#factorial

fact = 1

for i in range(1, 9):

    fact \*= i

print("The factorial of 8 is", fact)

#fabonacci

a, b = 0, 1

for \_ in range(5):

    print(a)

    a, b = b, a + b

#armstrong

def armstrong():

    num=int(input("enter a number "))

    arms=str(num)

    result=0

    for i in arms:

        result=result+int(i)\*\*len(arms)

    print(result)

    if result==num:print("yes it is an armstrong number")

    else:print("it is not an armstrong number")

armstrong()

#reverse an integer

num = 12345

reversed\_num = int(str(num)[::-1])

print("Reversed integer:", reversed\_num)

#recurssion

def factorial(n):

    if n == 0:

        return 1

    else:

        return n \* factorial(n-1)

num = 8

print("The factorial of", num, "is", factorial(num))

#recurssion

def factorial(n):

    if n == 0:

        return 1

    else:

        return n \* factorial(n-1)

num = 8

print("The factorial of", num, "is", factorial(num))

def recur\_factorial(n):

   if n == 1:

       return n

   else:

       return n\*recur\_factorial(n-1)

n=5

two pyramids from left side

n = 11

for i in range(n):

    if i < 5:

        print("\* " \* (i+1))

    else:

        print("\* " \* (n-i))

def diamond\_pattern(n):

    for i in range(1, n + 1):

        for j in range(1, n - i + 1):

            print(" ", end="")

        for k in range(1, 2 \* i):

            if k % 2 == 1:

                print("\*", end="")

            else:

                print(" ", end="")

        print("")

    for i in range(n - 1, 0, -1):

        for j in range(1, n - i + 1):

            print(" ", end="")

        for k in range(1, 2 \* i):

            if k % 2 == 1:

                print("\*", end="")

            else:

                print(" ", end="")

        print("")

# Example: Print a diamond pattern with 5 rows

n = 5

diamond\_pattern(n)

n = 5

for i in range(n):

    for j in range(n):

        if i == 0 or i == n - 1 or j == 0 or j == n - 1:

            print("\*", end="")

        else:

            print(" ", end="")

    print()

def converted\_hello\_pyramid():

    word = "HELLO"

    for i in range(len(word)\*2 - 1):

        if i < len(word):

            print(" " \* (len(word) - i - 1), end="")

            print(\*[word[j] for j in range(i+1)])

        else:

            print(" " \* (i - len(word)), end="")

            print(\*[word[j] for j in range(2\*len(word)-1 - i)])

converted\_hello\_pyramid()

H

H E

H E L

H E L L

H E L L O

H E L L

H E L

H E

H

username=['lionel','tanuja']

password=['123','124']

name=input('enter your name:')

for a,b in enumerate(username):

    print(a,b)

    if b==name:

        password=input('enter your password:')

        if password==password[a]:

             print('welcome')

i=50

attempts=0

while True:

   print('Enter your name?')

   name=input()

   if name !='lionel':

       continue

   print('Hello,lionel what is the password?')

   password=input()

   if password=='simmba':

      print('Access granted')

      break

   elif password!='simmba' and attempts < 2:

    print('one attempt is over. still two remaining')

    attempts +=1

   elif password!='simmba'and attempts < 3:

    print('two attempt is over. still one remaining')

    attempts+=1

   else:

        print('ur attempts are over. account is blocked')

        break

 #ATM

name=['lionel','tanuja','kabber','lion']

password=[123,1234,12345,1233]

balance=[1000000,2000000,500000,1000]

def withdraw(current):

    amount=int(input("Enter the amount : "))

    if amount<=balance[current]:

        balance[current]-=amount

        print("successfully")

    else:print("Insufficient funds")

def c\_balance(current):

    print("balance is ",balance[current])

def default(current):

    print("enter crt option")

u\_name=input('Enter your name : ')

for i in range(len(name)):

 if u\_name==name[i]:

    u\_password=int(input('Enter your password : '))

    if u\_password == password[i]:

        while True:

            print('1.withdraw\n2.balance')

            option=int(input('Enter your option : '))

            if option==0:break

            data={1:withdraw,2:c\_balance}

            result=data.get(option,default)

            result(i)

def get\_multiple\_occurrences\_indices(lst, user):

    indices = []

    for i, item in enumerate(lst):

        if item == user:

            indices.append(i)

    return indices

L = [1, 0, 3, 6, 0, 4, 5, 0, 6]

user = 0

indices = get\_multiple\_occurrences\_indices(L, user)

print(f"The indices of user {user} in the list are: {indices}")