In [2]:
```
class Account:
    counter=0
    def __init__ (self,openingbal=0):
        Account.counter+=1
        self.id=Account.counter
        self.bal=openingbal
        self.numTrans=0  # new
        self.maxTrans=2 # new

    def deposit(self,amount):
        if amount>0: # condition for adding amount
            self.bal+=amount

    def withdraw(self,amount):
        if amount>0 and self.bal>=amount:
            self.bal -= amount

    def __str__(self):
        return f"Account {self.id} has Rs. {self.bal}."

class SavingsAccount(Account):

    pass
class CurrentAccount(Account):

    pass

sa1=SavingsAccount()
ca1=CurrentAccount()
```

In [4]:
```
print(sa1)
```

```
Account 1 has Rs. 0.
```

In [5]:
```
print(ca1)
```

```
Account 2 has Rs. 0.
```

In [6]:
```
type(sa1)
```

Out[6]:
```
__main__.SavingsAccount
```

In [7]:
```
type(ca1)
```

Out[7]:
```
__main__.CurrentAccount
```

In [8]:
```
a1=Account()
type(a1)
```

Out[8]:
```
__main__.Account
```

In [ ]:

In [27]:
```
class Account:
    counter=0
    def __init__ (self,openingbal=0):
        Account.counter+=1
        self.id=Account.counter
        self.bal=openingbal
        self.numTrans=0  # new
        self.maxTrans=100 # new

    def deposit(self,amount):
        if amount>0 and self.numTrans < self.maxTrans : # condition for adding amount
            self.bal+=amount
            self.numTrans+=1

    def withdraw(self,amount):
```

```
        if amount>0 and self.bal>=amount and self.numTrans < self.maxTrans:
            self.bal -= amount
            self.numTrans+=1

    def __str__(self):
        return f"Account {self.id} has Rs. {self.bal}."

class SavingsAccount(Account):
    def __init__(self):
        super().__init__()
        self.maxTrans=2

class CurrentAccount(Account):
    def __init__(self):
        super().__init__()
        self.maxTrans=4

sa1=SavingsAccount()
ca1=CurrentAccount()
```

In [28]:

```
print(sa1)
sa1.deposit(100)

print(sa1)
sa1.deposit(100)
print(sa1)
sa1.deposit(100)
print(sa1)
sa1.deposit(100)
print(sa1)
```

```
Account 1 has Rs. 0.
Account 1 has Rs. 100.
Account 1 has Rs. 200.
Account 1 has Rs. 200.
Account 1 has Rs. 200.
```

In [29]:

```
print(ca1)
ca1.deposit(100)

print(ca1)
ca1.deposit(100)
print(ca1)
ca1.deposit(100)
print(ca1)
ca1.deposit(100)
print(ca1)
```

```
Account 2 has Rs. 0.
Account 2 has Rs. 100.
Account 2 has Rs. 200.
Account 2 has Rs. 300.
Account 2 has Rs. 400.
```

In [ ]:

In [38]:

```
class Account:
    counter=0
    def __init__ (self,openingbal=0):
        Account.counter+=1
        self.id=Account.counter
        self.bal=openingbal
        self.numTrans=0  # new
        self.maxTrans=100 # new

    def deposit(self,amount):
        if amount>0 and self.numTrans < self.maxTrans : # condition for adding amount
            self.bal+=amount
            self.numTrans+=1

    def withdraw(self,amount):
        if amount>0 and self.bal>=amount and self.numTrans < self.maxTrans:
            self.bal -= amount
            self.numTrans+=1

    def __str__(self):
        return f"Account {self.id} has Rs. {self.bal}."

class SavingsAccount(Account):
```

```
    def __init__(self):
        super().__init__()
        self.maxTrans=15

class CurrentAccount(Account):
    def __init__(self):
        super().__init__()
        self.maxTrans=4

sa1=SavingsAccount()
ca1=CurrentAccount()
```

In [39]:
```
print(sa1)
sa1.deposit(100)
print(sa1)
```

```
Account 1 has Rs. 0.
Account 1 has Rs. 100.
```

In [40]:
```
print(sa1.bal)
```

```
100
```

In [41]:
```
sa1.bal=10000000
```

In [42]:
```
print(sa1)
```

```
Account 1 has Rs. 10000000.
```

In [ ]:

In [54]:
```
class Account:

    counter = 0

    def __init__(self, openingBal=0):
        Account.counter += 1
        self.id = Account.counter
        self.__bal = openingBal
        self.numTrans = 0
        self.maxTrans = 2

    def deposit(self, amount):
        if amount >= 0 and self.numTrans < self.maxTrans:
            self.__bal += amount
            self.numTrans += 1

    def withdraw(self, amount):
        if amount >= 0 and self.__bal >= amount and self.numTrans < self.maxTrans:
            self.__bal -= amount
            self.numTrans += 1

    def getInterest(self):
        pass


    def __str__(self):
        return f"Acc {self.id} has {self.__bal}"  # new --> self.__bal



class SavingsAccount(Account):
    def __init__(self):
        super().__init__()

    def getInterest(self):  # new - Interest calculation for Savings Account
        interest = self.__bal*0.07
        print(f"Interest on Savings Account {self.id} is {interest}")


class CurrentAccount(Account):
    def __init__(self):
        super().__init__()
        self.maxTrans = 3
```

```
    def getInterest(self):  # new - Interest calculation for Current Account
        interest = (self.__bal*0.05)/self.numTrans
        print(f"Interest on current Account {self.id} is {interest}")


sa1 = SavingsAccount()
ca1 = CurrentAccount()

print(sa1)
sa1.deposit(100)
sa1.withdraw(50)
print(sa1)
sa1.getInterest()


print(ca1)
ca1.deposit(100)
ca1.deposit(100)
ca1.deposit(100)
print(ca1)
ca1.getInterest()
```

```
Acc 1 has 0
Acc 1 has 50
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/var/folders/hd/9z4dczb56dj54lb7q8w7s4zw0000gn/T/ipykernel_2824/4256074114.py in <module>
     55 sa1.withdraw(50)
     56 print(sa1)
---> 57 sa1.getInterest()
     58
     59

/var/folders/hd/9z4dczb56dj54lb7q8w7s4zw0000gn/T/ipykernel_2824/4256074114.py in getInterest(self)
     34
     35     def getInterest(self):  # new - Interest calculation for Savings Account
---> 36         interest = self.__bal*0.07
     37         print(f"Interest on Savings Account {self.id} is {interest}")
     38

AttributeError: 'SavingsAccount' object has no attribute '_SavingsAccount__bal'
```

In [44]:
```
print(sa1)
```

```
Account 1 has Rs. 0.
```

In [52]:
```
print(sa1.id)
```

```
1
```

In [46]:
```
print(sa1.__bal)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/var/folders/hd/9z4dczb56dj54lb7q8w7s4zw0000gn/T/ipykernel_2824/1165396804.py in <module>
----> 1 print(sa1.__bal)

AttributeError: 'SavingsAccount' object has no attribute '__bal'
```

In [47]:
```
sa1.__bal=100000000
```

In [48]:
```
print(sa1)
```

```
Account 1 has Rs. 0.
```

In [49]:
```
print(sa1.__bal)
```

```
100000000
```

In [50]:
```
print(sa1)
```

```
Account 1 has Rs. 0.
```

In [51]:
```
sa1.__money=10000
```

In [ ]:
```

```

In [ ]:
```

```

In [55]:
```
class Account:

    counter = 0

    def __init__(self, opening__bal=0):
        Account.counter += 1
        self.id = Account.counter
        self.__bal = opening__bal
        self.numTrans = 0
        self.maxTrans = 2

    def deposit(self, amount):
        if amount >= 0 and self.numTrans < self.maxTrans:
            self.__bal += amount
            self.numTrans += 1

    def withdraw(self, amount):
        if amount >= 0 and self.__bal >= amount and self.numTrans < self.maxTrans:
            self.__bal -= amount
            self.numTrans += 1

    def getInterest(self): # new
        pass


    def __str__(self):
        return f"Acc {self.id} has {self.__bal}"  # new --> self.____bal



class SavingsAccount(Account):
    def __init__(self):
        super().__init__()

    def getInterest(self):  # new - Interest calculation for Savings Account
        interest = self._Account__bal*0.07
        print(f"Interest on Account {self.id} is {interest}")


class CurrentAccount(Account):
    def __init__(self):
        super().__init__()
        self.maxTrans = 3

    def getInterest(self):  # new - Interest calculation for Current Account
        interest = (self._Account__bal*0.05)/self.numTrans
        print(f"Interest on Account {self.id} is {interest}")


sa1 = SavingsAccount()
ca1 = CurrentAccount()

print(sa1)
sa1.deposit(100)
sa1.withdraw(50)
print(sa1)
sa1.getInterest()


print(ca1)
ca1.deposit(100)
ca1.deposit(100)
ca1.deposit(100)
print(ca1)


ca1.getInterest()
```

```
Acc 1 has 0
Acc 1 has 50
Interest on Account 1 is 3.5000000000000004
Acc 2 has 0
Acc 2 has 300
Interest on Account 2 is 5.0
```

In [ ]:

In [ ]:

In [60]:

```python
class Account:

    counter = 0

    def __init__(self, opening__bal=0):
        Account.counter += 1
        self.id = Account.counter
        self.__bal = opening__bal
        self.numTrans = 0
        self.maxTrans = 2

    def deposit(self, amount):
        if amount >= 0 and self.numTrans < self.maxTrans:
            self.__bal += amount
            self.numTrans += 1

    def withdraw(self, amount):
        if amount >= 0 and self.__bal >= amount and self.numTrans < self.maxTrans:
            self.__bal -= amount
            self.numTrans += 1

    def getInterest(self):
        pass # new



    def __str__(self):
        return f"Acc {self.id} has {self.__bal}"  # new --> self.____bal

    def __repr__(self):
        return f"{id}"

class SavingsAccount(Account):
    def __init__(self):
        super().__init__()

    def getInterest(self):  # new - Interest calculation for Savings Account
        interest = self._Account__bal*0.07
        print(f"Interest on Account {self.id} is {interest}")


class CurrentAccount(Account):
    def __init__(self):
        super().__init__()
        self.maxTrans = 3

    def getInterest(self):  # new - Interest calculation for Current Account
        interest = (self.newbalance*0.05)/self.numTrans
        print(f"Interest on Account {self.id} is {interest}")


sa1 = SavingsAccount()
ca1 = CurrentAccount()

print(sa1)
sa1.deposit(100)
sa1.withdraw(50)
print(sa1)
sa1.getInterest()


print(ca1)
ca1.deposit(100)
ca1.deposit(100)
ca1.deposit(100)
print(ca1)


ca1.getInterest()
```

```
Acc 1 has 0
Acc 1 has 50
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/var/folders/hd/9z4dczb56dj54lb7q8w7s4zw0000gn/T/ipykernel_2824/3129768368.py in <module>
     57 sa1.withdraw(50)
     58 print(sa1)
---> 59 sa1.getInterest()
     60
     61

/var/folders/hd/9z4dczb56dj54lb7q8w7s4zw0000gn/T/ipykernel_2824/3129768368.py in getInterest(self)
     36
     37     def getInterest(self):  # new - Interest calculation for Savings Account
---> 38         interest = self.newbalance*0.07
     39         print(f"Interest on Account {self.id} is {interest}")
     40

AttributeError: 'SavingsAccount' object has no attribute 'newbalance'
```

In [57]:
```
print(sa1._Account__bal)
```

```
50
```

In [58]:
```
sa1._Account__bal=10000000
```

In [59]:
```
print(sa1)
```

```
Acc 1 has 10000000
```

In [ ]:
```

```

In [61]:
```
import math
```

In [62]:
```
type(math)
```

Out[62]:
```
module
```

In [63]:
```
print(math)
```

```
<module 'math' from '/Users/nikhilsanghi/opt/anaconda3/lib/python3.9/lib-dynload/math.cpython-39-darwin.so'>
```

In [64]:
```
import numpy
```

In [65]:
```
print(numpy)
```

```
<module 'numpy' from '/Users/nikhilsanghi/opt/anaconda3/lib/python3.9/site-packages/numpy/__init__.py'>
```

In [68]:
```
# help(numpy)
```

In [70]:
```
a=100000000000000
print(math.sqrt(a))
```

```
10000000.0
```

In [71]:
```
math.factorial(5)
```

Out[71]:
```
120
```

In [72]:
```
math.pow(3,2)
```

Out[72]:
```
9.0
```

In [73]:
```
math.pow(10,10)
```

Out[73]:
```
10000000000.0
```

In [75]:
```
print(math.floor(2.66))
print(math.floor(-1.66))
```

```
2
-2
```

In [76]:
```
print(math.ceil(2.66))
print(math.ceil(-1.66))
```

```
3
-1
```

In [ ]:

In [ ]:

In [38]:
```
import random
```

In [53]:
```
random.seed(50)
```

In [57]:
```
random.randint(0,100)
```

Out[57]:
```
81
```

In [59]:
```
import math
help(math.sqrt)
```

```
Help on built-in function sqrt in module math:

sqrt(x, /)
    Return the square root of x.
```

In [62]:
```
print(round(2.5))
print(round(3.5))
print(round(4.5))
print(round(5.5))
```

```
2
4
4
6
```

In [ ]:

In [ ]:

In [ ]: