In [5]:
```
class Cookie:
    pass

cookie1 = Cookie()
type (cookie1)
```

Out[5]:
```
__main__.Cookie
```

In [8]:
```
class Cookie:
    brand = ""
    color = ""

cookie1 = Cookie()
cookie1.brand = "Oreo"
cookie1.color = "Black"

cookie2 = Cookie()
cookie2.brand = "Maria"
cookie2.color = "Brown"

print (cookie1.brand,cookie2.brand)
```

```
Oreo Maria
```

In [9]:
```
class Cookie:
   def _init_(self,marca,color):
    self.brand = marca
    self.color = color

cookie1 = Cookie()
cookie1.brand = "Oreo"
cookie1.color = "Black"

cookie2 = Cookie()
cookie2.brand = "Maria"
cookie2.color = "Brown"

print (cookie1.brand,cookie2.brand)
```

```
Oreo Maria
```

In [13]:
```
class Cookie:
    def __init__(self,marca,color):
        self.brand = marca
        self.color = color

    def imprimir (self) :
        print('galleta: ', self.brand, self.color)

#--------------inicio programa
cookie1 = Cookie("Oreo","White")
cookie2 = Cookie("Maria", "Torrat")
print (cookie1.brand,cookie2.brand)

cookie1.imprimir()
```

```
Oreo Maria
galleta:  Oreo White
```

In [4]:
```python
class Galleta:
    def __init__(self):
        self.chocolate = False
        print("Se ha creado una galleta")


#------------------inici

galleta = Galleta()

print (galleta.chocolate)
```

```
Se ha creado una galleta
False
```

In [ ]:
```python
class Galleta:
    def __init__(self):
        self.chocolate = False
        print("Se ha creado una galleta")
    def cambiar (self,tipo):


#------------------inici

galleta = Galleta()

print (galleta.chocolate)
```

In [15]:
```python
class Cookie:
    def __init__(self,marca,color, forma = 'square'):
        self.brand = marca
        self.color = color
        self.shape = forma
        self.stock = 0

    def imprimir (self) :
        print('galleta: ', self.brand, self.color, self.shape)

#--------------inicio programa
cookie1 = Cookie("Oreo","White")
cookie2 = Cookie("Maria", "Torrat", "Rounded")
print (cookie1.brand,cookie2.brand)

cookie1.imprimir()
cookie2.imprimir()
```

```
Oreo Maria
galleta:  Oreo White square
galleta:  Maria Torrat Rounded
```

In [16]:
```
class Cookie:
    def __init__(self,marca,color, forma = 'square'):
        self.brand = marca
        self.color = color
        self.shape = forma
        self.stock = 0

    def imprimir (self) :
        print('galleta: ', self.brand, self.color, self.shape)

    def comprar (self, n) :
        self.stock += n
        print("quedan : ", self.stock)

    def vender (self,n) :
        self.stock -= n
        print("quedan : ", self.stock)

#--------------inicio programa
cookie1 = Cookie("Oreo","White")
cookie2 = Cookie("Maria", "Torrat", "Rounded")
print (cookie1.brand,cookie2.brand)

cookie1.imprimir()
cookie2.imprimir()

cookie1.comprar(10)
cookie1.vender(4)
```

```
Oreo Maria
galleta:  Oreo White square
galleta:  Maria Torrat Rounded
quedan :  10
quedan :  6
```

In [1]:
```
class Complejo :
    def __init__ (self, realpart, imagepart):
        self.r = realpart
        self.i = imagepart

    def imprimeReal (self):
        print ("parte real es ",self.r)

    def imprimeImag (self):
        print ("la parte imaginaria es ",self.i)

#------------------------inici
c1= Complejo (2,3)

c1.imprimeReal()
c1.imprimeImag()
```

```
parte real es  2
la parte imaginaria es  3
```

In [6]:
```
class Pelicula:
    def __init__(self,titulo,duracion,lanzam):
        self.titulo = titulo
        self.duracion = duracion
        self.lanzam = lanzam

    def imprimir (self):
        print ("Peli: ", self.titulo,
               "duración: ", self.duracion,
               "año: ", self.lanzam)

    def __str__(self):
        return (f"peli: {self.titulo} duracion: {self.duracion} año: {self.lanzam}"

#-----------------------------inici

p1=Pelicula("Matrix",120,2001)
p2=Pelicula("MatrixIII", 106, 2010)
p1.imprimir()
print (p1)
```

```
Peli:  Matrix duración:  120 año:  2001
peli: Matrix duracion: 120 año: 2001
```

In [9]:
```
class Catalogo:

    def __init__(self,peliculas = []):
        self.peliculas = peliculas

    def agregar (self,p) :
        self.peliculas.append(p)

    def mostrar (self) :
        for p in self.peliculas:
            print(p)


catalogo = Catalogo ([p1, p2])
catalogo.mostrar()
catalogo.agregar(Pelicula("MatrixIII", 106, 2010))
```

```
peli: Matrix duracion: 120 año: 2001
peli: MatrixIII duracion: 106 año: 2010
```

In [24]:
```
class Car:
    def __init__(self, marca, modelo, color, matricula):
        self.marca = marca
        self.modelo = modelo
        self.color = color
        self.matricula = matricula

    def __str__ (self):
        return ("marca{},modelo{}, color{}, matricula{}".format(self.marca,self.mod


# 1. llista_tots : Mostra tota la llista de cotxes amb totes les dades i
# ben posades en columnes
# 2. cerca_matricula: Mostra les dades d'un sol cotxe cercant per
# matrícula.


class Cars :
    def __init__ (self, cotxes=[]):
        self.cotxes = cotxes

    def agregar (self,c) :
        self.cotxes.append (c)

    def llistar_tots (self) :
        for c in self.cotxes:
            print(c)

    def cerca_matricula (self, matricula):
        for c in self.cotxes:
            if c.matricula == matricula:
                print("Trobada",c)

    def __find__ (self.matricula)


c1 = Car ("TOYOTA", "YARIS","ROJO","6789-CYR")
c2 = Car ("TOYOTA", "YARIS","VERDE","333-CYR")
cotxes = Cars ([c1,c2])
cotxes.llistar_tots()
cotxes.cerca_matricula("6789-CYR")
```

```
marcaTOYOTA,modeloYARIS, colorROJO, matricula6789-CYR
marcaTOYOTA,modeloYARIS, colorVERDE, matricula333-CYR
Trobada marcaTOYOTA,modeloYARIS, colorROJO, matricula6789-CYR
```

In [ ]:

```
class Car:
    def __init__(self, marca, modelo, color, matricula):
        self.marca = marca
        self.modelo = modelo
        self.color = color
        self.matricula = matricula

    def __str__ (self):
        return ("marca{},modelo{}, color{}, matricula{}".format(self.marca,self.mod


# 1. llista_tots : Mostra tota la llista de cotxes amb totes les dades i
# ben posades en columnes
# 2. cerca_matricula: Mostra les dades d'un sol cotxe cercant per
# matrícula.


class Cars :
    def __init__ (self, cotxes=[]):
        self.cotxes = cotxes

    def agregar (self,c) :
        self.cotxes.append (c)

    def llistar_tots (self) :
        for c in self.cotxes:
            print(c)

    def cerca_matricula (self, matricula):
        for c in self.cotxes:
            if c.matricula == matricula:
                print("Trobada",c)


c1 = Car ("TOYOTA", "YARIS","ROJO","6789-CYR")
c2 = Car ("TOYOTA", "YARIS","VERDE","333-CYR")
cotxes = Cars ([c1,c2])
cotxes.llistar_tots()
cotxes.cerca_matricula("6789-CYR")
```

In [36]:
```
class Car:
    def __init__(self, marca, modelo, color, matricula):
        self.marca = marca
        self.modelo = modelo
        self.color = color
        self.matricula = matricula

    def __str__ (self):
        return ("marca{},modelo{}, color{}, matricula{}".format(self.marca,self.mod


# 1. llista_tots : Mostra tota la llista de cotxes amb totes les dades i
# ben posades en columnes
# 2. cerca_matricula: Mostra les dades d'un sol cotxe cercant per
# matrícula.


class Cars :
    def __init__ (self, cotxes=[]):
        self.cotxes = cotxes

    def __find (self, matricula):
        for c in self.cotxes:
            if c.matricula == matricula:
                return(True)
        return(False)

    def agregar (self,c) :
        if not self.__find (c.matricula):
            self.cotxes.append (c)


    def llistar_tots (self) :
        for c in self.cotxes:
            print(c)

    def cerca_matricula (self, matricula):
        if self.__find (matricula):
            print ("Trobat")
        else:
            print ("No trobat")




c1 = Car ("TOYOTA", "YARIS","ROJO","6789-CYR")
c2 = Car ("TOYOTA", "YARIS","VERDE","333-CYR")
cotxes = Cars ([c1,c2])
cotxes.llistar_tots()
cotxes.cerca_matricula("6789-CYR")
cotxes.agregar(Car("Seat", "Leon", "Azul", "6845_JFH"))
cotxes.llistar_tots()
```

```
marcaTOYOTA,modeloYARIS, colorROJO, matricula6789-CYR
marcaTOYOTA,modeloYARIS, colorVERDE, matricula333-CYR
Trobat
marcaTOYOTA,modeloYARIS, colorROJO, matricula6789-CYR
marcaTOYOTA,modeloYARIS, colorVERDE, matricula333-CYR
marcaSeat,modeloLeon, colorAzul, matricula6845_JFH
```

In [47]:

```python
class Targeta:
    def __init__ (self, clave, titular):
        self.clave = clave
        self.__titular = titular
        self.__saldo = 0

    def mostrar_titular(self):
        print (self.__titular)

    def agregar_titular (self):
        return (self.__titular)

    def obtener_titular (self):
        return (self.__titular)

    def cambiar_titular (self, titular):
        self.__titular = titular

    def mostrar_saldo (self):
        print (self.__saldo)

    def cambiar_saldo (self, saldo):
        self.__saldo = saldo

    def __str__ (self):
        return f"{self.clave} titular:{self.__titular} saldo:{self.__saldo}"


t1 = Targeta ("1111-4444-7777", "Marisa Garcia")
t1.cambiar_titular ("Marisa çGarcia Castellón")
t1.cambiar_saldo(300)
t1.mostrar_titular()
t1.mostrar_saldo()
print(t1)
```

```
Marisa çGarcia Castellón
300
1111-4444-7777 titular:Marisa çGarcia Castellón saldo:300
```

In [48]:
```
class Perro:
    def sonido(self):
        print('Guauuuuu!!!')
class Gato:
    def sonido(self):
        print('Miaaauuuu!!!')
class Vaca:
    def sonido(self):
        print('Múuuuuuuu!!!')
#----------------------------------- Definición de funciones
def a_cantar(animales):
    for animal in animales:
        animal.sonido()
#----------------------------------- Inicio del programa
perro1 = Perro() ; perro2 = Perro()
gato1 = Gato() ; gato2 = Gato()
vaca = Vaca()
granja = [perro1, gato1, vaca, gato2, perro2]
a_cantar(granja)
```

```
Guauuuuu!!!
Miaaauuuu!!!
Múuuuuuuu!!!
Miaaauuuu!!!
Guauuuuu!!!
```

In [ ]:
```
class A :
    def sabe_nadar (self):
        print ("-- mira como nado --")
class B :
    def sabe_volar (self):
        print ("-- mira como vuelo --")
class C (A) :
    def sabe_patinar (self):
        print ("-- mira como patino --")

a = C()
x.
```

In [1]:
```
class Producto:
    def __init__(self,referencia,nombre,pvp,descripcion):
        self.referencia = referencia
        self.nombre = nombre
        self.pvp = pvp
        self.descripcion = descripcion
    def __str__(self):
        return f"Referencia\t {self.referencia}\n" \
        f"Nombre\t\t {self.nombre}\n" \
        f"PVP\t\t {self.pvp}\n" \
        f"Descripcion\t {self.descripcion}\n "

class Adorno(Producto):
    pass
adorno = Adorno(2034, "Vaso adornado", 15, "Vaso de porcelana")
print(adorno)
```

```
Referencia      2034
Nombre          Vaso adornado
PVP             15
Descripcion     Vaso de porcelana
```

In [2]:

```python
class Producto:
    def __init__(self,referencia,nombre,pvp,descripcion):
        self.referencia = referencia
        self.nombre = nombre
        self.pvp = pvp
        self.descripcion = descripcion
    def __str__(self):
        return f"Referencia\t {self.referencia}\n" \
                f"Nombre\t\t {self.nombre}\n" \
                f"PVP\t\t {self.pvp}\n" \
                f"Descripcion\t {self.descripcion}\n "

class Adorno(Producto):
    pass

class Alimento(Producto):
    productor = ""
    distribuidor = ""
    def __str__(self):
        return super().__str__() + \
            f"PRODUCTOR\t {self.productor}\n" \
            f"DISTRIBUIDOR\t {self.distribuidor}\n"

class Libro(Producto):
    isbn = ""
    autor = ""
    def __str__(self):
        return super().__str__() + \
            f"ISBN\t\t {self.isbn}\n" \
            f"AUTOR\t\t {self.autor}\n"

alimento = Alimento(2035, "Botella de Aceite de Oliva", 5, "250 ML")
alimento.productor = "La Aceitera"
alimento.distribuidor = "Distribuciones SA"
print(alimento)

adorno = Adorno(2034, "Vaso adornado", 15, "Vaso de porcelana")
print(adorno)
```

```
Referencia      2035
Nombre          Botella de Aceite de Oliva
PVP             5
Descripcion     250 ML
 PRODUCTOR      La Aceitera
DISTRIBUIDOR    Distribuciones SA

Referencia      2034
Nombre          Vaso adornado
PVP             15
Descripcion     Vaso de porcelana
```

In [3]:
```python
#----------------------------- Programa principal : main
class Producto:
    def __init__(self,referencia,nombre,pvp,descripcion):
        self.referencia = referencia
        self.nombre = nombre
        self.pvp = pvp
        self.descripcion = descripcion
    def __str__(self):
        return f"REFERENCIA\t {self.referencia}\n" \
               f"NOMBRE\t\t {self.nombre}\n" \
               f"PVP\t\t {self.pvp}\n" \
               f"DESCRIPCIÓN\t {self.descripcion}\n"


#
#----------------------------- Programa principal : main
p=Producto("J72w", "Tornillo y tuerca 4mm", 2.5, "blister de 20 unidades")
print(p)
```

```
REFERENCIA       J72w
NOMBRE           Tornillo y tuerca 4mm
PVP              2.5
DESCRIPCIÓN      blister de 20 unidades
```

In [4]:
```
class Textil (Producto):
    def __init__(self,referencia,nombre,pvp,descripcion, color,material):
        self.referencia = referencia
        self.nombre = nombre
        self.pvp = pvp
        self.descripcion = descripcion
        self.color = color
        self.material = material

    def __str__(self):
        return super().__str__() + \
                    f"COLOR\t\t {self.color}\n" \
                    f"MATERIAL\t {self.material}\n"

class Brico (Producto):

    def __init__(self,referencia,nombre,pvp,descripcion, medida, peso):
        self.referencia = referencia
        self.nombre = nombre
        self.pvp = pvp
        self.descripcion = descripcion
        self.medida = medida
        self.peso = peso

    def __str__(self):
        return super().__str__() + \
                    f"MEDIDA\t\t {self.medida}\n" \
                    f"PESO\t\t {self.peso}\n"


t1= Textil ("ref1", "cortina", 12, "de algodon", "blaco", "algodon")
print (t1)
b1 = Brico ("ref456", "Tornillo", 0.5, "rosca", 2, 100)
print (b1)
```

```
REFERENCIA       ref1
NOMBRE           cortina
PVP              12
DESCRIPCIÓN      de algodon
COLOR            blaco
MATERIAL         algodon

REFERENCIA       ref456
NOMBRE           Tornillo
PVP              0.5
DESCRIPCIÓN      rosca
MEDIDA           2
PESO             100
```

In [5]:
```
class Carro:

    def __init__ (self, carrito=[]):
        self.carrito = carrito


    def mostrar_carrito(self):
        print ("--------CARRITO-------")
        for p in self.carrito:
            print (p)

    def agregar_carrito (self, producto):
        self.carrito.append(producto)

    def total_compra (self):
        suma = 0
        for p in self.carrito:
            suma += p.pvp
        print (f"Total carro: {suma}")

    def eliminar_de_carrito (self,self.referencia):
        nueva = []
        for p in self.carrito:
            if referencia != p.referencia:
                nueva.append(p)

compra = Carro ([t1,b1])
compra.agregar_carrito(alimento)
compra.mostrar_carrito()
compra.total_compra()
compra.eliminar_de_carrito(ref456)
```

```
  Cell In[5], line 21
    def eliminar_de_carrito (self,self.referencia):
                                     ^
SyntaxError: invalid syntax
```

In [12]:
```
class Client :
    nom = ""
    cognoms = ""

c = Client()
c.nom = "Marta"
c.cognoms = "Lopez"

print(c.__dict__)

c.__dict__["nom"] = "Montse"
c.__dict__["telf"] = "932193499"

#print(c.__dict__)
print (c.nom, c.telf)
print(c.__dict__)
```

```
{'nom': 'Marta', 'cognoms': 'Lopez'}
Montse 932193499
{'nom': 'Montse', 'cognoms': 'Lopez', 'telf': '932193499'}
```

In [ ]:
```
class Client :
    def __init__(self,nom,cognom,telf)
        self.nom = nom
        self.cognom = cognom
        self.telf = telf

c = Client()
c.nom = "Marta"
c.cognoms = "Lopez"

print(c.__dict__)

c.__dict__["nom"] = "Montse"
c.__dict__["telf"] = "932193499"

#print(c.__dict__)
print (c.nom, c.telf)
print(c.__dict__)
```