Employee_Code

```
class Employee:
    count=0
    def __init__(self,Name,Id,sal):    #self reprenets the current object.
        self.Name=Name
        self.Id=Id
        self.sal=sal
        Employee.count+=1
    def disp(self):
        print('Name=',self.Name,'Id=',self.Id,'salary=',self.sal)
E1=Employee("Sujit",101,7000)
E2=Employee("sayan",102,80000)
E1.disp()
E2.disp()
print(Employee.count)
```

1. WAP to print all odd non prime numbers from 1 to 50.

Ans:-

```
print("Odd Non-Prime Numbers from 1 to 50:")

for num in range(1, 51):
    if num % 2 != 0:  # odd
        # check prime
        if num <= 1:
            print(num, end=" ")
        else:
            is_prime = True
            for i in range(2, num):
                if num % i == 0:
                    is_prime = False
                    break
            if not is_prime:
                print(num, end=" ")
```

2. Write a program to find out whether a number is a palindrom number or not.

Ans:-

```
num = int(input("Enter a number: "))
rev = 0
n = num
while n > 0:
    rev = rev * 10 + (n % 10)
    n //= 10
if num == rev:
    print("Palindrome Number")
else:
    print("Not Palindrome")
```

3. Write a program to repeatedly perform sum of digits of a number untill the result is converted to a single digit.

Ans:-

```
num = int(input("Enter a number: "))

while num > 9:  # repeat until single digit
    s = 0
    while num > 0:
        s += num % 10
        num //= 10
    num = s

print("Single digit sum =", num)
```

4. WAP to find out whether a number is a perfect number or not.

Ans:-

```
num = int(input("Enter a number: "))
sum_div = 0

for i in range(1, num):
    if num % i == 0:
        sum_div += i
```

```
if sum_div == num:
    print(num, "is a Perfect Number")
else:
    print(num, "is NOT a Perfect Number")
```

1. Student Class – Create 2 Students & Display Details

```
class Student:
    def __init__(self, name, roll, marks):
        self.name = name
        self.roll = roll
        self.marks = marks

    def display(self):
        print("Name:", self.name, " Roll:", self.roll, " Marks:", self.marks)


# Create 2 student objects
s1 = Student("Rahul", 101, 85)
s2 = Student("Sumi", 102, 90)

# Display details
s1.display()
s2.display()
```

2. Complex Class – Print as a + bi & Count Objects

```
class Complex:
    count = 0

    def __init__(self, real, imag):
        self.real = real
        self.imag = imag
        Complex.count += 1

    def display(self):
        print(f"{self.real} + {self.imag}i")


# Creating multiple objects
c1 = Complex(3, 4)
c2 = Complex(5, 6)
c3 = Complex(1, -2)

c1.display()
c2.display()
c3.display()

print("Total Complex Numbers Created =", Complex.count)
```

3. Account Class – Deposit, Withdraw, Check Balance + Count Accounts

```
class Account:
    count = 0

    def __init__(self, name, accno, balance):
        self.name = name
        self.accno = accno
        self.balance = balance
        Account.count += 1

    def deposit(self, amt):
        self.balance += amt
        print("Deposited:", amt)

    def withdraw(self, amt):
        if amt <= self.balance:
            self.balance -= amt
            print("Withdrawn:", amt)
        else:
            print("Insufficient Balance!")

    def check_balance(self):
        print("Current Balance:", self.balance)
```

```
# Create multiple accounts
a1 = Account("Sujit", 1001, 5000)
a2 = Account("Riya", 1002, 8000)

a1.deposit(1000)
a1.withdraw(3000)
a1.check_balance()

a2.withdraw(9000)
a2.check_balance()

print("Total Accounts Created =", Account.count)
```

4. Car Class – Create Objects & Count Cars

```
class Car:
    count = 0

    def __init__(self, car_id, car_name):
        self.car_id = car_id
        self.car_name = car_name
        Car.count += 1

    def display(self):
        print("Car ID:", self.car_id, " Car Name:", self.car_name)


# Create 2 car objects
car1 = Car(1, "BMW")
car2 = Car(2, "Audi")

car1.display()
car2.display()

print("Total Cars Created =", Car.count)
```