Lets create Class, objects and lets import datetime too

In [6]:
```
from datetime import date

class Person:
    def __init__(self,name,surname,birthdate,address,phone,email):
        self.name=name
        self.surname=surname
        self.birthdate=birthdate
        self.address=address
        self.phone=phone
        self.email=email

#lets create method, which will calculate age
    def age(self):
        today=date.today()
        age=today.year-self.birthdate.year
        return age
```

In [8]:
```
#creating class
person1= Person("ram","thapa",date(1988,10,1),"Baneshwor",56782219,"ramthapa@gmail.com")
```

In [9]:
```
print(person1.age)
```

```
<bound method Person.age of <__main__.Person object at 0x000001CF7C8BDD50>>
```

In [10]:
```
person1.age()
```

Out[10]:
```
35
```

In [11]:
```
print(person1.birthdate)
```

```
1988-10-01
```

In [15]:
```
#If suppose, i need to return 35 years, 11 month and 5 days (in this way in this pattern?)
from datetime import date

class Person:
    def __init__(self,name,surname,birthdate,address,phone,email):
        self.name=name
        self.surname=surname
        self.birthdate=birthdate
        self.address=address
        self.phone=phone
        self.email=email

#lets create method, which will calculate age
    def age(self):
        today=date.today()
        age=today.year-self.birthdate.year
        month=today.month-self.birthdate.month
        day=today.day-self.birthdate.day
        return f"{age} years,{month}months and {day}days"
```

In [16]:
```
person2= Person("shyam","thapa",date(1988,10,1),"Baneshwor",56782219,"ramthapa@gmail.com")
```

In [17]:
```
person2.age()
```

Out[17]:
```
'35 years,0months and 14days'
```

In [18]:
```
#Here we learn how to import datetime and revise creating class and objects, methods.
#and learn how to calculate age using methods.
```

getattr, setattr and hasattr

In [23]:
```
#example of getattr (first lets see example and we will look into defination)

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

employee1=Employee(21,"ram")

print(getattr(employee1,"age"))
print(getattr(employee1,"name"))
print(getattr(employee1,"salary",0))
```

```
21
ram
0
```

In [24]:
```
#see if we did not have anything to print like instead of error message its showing zero
#Or we can do this, now I think we understand use of getattr (its get attribute)
print(getattr(employee1,"salary","sorry there is no salary"))
```

```
sorry there is no salary
```

In Python, getattr is a built-in function used to retrieve the value of an attribute from an object. It takes three arguments:

The object from which you want to retrieve the attribute. The name of the attribute you want to access as a string. An optional default value (if the attribute doesn't exist on the object).

In [26]:
```
#now setattr - set attribute
#we can set attribute after we create class and object too like

class Home:
   pass

livingroom=Home()
```

In [27]:
```
#now we have not intialize anything in class and not in object

setattr(livingroom,"color","red")
```

In [28]:
```
print(livingroom.color)
```

```
red
```

In [29]:
```
#But, main use of setattr is to update the object:
class Person:

  # Set attributes during initialization
  def __init__(self, name, age):
    self.name = name
    self.age = age

# Create a Person object
p1 = Person('John', 25)

# Use setattr to update the age
setattr(p1, 'age', 26)

print(p1.name) # 'John'
print(p1.age) # 26
```

```
John
26
```

In [30]:
```
#hasattr - Used to check if an attribute exists in an object.Returns True if attribute exists, else False.
```

In [32]:
```
class Person:

  # Set attributes during initialization
  def __init__(self, name, age):
    self.name = name
    self.age = age

# Create a Person object
p1 = Person('John', 25)
```

In [33]:
```
print(hasattr(p1,"age"))
```

```
True
```

In [34]:
```
print(hasattr(p1,"salary"))
```

```
False
```