1)

```
def prime(n,i):
    if n%i==0:
        if n!=i:
            return False
        else:
            return prime(n,i+1)
    return True
def cir(a):
    s=str(a)
    a=int(s[1:]+s[0])
    return a
a=int(input("Enter No:"))
print(a)
if prime(a,i=2):
    x=cir(a)
    print(x)
    if a!=x:
        prime(x,i=2)
    print("Circular Prime")

else:
    print("Not Circular Prime No")
```

The Fact. of 5 in 120

5 * 4 * 3 * factorial(2)

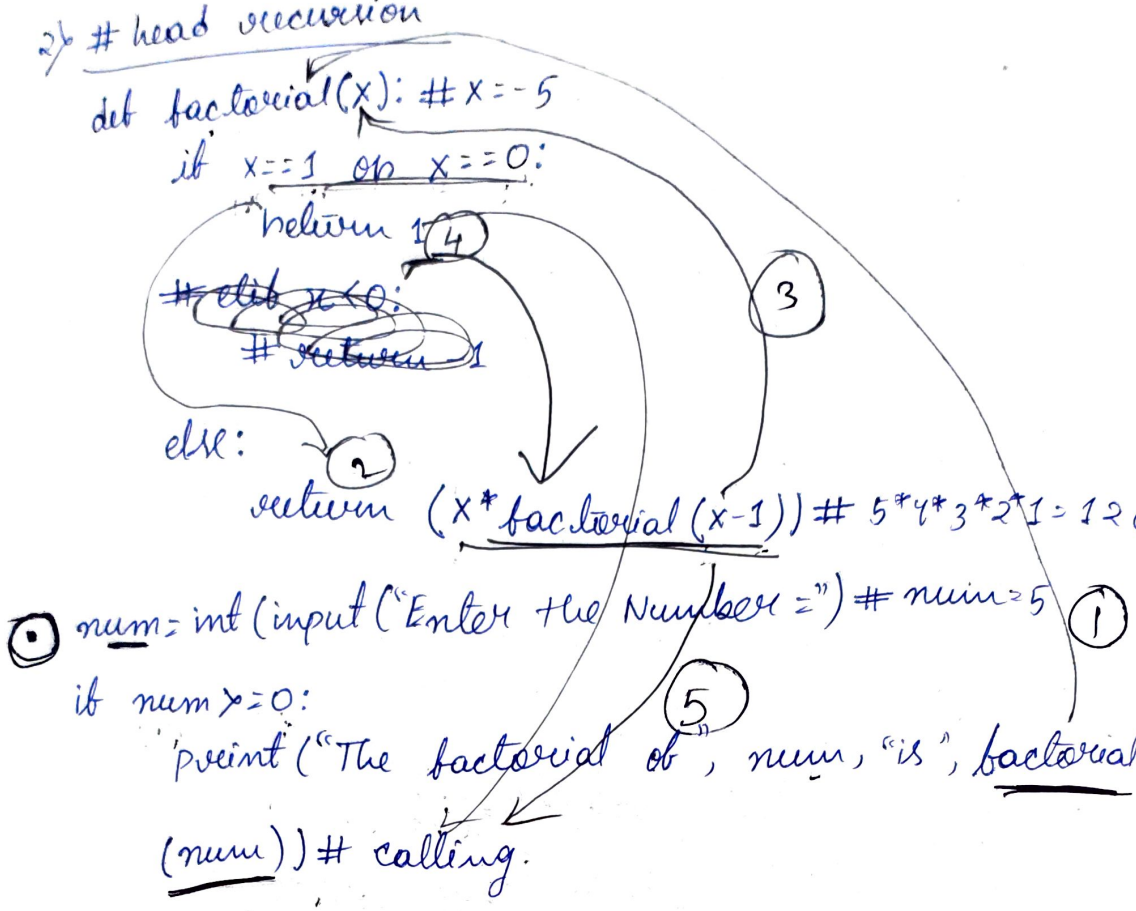
5 * 4 * 3 * 2 * factorial(1)

5 * 4 * 3 * 2 * 1 = 120

2) # head recursion

```
def factorial(x): # x = 5
    if x==1 or x==0:
        return 1
    # elif x<0:
    #     # return -1
    else:
        return (x*factorial(x-1)) # 5*4*3*2*1 = 120

num = int(input("Enter the Number =")) # num = 5
if num >= 0:
    print("The factorial of", num, "is", factorial(num)) # calling.
else:
    # x = factorial(num) # x = -1
    print("-ve no. fact. not possible")
```



Enter the Number = 0/1

num = 0 X = 0

The factorial of 0 is 1

Enter the no. = 5

num = 5 X = 5 4 3

X * factorial(X-1)

5 * factorial(4)

5 * 4 * factorial(3)

3) def recursive_factorial(n): # n = 1

if n == 1:

return n (1)

else: (2)

return n * recursive_factorial (n-1) (3) (4)

~~recursive_factorial(n-1) # 120~~

num = int(input("Enter the No. = "))

if num < 0:

print("Invalid input! Please enter a positive number")

elif num == 0 or num == 1:

print("Factorial ~~of number~~ 0 is 1") (5) (1)

else:

print("Factorial of number", num, "=", recursive_factorial(num)) # num = 5

Enter the No. = -5.

num = -5

5 * n-1(4)

5 * 4 * n-1(3)

5 * 4 * 3 * n-1(2)

n = 5 4 3 2

5 * 4 * 3 * 2 * n-1(1)

= 5 * 4 * 3 * 2 * 1

= 120

n = 0/1

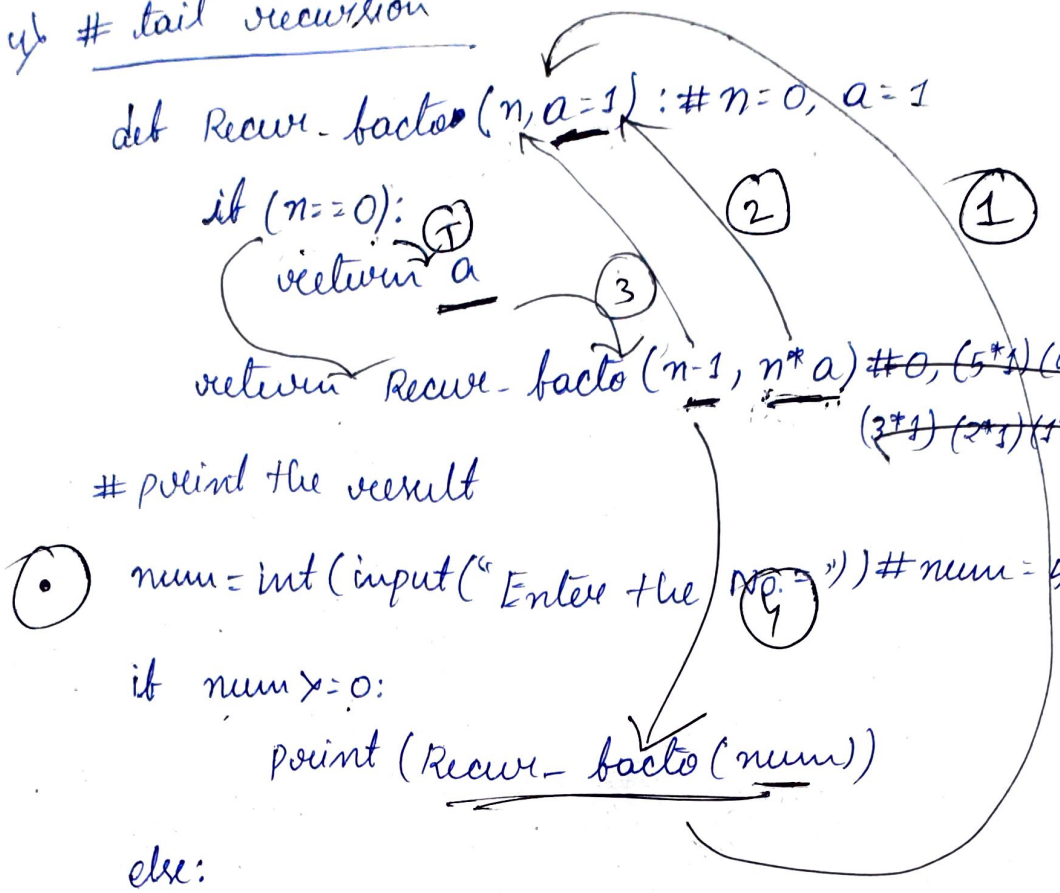
num = 1

n = 2

num = 5

tail recursion

```
def Recur_facto(n, a=1): # n=0, a=1
    if (n==0):
        return a
    return Recur_facto(n-1, n*a) # 0, (5*1) (4*1) (3*1) (2*1) (1*1)
# print the result
num = int(input("Enter the NO. = ")) # num = 5
if num >= 0:
    print(Recur_facto(num))
else:
    print("-ve no. factorial can't be possible")
```



Enter the NO. = 4

num = 4

n = 4, a = 1 4 12 24

recur_facto (3, 4)

(2, 12)

(1, 24)

(0, 24)

Tail recursion will take less memory

Head " more memory

fibonacci recursion

```
def fibo(n): #n=2
    if n==0:
        return 0
    elif n==1:
        return 1
    else:
        return fibo(n-1)+fibo(n-2) #1+1+3

n=int(input("Enter the Range="))  #n=5
for i in range(n):
    print(fibo(i), end=" ")
```

Sum = Sum + fact(rem)

Sum = 0 + 120

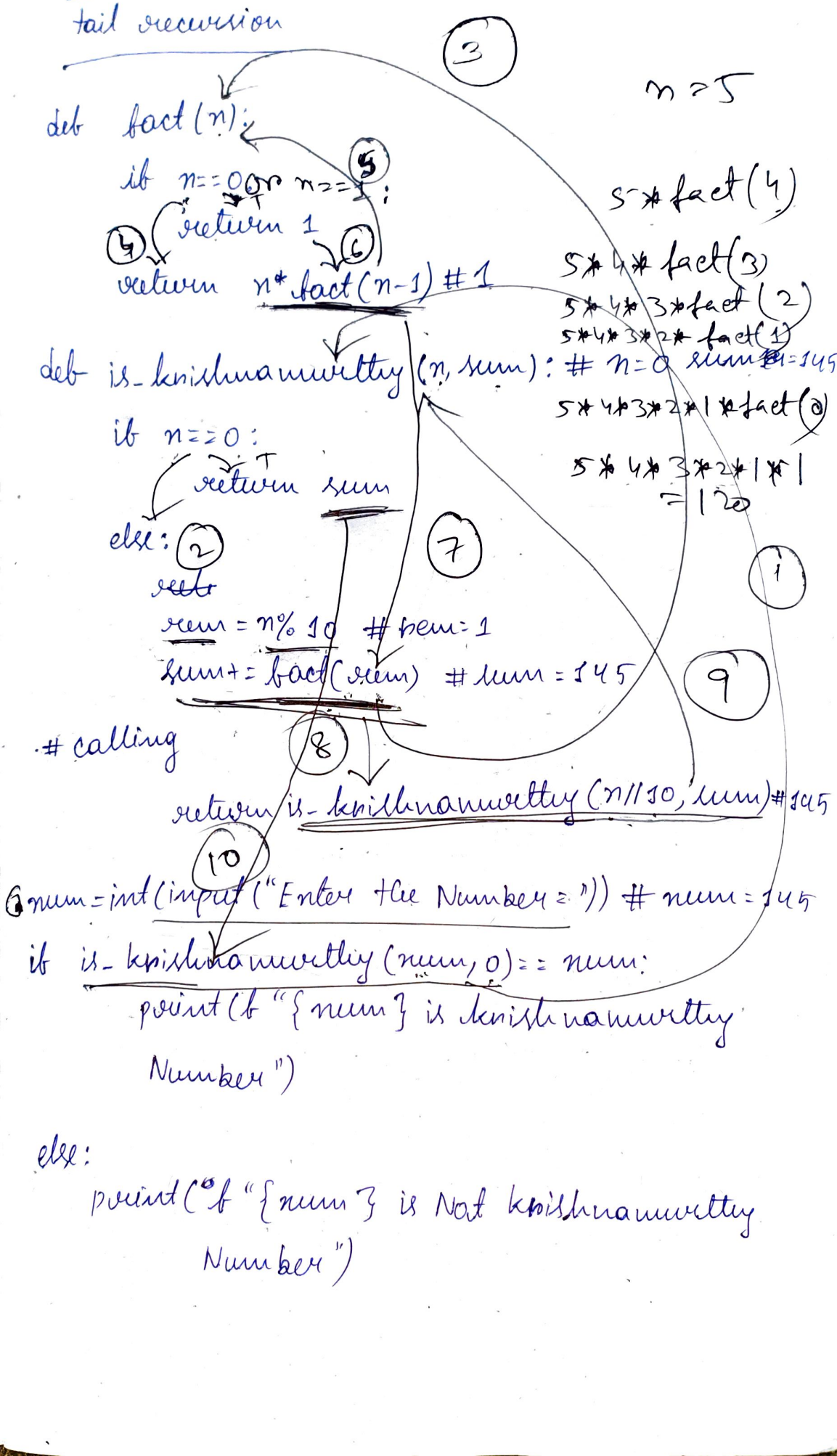
= 120 + 24

= 144 + 1

145

Krishnamurthy 145 → 5! = 120 4! = 24 1! = 1

tail recursion



Enter the number = 145

num = 145

n = 145 0 s = 0 120 144 145

rem = 5 4 1

n = 145

↳ 5! = 120

4! = 24

1! = 1

+

145

n = 40585

5! = 120

8! = 40320

5! = 120

0! = 1

4! = 24

n = 125

5! = 120

2! = 2

2! = 2

1! = 1

123 ✗

n = 163

3 + sod(16)

3 + 6 + sod(1)

3 + 6 + 1 + sod(0)

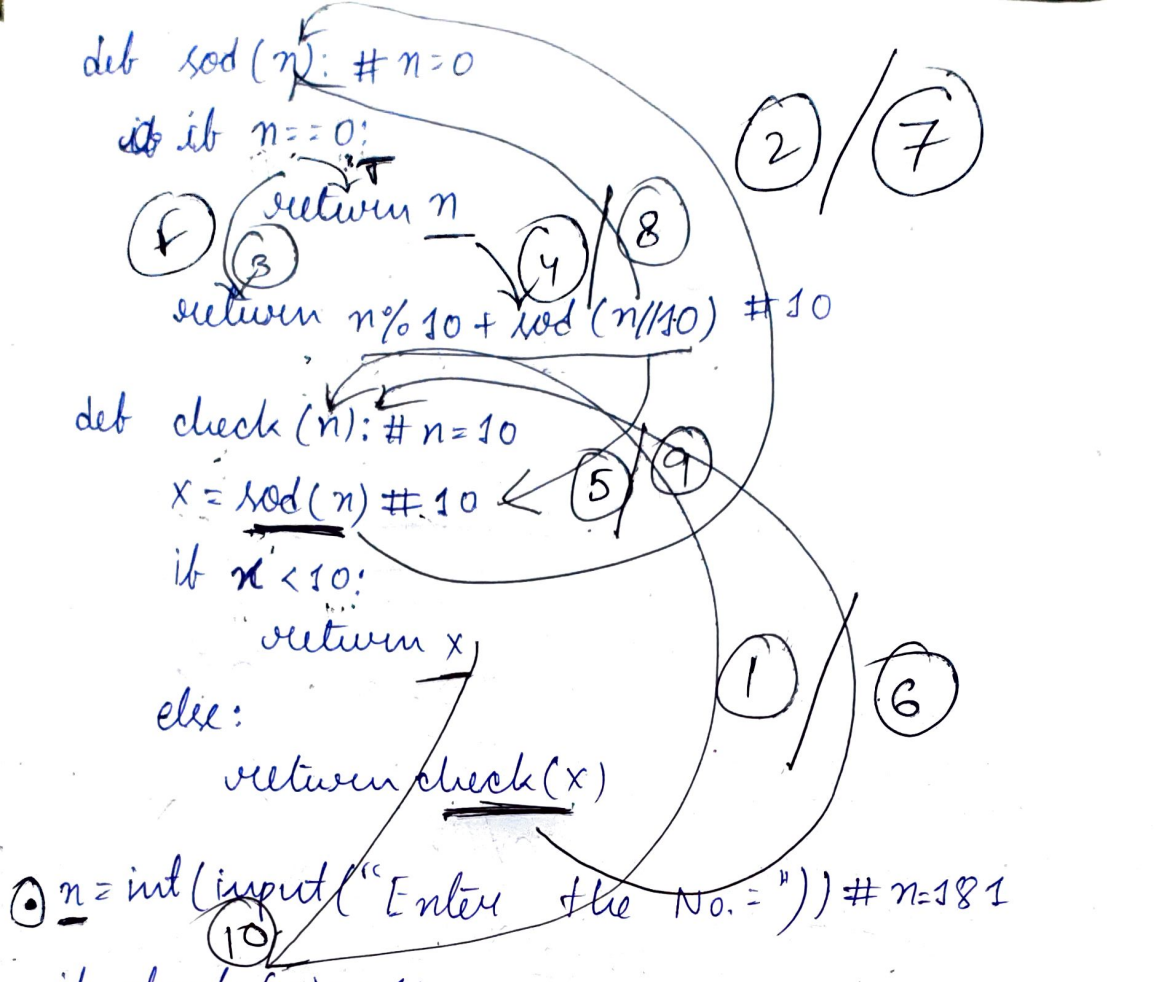
3 + 6 + 1 + 0

= 10

0 10

n = 163

X = 10



Magic No. = 19, 28, 37, 46,
55, 64, 73, 82, 91,
118, 127, 181, 172,
100, 163

↳ s.o.d = 10

n % 10 + Sod(n//10) 6+1+0 ↳ 0+1

0 + Sod(1) → 0+1+Sod(0)

Sum of digit of a no.

```
def sod(n): # n=0
    if n==0:
        return n
    return n%10+sod(n//10) # 4+3+2+1+0=10
n=int(input("Enter the No. = ")) # n=1234
print("Sum of Digit =", sod(n)) # calling
```

Enter the No. = 124

n = 124

n%10 + sod(n//10)

4 + sod(12)

4+2 + sod(12//10)

4+2+sod(1)

4+2+1 + sod(1//10)

4+2+1 + sod(0) → 4+2+1+0 = 7

n = 124

10 | 124 | 12
10
24
20
4

Wap to check two numbers are twin prime or not

```
def is_prime(n,i): # 7,0,7
    if i==1:
        return True
    else:
        if n%i==0:
            return False
        return is_prime(n,i-1) # 2
num=int(input("Enter the 1st Number: ")) #5
```

```
num2 = int(input("Enter The 2nd Number:"))#7
if is_prime(num, num-1) and
   is_prime(num2, num2-1) and
   abs(num - num2) == 2:
      print(f"{num} and {num2} are
         Twin Prime")

else:
      print(f"{num} and {num2} are Not
         Twin Prime")
```

---

n = 127

~~n = 0~~

7 + sod(12)

7 + 2 + sod(1)

7 + 2 + 1 + sod(0)

7 + 2 + 1 + 0 = 10

sum of digit = 10