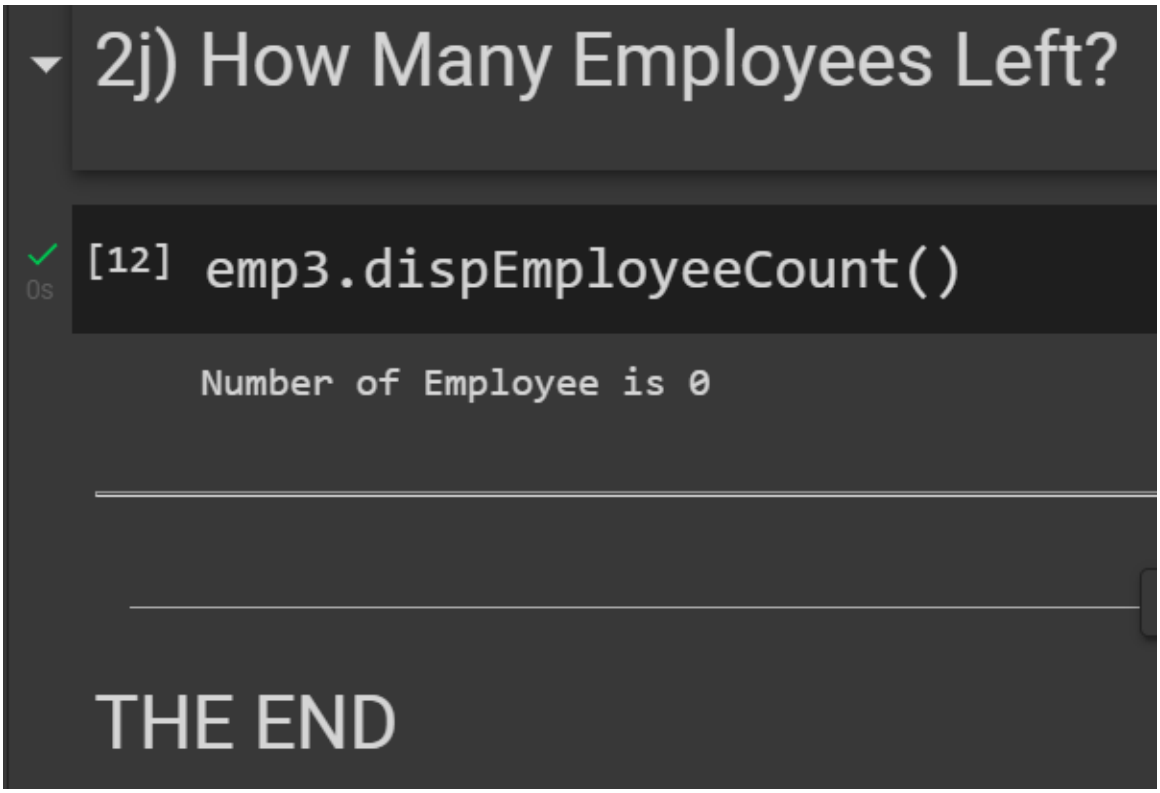


H. DELETE ALLY AWAY

I. DELETE BELINDA AWAY



J. HOW MANY EMPLOYEES LEFT?



ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.