create a class called bank credit and debit method

In [9]:

```
class BankAcc:
    balance=0
    def __init__(self):

        print("Hello!!! Welcome to the Deposit & Withdrawal Machine")
class Credit():
    def deposit(self):
        amt=float(input("Enter the amount to be Deposited: "))
        balance += amt
        print("Amount Deposited:",amt)
class Debit():
    def withdraw(self):
        amt = float(input("Enter the amount to be Withdrawn: "))
        if (balance >= amt):
            balance -= amt
            print("You Withdrew:", amt)
        else:
            print("Insufficient balance")
class balance():
    def balance(self):
        print("\n Net Available Balance=",balance)
obj = BankAcc()
obj.intialamt()
obj1 = Credit()
obj1.deposit()
obj2 = Debit()
obj2.withdraw()
obj3=balance()
obj3.balance()
```

```
Hello!!! Welcome to the Deposit & Withdrawal Machine

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [9], in <cell line: 23>()
     21         print("\n Net Available Balance=",balance)
     22 obj = BankAcc()
---> 23 obj.intialamt()
     24 obj1 = Credit()
     25 obj1.deposit()

AttributeError: 'BankAcc' object has no attribute 'intialamt'
```

Constructor

In [ ]:

```
Constructor is generally used for instanting the object.
Constructor is a method that is called when an object is created. This method is define
```

In [17]:

```python
class sample:
    def __init__(self):
        print("Hello Welcome to Sret")
call=sample()
```

```
Hello Welcome to Sret
```

In [20]:

```python
class sample:
    def __init__(self,name,age):
        self.name = name
        self.age = age
    def display(self):
        print("From constructor",self.name,self.age)
s = sample("python",18)
s.display()
```

```
From constructor python 18
```

In [21]:

```python
class Account:
    def __init__(self):
        self.balance=0
        print("Your account is created.")
    def deposit(self):
        amount = int(input("Enter the amount to deposit: "))
        self.balance += amount
        print("Your new balance = ",self.balance)
    def withdraw(self):
        amount = int(input("Enter the amount to Withdraw: "))
        if (amount>=self.balance):
            print("Insufficient Balance")
        else:
            self.balance -= amount
            print("Your remaining balance = ",self.balance)
    def enquiry(self):
        print("Your balance",self.balance)
ac = Account()

```

```
Your account is created.
```

In [22]:

```python
ac.deposit()
```

```
Enter the amount to deposit: 10000
Your new balance =  10000
```

In [23]:

```
ac.withdraw()
```

```
Enter the amount to Withdraw: 5000
Your remaining balance =  5000
```

In [24]:

```
ac.enquiry()
```

```
Your balance 5000
```

Method Overloading

In [26]:

```
class Person:
    def Hello(seld,name=None):
        if name is not None:
            print("hello"+name)
        else:
            print("Hello")
obj = Person()
obj.Hello()
obj.Hello("User !")
```

```
Hello
helloUser !
```

if u hide any variable/attribute then outside of the class we cannot acess it

Data Hiding

In [32]:

```
#data hiding
class JustCounter:
    __secretCount = 0
    def count(self):
        self.__secretCount = self.__secretCount+1
        print(self.__secretCount)
c= JustCounter()
c.count()
c.count()
print("Outside the class")
print(c.__secretCount)
```

```
1
2
Outside the class

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [32], in <cell line: 11>()
      9 c.count()
     10 print("Outside the class")
---> 11 print(c.__secretCount)

AttributeError: 'JustCounter' object has no attribute '__secretCount'
```

In [ ]:

```

```