In [11]:
```
#it is clearly seen that self and obj is referring to the same object
class check:
    def __init__(self):
        print("Address of self = ",id(self))

obj = check()
print("Address of class object = ",id(obj))
```

```
Address of self =  2358195432752
Address of class object =  2358195432752
```

In [12]:
```
# Self is always required as the first argument
class check:
    def __init__(self):
        print("This is Constructor")

obj = check()
print("Worked fine")
```

```
This is Constructor
Worked fine
```

In [1]:
```
class Car:
    def __init__(self,name,color): # Constructor
        self.name=name
        self.color=color
    def display(self):
        print("Car Name:",self.name,"Color:",self.color)
    def __del__(self):  # Destructor
        print("object Destroyed")

c1=Car("Audi","Blue")
c2=Car("BMW","Black")
c1.display()
c2.display()
del c1
c1.display()
```

```
Car Name: Audi Color: Blue
Car Name: BMW Color: Black
object Destroyed

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-bb17cf260da0> in <module>
     13 c2.display()
     14 del c1
---> 15 c1.display()

NameError: name 'c1' is not defined
```

In [ ]:
```
############### Generators #####################
```

In [2]:
```
def fun():
    yield 10
    yield 20
    yield 30
    yield 40
for i in fun():
    print(i)
```

```
10
20
30
40
```

In [14]:
```
def city_generator():
    yield "Hamburg"
    yield "Konstanz"
    yield "Berlin"
    yield "Zurich"
    yield "Schaffhausen"
    yield "Stuttgart"

city = city_generator()
print(next(city))
print(next(city))
print(next(city))
print(next(city))
print(next(city))
print(next(city))
print(next(city))
```

```
Hamburg
Konstanz
Berlin
Zurich
Schaffhausen
Stuttgart

---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-14-6f4cd258dfbc> in <module>
     14 print(next(city))
     15 print(next(city))
---> 16 print(next(city))

StopIteration:
```

In [15]:
```python
def table(n):
    for i in range(1,11):
        yield n*i

for i in table(15):
    print(i)
```

```
15
30
45
60
75
90
105
120
135
150
```

## Data Hiding /Encapsulation

In [7]:
```python
# Data Hiding, Public,Protected & Private Variables
class MyClass(object): # Defining class
    def __init__(self, x, y, z):
        self.var1 = x # public data member
        self._var2 = y # protected data member
        self.__var3 = z # private data member

obj = MyClass(3,4,5)
print(obj.var1)
print(obj._var2)
```

```
3
4
```

In [11]:
```python
print(obj.__var3)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-11-8aa45be4e347> in <module>
----> 1 print(obj.__var3)

AttributeError: 'MyClass' object has no attribute '__var3'
```

In [13]:
```python
print(obj._MyClass__var3)
```

```
5
```

In [ ]:
```python
############
```

In [ ]:
```
# The internal representation of an object of foo class ①-⑥ is hidden outside th
class foo:
    def __init__(self, a, b):
        self.a = a
        self.b = b
    def add(self):
        return self.a * self.b

# Any accessible member (data/method) of an object of foo is restricted and can c
# Implementation of add() method is hidden → Abstraction.
foo_object = foo(3,4)
foo_object.add()
```

In [8]:
```
class Human:
    species="Homo Sapiens" #Class Variable
    def __init__(self,name,age,gender): #Constructor Method
        self.name=name
        self.age=age
        self.gender=gender
```

In [9]:
```
x=Human('Parth',32,"Male")   #Object1 Created
y=Human('Aakash',21,'Male')  #Object2 Created
print(x.name)
print(y.age)
```

```
Parth
21
```

In [5]:
```
del Human
```

In [6]:
```
x=Human('Parth',32,"Male")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-6-ccad441254f6> in <module>
----> 1 x=Human('Parth',32,"Male")

NameError: name 'Human' is not defined
```

In [7]:
```
class Employee:
    def __init__(self):
        print('Employee Class created')
    def __del__(self):
        print('Destructor called, Employee deleted')
```

In [8]:
```
obj=Employee()
```

```
Employee Class created
```

In [9]:
```
del obj
```

```
Destructor called, Employee deleted
```

In [10]:
```
#Use of __init__ Method:
class Disp:
    var=10
    def display(self):
        print('In Class Method....')

obj=Disp()
print(obj.var)
obj.display()
```

```
10
In Class Method....
```

In [11]:
```
class Disp:
    def __init__(self,val):
        print('In Class Method....')
        self.val=val
        print('The value is:',self.val)

obj=Disp(10)
```

```
In Class Method....
The value is: 10
```

```
###############################
```

In [12]:
```
class Students:
    def __init__(self,name):
        self.name=name
        self.marks=[]
    def entermarks(self):
        for i in range(3):
            m=int(input('Enter marks of %s in Subject %d:'%(self.name,i+1)))
            self.marks.append(m)
    def display(self):
        print(self.name,'got',self.marks)
```

In [14]:
```
s1=Students('Parth')
s1.entermarks()
s1.display()
s2=Students('Aakash')
s2.entermarks()
s2.display()
```

```
Enter marks of Parth in Subject 1:21
Enter marks of Parth in Subject 2:22
Enter marks of Parth in Subject 3:32
Parth got [21, 22, 32]
Enter marks of Aakash in Subject 1:12
Enter marks of Aakash in Subject 2:14
Enter marks of Aakash in Subject 3:21
Aakash got [12, 14, 21]
```

In [10]:
```
class Student:
    branch = "IT"   #class variable
    def __init__(self, rollno, name): #constructor method
        self.rollno = rollno         #instance variable
        self.name = name             #instance variable

    def enter_marks(self): ##function to enter marks of 5 subject
        self.marks = []
        for i in range(5):
            self.marks.append(int(input('Enter marks: ')))

    def display(self):               #instance method
        print('Name: ', self.name, 'Rollno:', self.rollno, 'Branch', self.branch)
        if len(self.marks)>0:
            print(self.marks)

    def calculate_avg(self):         #instance method
        sum = 0
        for i in self.marks:
            sum += i
        print('Average marks', sum/5)

    def __del__(self):            # destructor method
        print('Object destroyed')
```

In [9]:
```python
s1 = Student(1, 'John')
s2 = Student(2, 'Arman')
s1.enter_marks()
s1.display()
s1.calculate_avg()
s2.enter_marks()
s2.display()
s2.calculate_avg()
```

```
Enter marks: 21
Enter marks: 22
Enter marks: 23
Enter marks: 24
Enter marks: 25
Name:  John Rollno: 1 Branch IT
[21, 22, 23, 24, 25]
Average marks 23.0
Enter marks: 15
Enter marks: 19
Enter marks: 24
Enter marks: 21
Enter marks: 22
Name:  Arman Rollno: 2 Branch IT
[15, 19, 24, 21, 22]
Average marks 20.2
```

```
In [17]: ##Program to create class string and make methods to count uppercase,lowercase,vo
class String:
    def __init__(self):
        self.vowels=0
        self.consonants=0
        self.spaces=0
        self.uppercase=0
        self.lowercase=0
        self.string=input("Enter a string:")
    def count_uppercase(self):
        for letter in self.string:
            if letter.isupper():
                self.uppercase+=1
    def count_lowercase(self):
        for letter in self.string:
            if letter.islower():
                self.lowercase+=1
    def count_vowels(self):
        for letter in self.string:
            if letter in 'aeiouAEIOU':
                self.vowels+=1
    def count_spaces(self):
        for letter in self.string:
            if letter.isspace():
                self.spaces+=1
    def count_consonants(self):
        for letter in self.string:
            if (letter.isalpha()) and (letter not in 'aeiouAEIOU'):
                self.consonants+=1
    def compute_stats(self):
        self.count_uppercase()
        self.count_lowercase()
        self.count_vowels()
        self.count_consonants()
        self.count_spaces()
    def show_stats(self):
        print('Vowels:%d'%self.vowels)
        print('Consonants:%d'%self.consonants)
        print('Spaces:%d'%self.spaces)
        print('Uppercase:%d'%self.uppercase)
        print('Lowercase:%d'%self.lowercase)
```

```
In [18]: s=String()
s.compute_stats()
s.show_stats()
```

```
Enter a string:This is Python class. You are learning OOP.
Vowels:13
Consonants:21
Spaces:7
Uppercase:6
Lowercase:28
```

```
In [19]: # Program to create class Atm and make methods to create pin,
         # change pin,Check balance, Withdraw.
         class Atm:
         # constructor
             def __init__(self):
                 self.pin=''
                 self.balance=0
                 self.menu()

             def menu(self):
                 user_input=input("""
                 Hi how can I help you?
                 1. Press 1 to create pin
                 2. Press 2 to change pin
                 3. Press 3 to check balance
                 4. Press 4 to withdraw
                 5. Anything else to exit
                 """)
                 if user_input=='1':
                     self.create_pin()
                 elif user_input=='2':
                     self.change_pin()
                 elif user_input =='3':
                     self.check_balance()
                 elif user_input=='4':
                     self.withdraw()
                 else:
                     exit()

             def create_pin(self):
                 user_pin=input('enter your pin')
                 self.pin=user_pin
                 user_balance=int(input('enter balance'))
                 self.balance=user_balance
                 print('pin created successfully')
                 self.menu()

             def change_pin(self):
                 old_pin=input('enter old pin')
                 if old_pin ==self.pin:
                     # let him change the pin
                     new_pin=input('enter new pin')
                     self.pin=new_pin
                     print('pin change successful')
                     self.menu()
                 else:
                     print('enter correct pin')
                     self.menu()

             def check_balance(self):
                 user_pin=input('enter your pin')
                 if user_pin==self.pin:
                     print('your balance is ',self.balance)
                 else:
                     print('enter correct pin')
                 self.menu()
```

```
    def withdraw(self):
        user_pin=input('enter the pin')
        if user_pin==self.pin:
            # allow to withdraw
            amount=int(input('enter the amount'))
            if amount<=self.balance:
                self.balance=self.balance-amount
                print('withdrawl successful.balance is',self.balance)
            else:
                print('increase your balance')
        else:
            print('enter correct pin')
        self.menu()
```

In [20]:
```
sbi=Atm()
```

```
        Hi how can I help you?
        1. Press 1 to create pin
        2. Press 2 to change pin
        3. Press 3 to check balance
        4. Press 4 to withdraw
        5. Anything else to exit
        1
enter your pin2345
enter balance25000
pin created successfully

        Hi how can I help you?
        1. Press 1 to create pin
        2. Press 2 to change pin
        3. Press 3 to check balance
        4. Press 4 to withdraw
        5. Anything else to exit
        2
enter old pin2345
enter new pin1234
pin change successful

        Hi how can I help you?
        1. Press 1 to create pin
        2. Press 2 to change pin
        3. Press 3 to check balance
        4. Press 4 to withdraw
        5. Anything else to exit
        3
enter your pin1234
your balance is  25000

        Hi how can I help you?
        1. Press 1 to create pin
        2. Press 2 to change pin
        3. Press 3 to check balance
        4. Press 4 to withdraw
        5. Anything else to exit
        4
enter the pin1234
enter the amount20000
withdrawl successful.balance is 5000

        Hi how can I help you?
        1. Press 1 to create pin
        2. Press 2 to change pin
        3. Press 3 to check balance
        4. Press 4 to withdraw
        5. Anything else to exit
        5
```

In [5]:
```
""" Bank Account class:
Create a Python class called BankAccount which represents a bank account,
having as attributes: accountNumber (numeric type), name
(name of the account owner as string type), balance.
Create a constructor with parameters: accountNumber, name, balance.
Create a Deposit() method which manages the deposit actions.
Create a Withdrawal() method  which manages withdrawals actions.
Create an bankFees() method to apply the bank fees with a
percentage of 5% of the balance account.
Create a display() method to display account details.
Give the complete code for the  BankAccount class.
"""

class BankAccount:
    # create the constuctor with parameters: accountNumber, name and balance
    def __init__(self,accountNumber, name, balance):
        self.accountNumber = accountNumber
        self.name = name
        self.balance = balance

    # create Deposit() method
    def Deposit(self , d ):
        self.balance = self.balance + d

    # create Withdrawal method
    def Withdrawal(self , w):
        if(self.balance < w):
            print("impossible operation! Insufficient balance !")
        else:
            self.balance = self.balance - w
    # create bankFees() method
    def bankFees(self):
        self.balance = (95/100)*self.balance

    # create display() method
    def display(self):
        print("Account Number : " , self.accountNumber)
        print("Account Name : " , self.name)
        print("Account Balance : " , self.balance)

# Testing the code :
newAccount = BankAccount(2178514584, "Albert" , 2700)
# Creating Withdrawal Test
newAccount.Withdrawal(300)
# Create deposit test
newAccount.Deposit(200)
# Display account informations
newAccount.display()
```

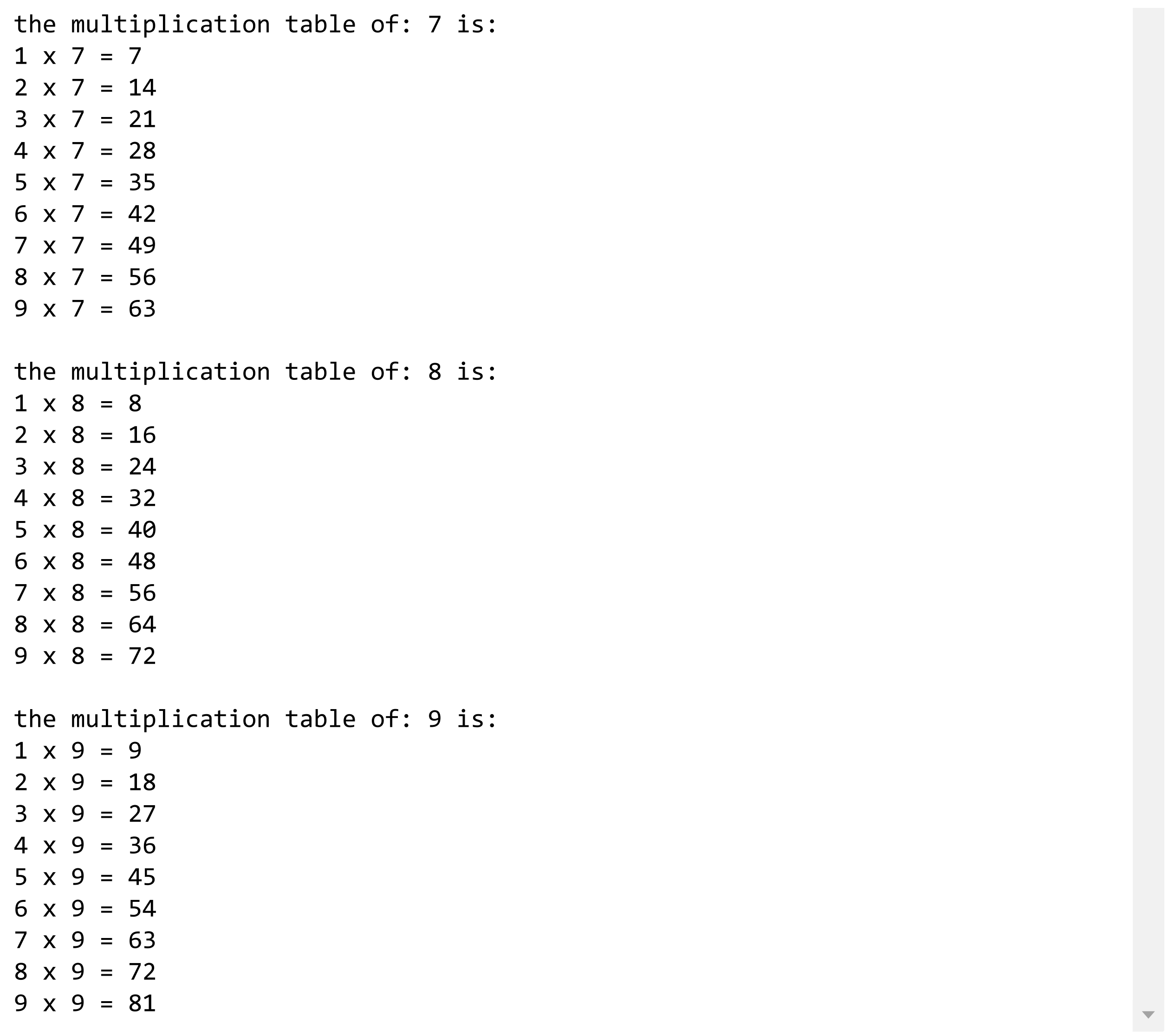
```
Account Number :  2178514584
Account Name :  Albert
Account Balance :  2600
```

```
In [6]: """ Computation class:
        1 - Create a Coputation class with a default constructor
        (without parameters) allowing to perform
        various calculations on integers numbers.
        2 - Create a method called Factorial() which allows to calculate the factorial of
        integer. Test the method by instantiating the class.
        3 - Create a method called Sum() allowing to calculate the sum of the first n
        integers 1 + 2 + 3 + .. + n. Test this method.
        4 - Create a method called testPrim() in  the Calculation
        class to test the primality of a given integer. Test this method.
        4 - Create  a method called testPrims() allowing to test if
        two numbers are prime between them.
        5 - Create a tableMult() method which creates and displays
        the multiplication table of a given integer.
        Then create an allTablesMult() method to display all the
        integer multiplication tables 1, 2, 3, ..., 9.
        6 - Create a static listDiv() method that gets all
        the divisors of a given integer on new list called  Ldiv.
        Create another listDivPrim() method that gets all
        the prime divisors of a given integer."""


        class Computation:
            def __init__ (self):
                pass
        # --- Factorial ------------
            def factorial(self, n):
                j = 1
                for i in range (1, n + 1):
                    j = j * i
                return j

        # --- Sum of the first n numbers ----
            def sum (self, n):
                j = 1
                for i in range (1, n + 1):
                    j = j + i
                return j

        # --- Primality test of a number ------------
            def testPrim (self, n):
                j = 0
                for i in range (1, n + 1):
                    if (n% i == 0):
                        j = j + 1
                if (j == 2):
                    return True
                else:
                    return False

        # --- Primality test of two integers ------------
            def testprims (self, n, m):

                # initialize the number of commons divisors
                commonDiv = 0
                for i in range (1, n + 1):
```

```python
            if (n% i == 0 and m% i == 0):
                commonDiv = commonDiv + 1
        if commonDiv == 1:
            print ("The numbers", n, "and", m, "are co-primes")
        else:
            print ("The numbers", n, "and", m, "are not co-primes")

#---Multiplication table-------------
    def tableMult (self, k):
        for i in range (1,10):
            print (i, "x", k, "=", i * k)

# --- All multiplication tables of the numbers 1, 2, .., 9
    def allTables (self):
        for k in range (1,10):
            print ("\nthe multiplication table of:", k, "is:")
            for i in range (1,10):
                print (i, "x", k, "=", i * k)

# ----- list of divisors of an integer
    def listDiv (self, n):
        # initialization of the list of divisors
        lDiv = []
        for i in range (1, n + 1):
            if (n% i == 0):
                lDiv.append (i)
        return lDiv

# ------ list of prime divisors of an integer ----------------
    def listDivPrim (self, n):
        # initialization of the list of divisors
        lDiv = []
        for i in range (1, n + 1):
            if (n% i == 0 and self.testPrim (i)):
                lDiv.append (i)
        return lDiv

# Instantiation example
Comput= Computation ()
Comput.testprims (13, 7)
print ("List of divisors of 18:", Comput.listDiv (18))
print ("List of prime divisors of 18:", Comput.listDivPrim (18))
Comput.allTables ()
```

```
The numbers 13 and 7 are co-primes
List of divisors of 18: [1, 2, 3, 6, 9, 18]
List of prime divisors of 18: [2, 3]

the multiplication table of: 1 is:
1 x 1 = 1
2 x 1 = 2
3 x 1 = 3
4 x 1 = 4
5 x 1 = 5
6 x 1 = 6
7 x 1 = 7
8 x 1 = 8
```

```
9 x 1 = 9

the multiplication table of: 2 is:
1 x 2 = 2
2 x 2 = 4
3 x 2 = 6
4 x 2 = 8
5 x 2 = 10
6 x 2 = 12
7 x 2 = 14
8 x 2 = 16
9 x 2 = 18

the multiplication table of: 3 is:
1 x 3 = 3
2 x 3 = 6
3 x 3 = 9
4 x 3 = 12
5 x 3 = 15
6 x 3 = 18
7 x 3 = 21
8 x 3 = 24
9 x 3 = 27

the multiplication table of: 4 is:
1 x 4 = 4
2 x 4 = 8
3 x 4 = 12
4 x 4 = 16
5 x 4 = 20
6 x 4 = 24
7 x 4 = 28
8 x 4 = 32
9 x 4 = 36

the multiplication table of: 5 is:
1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
4 x 5 = 20
5 x 5 = 25
6 x 5 = 30
7 x 5 = 35
8 x 5 = 40
9 x 5 = 45

the multiplication table of: 6 is:
1 x 6 = 6
2 x 6 = 12
3 x 6 = 18
4 x 6 = 24
5 x 6 = 30
6 x 6 = 36
7 x 6 = 42
8 x 6 = 48
9 x 6 = 54
```



```
the multiplication table of: 7 is:
1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
4 x 7 = 28
5 x 7 = 35
6 x 7 = 42
7 x 7 = 49
8 x 7 = 56
9 x 7 = 63

the multiplication table of: 8 is:
1 x 8 = 8
2 x 8 = 16
3 x 8 = 24
4 x 8 = 32
5 x 8 = 40
6 x 8 = 48
7 x 8 = 56
8 x 8 = 64
9 x 8 = 72

the multiplication table of: 9 is:
1 x 9 = 9
2 x 9 = 18
3 x 9 = 27
4 x 9 = 36
5 x 9 = 45
6 x 9 = 54
7 x 9 = 63
8 x 9 = 72
9 x 9 = 81
```

In [7]:
```python
"""Write a Rectangle class in Python language, allowing you
to build a rectangle with length and width attributes.
Create a Perimeter() method to calculate the perimeter of the rectangle and a Are
method to calculate the area of the rectangle.
Create a method display() that display the length, width, perimeter and area of a
object created using an instantiation on rectangle class.
Create a Parallelepipede child class inheriting from the Rectangle
class and with a height attribute and another Volume()
method to calculate the volume of the Parallelepiped."""

class Rectangle:
    # define constructor with attributes: length and width
    def __init__(self, length , width):
        self.length = length
        self.width = width

    # Create Perimeter method
    def Perimeter(self):
        return 2*(self.length + self.width)

    # Create area method
    def Area(self):
        return self.length*self.width

    # create display method
    def display(self):
        print("The length of rectangle is: ", self.length)
        print("The width of rectangle is: ", self.width)
        print("The perimeter of rectangle is: ", self.Perimeter())
        print("The area of rectangle is: ", self.Area())
class Parallelepipede(Rectangle):
    def __init__(self, length, width , height):
        Rectangle.__init__(self, length, width)
        self.height = height

    # define Volume method
    def volume(self):
        return self.length*self.width*self.height

myRectangle = Rectangle(7 , 5)
myRectangle.display()
print("----------------------------------")
myParallelepipede = Parallelepipede(7 , 5 , 2)
print("the volume of myParallelepipede is: " , myParallelepipede.volume())
```

```
The length of rectangle is:  7
The width of rectangle is:  5
The perimeter of rectangle is:  24
The area of rectangle is:  35
----------------------------------
the volume of myParallelepipede is:  70
```

In [ ]:
```
##You owned a pizza outlet named Olly's Pizza and want to create a python program
##handle the customers and revenue. Create the following classes with the followi

Class pizza containing:

1.  init method :
To initialize the size (small, medium, large) , Toppings (corn, tomato, onion, ca
Note : One pizza can have only one size but many toppings and cheese.

2.  Price method:
To calculate the prize of the pizza in the following way:
Small=50 ,medium=100, large=200
Each topping costs 20 rupees extra, except broccoli, olives and mushroom, which a
Each type of cheese costs an extra 50 rupees.

Class Order containing:

1.  Init method:
To initialize the name, customer id of customer who place the order.

2.  Order method:
To allow customer to select pizzas with choice of toppings and cheese.

3.  Bill method:
To generate details about each pizza order by customer and the total cost of the

Note: A Customer can get multiple pizzas in one order.
```

In [16]:
```
class Pizza:
    def __init__(self, size, toppings, cheese):
        self.size = size
        self.toppings = toppings
        self.cheese = cheese

    def price(self):
        self.cost = 0
        if self.size == 'small':
            self.cost += 50
        elif self.size == 'medium':
            self.cost += 100
        else:
            self.cost += 200
        topping_prices_20 = ['corn', 'tomato', 'onion', 'capsicum']
        topping_prices_50 = ['mushroom', 'olives', 'broccoli']
        for topping in self.toppings:
            if topping in topping_prices_20:
                self.cost += 20
            else:
                self.cost += 50
        #for cheese
        self.cost += 50 * len(self.cheese)
        return self.cost

class Order:
    def __init__(self, name, customerid):
        self.name = name
        self.customerid = customerid

    def order(self, n):
        self.pizzas = []
        for i in range(n):
            toppings = []
            cheese = []
            print('Customize Pizza',i+1)
            size = input('Select size: ')
            t = int(input('How many toppings: '))
            for i in range(t):
                toppings.append(input('Enter toppings: '))
            t = int(input('How many cheese: '))
            for i in range(t):
                cheese.append(input('Enter cheese: '))
            self.pizzas.append(Pizza(size, toppings, cheese))

    def bill(self):
        self.total = 0
        count = 1
        for p in self.pizzas:
            print('Pizza', count)
            print("Size:",p.size)
            print("Toppings:",p.toppings)
            print("Cheese:",p.cheese)
            self.total += p.price()
            count += 1
        print('Total bill amount:', self.total)
```

```
number=int(input("How many pizzas you want to order: "))
order1 = Order('Tejas', 1)
order1.order(number)
order1.bill()
```

```
How many pizzas you want to order: 1
Customize Pizza 1
Select size: small
How many toppings: 3
Enter toppings: tomato
Enter toppings: onion
Enter toppings: brocolli
How many cheese: 1
Enter cheese: mozarella
Pizza 1
Size: small
Toppings: ['tomato', 'onion', 'brocolli']
Cheese: ['mozarella']
Total bill amount: 190
```

```
#Write a class called Product. The class should have fields called name,
amount, and price, holding the product’s name, the number of items of that
product in stock, and the regular price of the product. There should be a
method get_price that receives the number of items to be bought and returns a
the cost of buying that many items, where the regular price is charged for
orders of less than 10 items, a 10% discount is applied for orders of between
10 and 99 items, and a 20% discount is applied for orders of 100 or more items.
There should also be a method called make_purchase that receives the number of
items to be bought and decreases amount by that much.
```

In [21]:
```python
class Product:

    def __init__(self, name, amount, price):
        self.name = name
        self.amount = amount
        self.price = price

    def get_price(self, number_to_be_bought):
        discount = 0
        if number_to_be_bought < 10:
            pass
        elif 10 <= number_to_be_bought < 99:
            discount = 10
        else:
            discount = 20
        price = (100 - discount) / 100 * self.price
        return price * number_to_be_bought

    def make_purchase(self, quantity):
        self.amount -= quantity

# name = input('name:')
# amount = int(input('Digit amount of items'))
# price = int(input('Digit price of items'))

name, amount, price = 'shoes', 200, 33

shoes = Product(name, amount, price)
# quantity = int(input('Digit amount of items to buy'))

q1 = 4
print(f'cost for {q1} {shoes.name} = {shoes.get_price(q1)}')
shoes.make_purchase(q1)
print(f'remaining stock: {shoes.amount}\n')

q2 = 12
print(f'cost for {q2} {shoes.name} = {shoes.get_price(q2)}')
shoes.make_purchase(q2)
print(f'remaining stock: {shoes.amount}\n')

q3 = 112
print(f'cost for {q3} {shoes.name} = {shoes.get_price(q3)}')
shoes.make_purchase(q3)
print(f'remaining stock: {shoes.amount}\n')
```

```
cost for 4 shoes = 132.0
remaining stock: 196

cost for 12 shoes = 356.4
remaining stock: 184

cost for 112 shoes = 2956.8
remaining stock: 72
```

```
#You need to create the foundations of an e-commerce engine for a B2C
(business-to-consumer) retailer. You need to have a class for a customer called
User, a class for items in inventory called Item, and a shopping cart class
called Cart. Items go in Carts, and Users can have multiple Carts. Also,
multiple items can go into Carts, including more than one of any single item.
```

In [1]:
```python
class User:
    def __init__(self, id, name):
        self.id = id
        self.name = name
    def display_user(self):
        print('ID', self.id, 'Name:', self.name)

class Item:
    def __init__(self, id, name, price, sold, available):
        self.id = id
        self.name = name
        self.price = price
        self.sold = sold
        self.available = available

class Cart:
    def __init__(self, user):
        self.user = user
        self.cart_items = []
    def insert_items(self, item, quantity):
        for i in range(quantity):
            if item.available == 0:
                print('Out of stock')
                break
            self.cart_items.append(item)
            item.sold += 1
            item.available -= 1
    def display_cart(self):
        print('This Cart belongs to', self.user.name, 'with ID', self.user.id)
        self.total = 0
        for i in self.cart_items:
            print('Item',i.name)
            self.total += i.price
        print('Total price = ', self.total)

#Creating a user
user1 = User(1, 'tejas')
user1.display_user()

#Creating items
apple = Item(1, 'apple', 100, 0, 10)
greencoconut = Item(2, 'green coconut', 150, 0, 1)
milk = Item(2, 'milk', 24, 0, 100)

#Creating two carts for user1
cart1 = Cart(user1)
cart2 = Cart(user1)

#Adding items to cart1
cart1.insert_items(apple, 2)
cart1.insert_items(milk, 3)

#Adding items to cart2
cart2.insert_items(greencoconut, 3)
cart2.insert_items(milk, 20)
```

```
#Displaying details of cart1 and cart2
cart1.display_cart()
cart2.display_cart()
```

```
ID 1 Name: tejas
Out of stock
This Cart belongs to tejas with ID 1
Item apple
Item apple
Item milk
Item milk
Item milk
Total price =  272
This Cart belongs to tejas with ID 1
Item green coconut
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Item milk
Total price =  630
```

```
In [ ]:
```