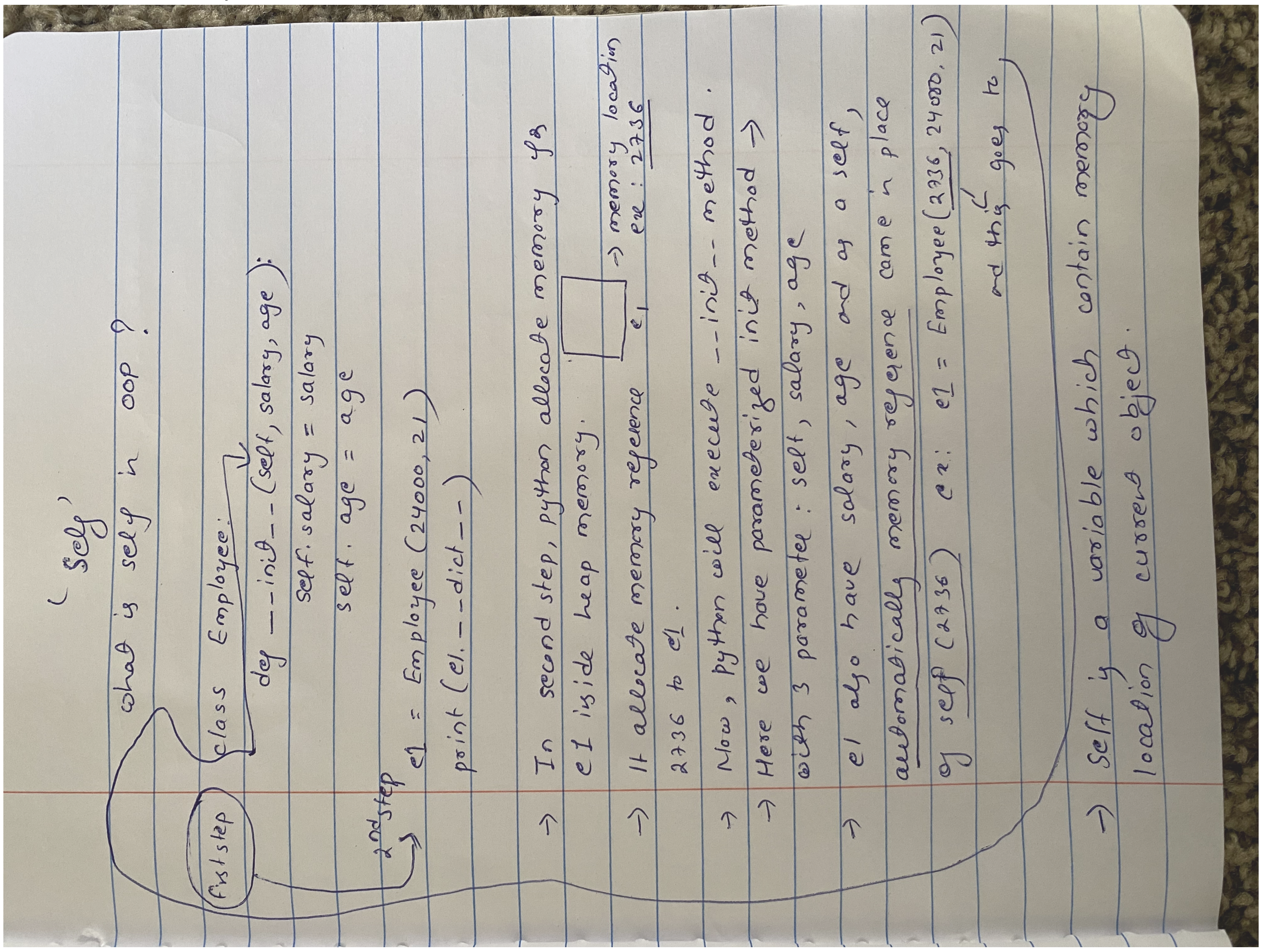
How self work in memory, what is self?



'Self'

what is self in OOP ?

First step

class Employee:
    def __init__(self, salary, age):
        self.salary = salary
        self.age = age

2nd step

e1 = Employee (24000, 21)
print (e1.__dict__)

→ In second step, python allocate memory for e1 inside heap memory.

→ memory location ex : 2736

e1

→ It allocate memory reference 2736 to e1.

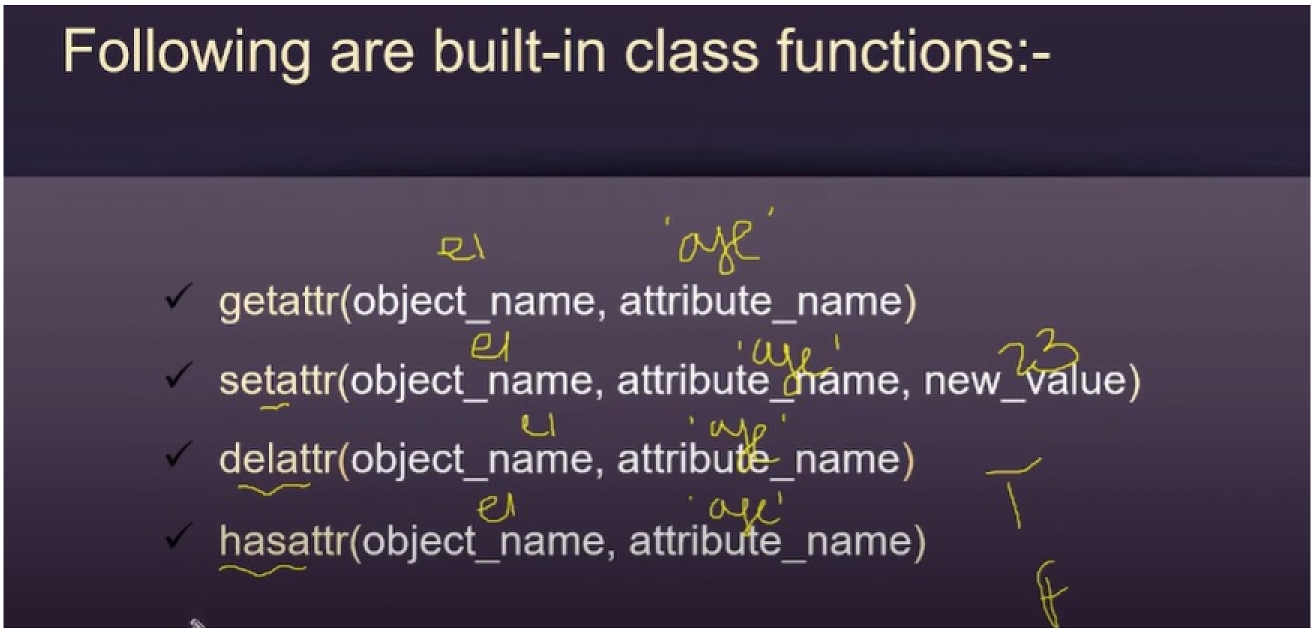
→ Now, python will execute __init__ method.

→ Here we have parameterized init method → with 3 parameter : self, salary, age

→ e1 also have salary, age and as a self, automatically memory reference came in place of self (2736) ex: e1 = Employee(2736, 24000, 21) and this goes to

→ Self is a variable which contain memory location of current object.

Built in class



Following are built-in class functions:-

- ✓ getattr(object_name, attribute_name)
- ✓ setattr(object_name, attribute_name, new_value)
- ✓ delattr(object_name, attribute_name)
- ✓ hasattr(object_name, attribute_name)

e1
'age'
e1
'age'
23
e1
'age'
T
e1
'age'
F

In [2]:
```
#Example
class student:

    school_name="Texas intl school"       #class variable

    def __init__(self,name,age,gpa):    #constructor and instance varibale
        self.name=name
        self.age=age
        self.gpa=gpa

    def display(self):     #method
        print("Student name is {} and age is {} and achieved {} GPA".format(self.name,self.age,self.gpa))

#create object
student1=student("mark",21,3.25)
```

In [3]:
```
print(student1.gpa)
```
```
3.25
```
In [4]:
```
print(student1.school_name)
```
```
Texas intl school
```
In [8]:
```
#lets use getattr -- get attribute
print(getattr(student1,"age"))
```
```
21
```
In [11]:
```
#now lets use setattr - to update
setattr(student1,"age",33)
```
In [12]:
```
#does it change, lets use __dict__
print(student1.__dict__)
```
```
{'name': 'mark', 'age': 33, 'gpa': 3.25}
```
In [13]:
```
#see that gets changed.
```
In [14]:
```
#lets use delattr
delattr(student1,"age")
```
In [15]:
```
print(student1.__dict__)
```
```
{'name': 'mark', 'gpa': 3.25}
```
In [16]:
```
#see age has been deleted
```
In [17]:
```
#lets use hasattr
print(hasattr(student1,"gpa"))
print(hasattr(student1,"age"))
```
```
True
False
```
In [ ]:
```
#True means yes- gpa attribute is in student1 object and false means it is not available.
```