OPERATORS

Arithmetic operators

+:

* Used with minimum of 2 numbers to perform 🡺addition
* Used with a single number to represent a positive number

print(5 + 3 + 2) #addition

print(88.5 + 99.2)  #addition

print(5 + 8 + 9 + 4)#addition

print(+8)#positive

print(+88.5)#positive number

-:

print(67 - 32 - 7 -2)#subtration

print(45 - 67)#subtration

print(1000.8 - 86.6 - 77.9)#substration

print(-66.6)#negative number

print(-5)#negative number

* Used with minimum of 2 numbers to perform 🡺 subtraction
* Used with a single number to represent a negative number

\*:

print(3 \* 7 \* 6 \* 9 \* 77)#multiplication

print(4 \* 99)#multiplication

print(88.5 \* 99.2)#multiplication

print(\*8)#error

* Used with compulsorily with minimum of 2 numbers to perform 🡺 multiplication

\*\*(exponentiation):

print(4\*\*3)# 2\*2\*2 = 8

print((-6)\*\*2)

print(7.2\*\*2)

* Return exponent value

Division operation

1. Floor division //

Return non decimal output

1. Float division /

Return decimal output

1. Modulous %

Return quotient

#floor div

print(20 // 2)

print(22 // 7)

#float div

print(200 / 2)

print(22 / 7)

#modulus for remainder in return

print(10 % 2)

print(22 % 7)

print(3 % 77)

print(8 % 88)

a = 100

b = 100

c = 100

d = a

print(a == b == c == d)

#same shared memory

print(a is d)

print(b is c)

print(a is c)

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

#same variable different memory

x = 10

print(x)

print(id(x))

x = 20

print(x)

print(id(x))

x = 30

print(x)

print(id(x))

a = 100

b = 100

c = 100

d = a

print(a == b == c == d)

#same shared memory

print(a is d)

print(b is c)

print(a is c)

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

#same variable different memory

x = 10

print(x)

print(id(x))

x = 20

print(x)

print(id(x))

x = 30

print(x)

print(id(x))

s1 ="abbc123"

print("a" in s1)

print("bbc" in s1)

print("12 in s1")

#print(23 in s1)

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

a = 10

print(1 in a) #error It id single valued data #error

 a = 10

print("1"in a) #error

a = ""

print("ab" in a)

**Control flow statements**

The statements that control the execution flow of a program

3 types:

1, Conditional statem:

* Simple if
* If – else
* elif / if – elif / elif ladder

Note: Combination of these above statm’s are “nested if conditions”

2. Looping statm’s:

* For loop

i. for with range() incre, decre

ii. for without range()

* While loop

i. incre

ii. decre

3. Jumping stmts:

* Break, continue, return

Conditional stamts:

Simple if :

age = int(input("enter your age"))

if (age >= 18):

    print("you can vote")

print("you are a citizen")

If – else:

Syx:

If bool\_cond

Logic

Else:

Alternate

Remain stmts

**Note: Divisibility or factor the first operator used is mod %**

num = int(input("enter the number"))

if (num % 2 == 0):

    print(f"{num} is even")

else:

    print(f"{num} is odd")

print("Prog exec")

**elif:**

It will be used when in a given scenario there will be multiple condi to be checked and for each condi there will be specific logic

Syntax:

If bool\_cond1:

#logic1

Elif bool\_cond2:

#logic2

Elif bool\_cond3:

#logic3

:

:

Else:

#alternate logic

Program to print statements accordingly

If the num is posit but even : posit even num

If num is neg but even : neg even num

If the num is posit but odd: posit odd num

If num is neg but odd: neg odd num

**Programming numerical**

1. write a program to check even or odd

n = int(input("enter num"))

if (n % 2 == 0):

    print(f"{n} is even")

else:

    print(f"{n} is odd")

2. write a program to check even or odd using customized function

**Flag: standard variable name to hold a Boolean value**

def evenodd(n):

    if (n % 2 == 0):

        return True #even

    else:

        return False #odd

n = int(input("enter num"))

flag = evenodd(10)

if flag:

    print(f"{n} is even")

else:

    print(f"{n} is odd")

Optimized

def evenodd(n):

    return (n % 2 == 0)

n = int(input("enter num"))

flag = evenodd(10)#Flag is a standard variable name to hold a Boolean value

if flag:

    print(f"{n} is even")

else:

    print(f"{n} is odd")

Tracing

n = 5#false

p(n%2 == 0)

1 == 0 #false

------------------

n = 10

print(n%2 == 0)

0 == 0 #false

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

N = 5

If 5 %2 == 0: false

Else: result false

N = 10

If 10 % 2 == 0#true

Res true

3. Write a program to print all even numbers of user defined range

def evenodd(n):

    return n % 2 == 0

sr = int(input("enter start range"))

er = int(input("Enter end value"))

for i in range(sr,er+1):

    flag = evenodd(i)

    if flag:

        print(i, end=" ")

print("\n Odd numbers:")

for i in range(sr,er+1):

    flag = evenodd(i)

    if not flag:

        print(i, end=" ")

or

def evenodd(n):

    return n % 2 == 0

sr = int(input("enter start range"))

er = int(input("Enter end value"))

if (sr > er):

    print("invalid input")

else:

    for i in range(sr,er+1):

        flag = evenodd(i)

        if flag:

            print(i, end=" ")

    print("\n Odd numbers:")

    for i in range(sr,er+1):

        flag = evenodd(i)

        if not flag:

            print(i, end=" ")

4. Write a function to print first N even numbers

def evenodd(n):

    return n % 2 == 0

count = int(input("enter the count of even num to be printed"))

series = 1

while count > 0:

    flag = evenodd(series)

    if flag:

        print(series, end=" ")

        count -= 1

    series += 1

5. Write a function to print first N odd numbers

Leap Year:

6. write a program to find leap year or not

year = int(input("enter year:"))

if (year % 100!= 0 and year % 4 == 0) or (year % 400 == 0):

        print(f" {year}Leap Year")

else:

       print(f"{year} not a leap year")

7. write a program to find leap year or not using optimized function

def leapyear(year):

        if (year % 100!= 0 and year % 4 == 0) or (year % 400 == 0):

                return True

        else:

                return False

year = int(input("enter year:"))

flag = leapyear(year)

if flag:

       print(f" {year}Leap Year")

else:

       print(f"{year} not a leap year")

or

def leapyear(year):

        return (year % 100!= 0 and year % 4 == 0) or (year % 400 == 0)

year = int(input("enter year:"))

flag = leapyear(year)

if flag:

       print(f" {year}Leap Year")

else:

       print(f"{year} not a leap year")

8. Write a program to print all the leap years of a user defined range

def leapyear(year):

        return (year % 100!= 0 and year % 4 == 0) or (year % 400 == 0)

sr = int(input("enter start value"))

er = int(input("enter end value"))

if (sr > er):

        print("Invalid input")

else:

        print("leap years")

        for i in range(sr,er+1):

                flag = leapyear(i)

                if flag:

                        print(i)

9. Write a program to print leap and non-leap year separate

def leapyear(year):

        return (year % 100!= 0 and year % 4 == 0) or (year % 400 == 0)

sr = int(input("enter start value"))

er = int(input("enter end value"))

if (sr > er):

        print("Invalid input")

else:

        print("leap years")

        for i in range(sr,er+1):

                flag = leapyear(i)

                if flag:

                        print(i)

        print("\nNot leap year")

        for i in range(sr,er+1):

                flag = leapyear(i)

                if not flag:

                        print(i)

10. Write a program to count the num of digits in given num:

n = int(input("enter num:"))

count = 0

while (n > 0):

    n = n // 10

    count = count + 1

print(count)

for negative

n = int(input("enter num:"))

if (n < 0):

        n = n \* -1

count = 0

while (n > 0):

    n = n // 10

    count = count + 1

print(count)

11. Write a program to count the num of digits in given num using custumized functions

def countdig(n):

      if (n < 0):

          n = n \* -1

      while (n > 0):

          n = n // 10

          count = count + 1

      return count

n = int(input("enter num:"))

count = 0

res = countdig(n)

print(f"The count of digit in {n} is: {res}")

12. Write a program to print the count of each individual num in user defined range

Expected o/p

Sr = 9 er = 12

The count of digits in 9 is 1

The count of digits in 10 is 2

Note: To remove a digit from a given number 🡺num//10

Whenever we need to carry forward the current updated value to next cycle for further operations, use the same variable for operation and left hand side for updating

To convert -tive to +tive 🡺 num \* -1

def countdig(n):

      if (n < 0):

          n = n \* -1

      count = 0

      while (n > 0):

          n = n // 10

          count = count + 1

      return count

sr = int(input("enter start range"))

er = int(input("enter end range"))

if(sr > er):

     print("Invalid input")

else:

     for i in range(sr,er+1):

          res = countdig(i)

          print(f"The count of digit in {i} is: {res}")

          i = i+1

13. Armstrong number

def countdig(n):

    count = 0

    while (n > 0):

          n = n // 10

          count = count + 1

    return count

n = int(input("enter number:"))

temp = n

if (n < 0):

          n = n \* -1

pow = countdig(n)

asn = 0

while( n > 0):

    rem = n % 10 #get last digit

    asn = asn + (rem \*\* pow) #sum up the power of digits

    n = n // 10 #remove digit

if (temp == asn):

      print(f"{temp} is an Armstrong number")

else:

      print(f"{temp} is not an Armstrong number")

for negative numbers

def countdig(n):

    count = 0

    while (n > 0):

          n = n // 10

          count = count + 1

    return count

n = int(input("enter number:"))

temp = n

if (n < 0):

          n = n \* -1

pow = countdig(n)

asn = 0

while( n > 0):

    rem = n % 10 #get last digit

    asn = asn + (rem \*\* pow) #sum up the power of digits

    n = n // 10 #remove digit

if(temp < 0):

      asn = asn \* -1

if (temp == asn):

      print(f"{temp} is an Armstrong number")

else:

      print(f"{temp} is not an Armstrong number")

14. Write a program to check whether the given number is Armstrong or not using customized function

def countdig(n):

    count = 0

    while (n > 0):

          n = n // 10

          count = count + 1

    return count

def armstrong(n):

      asn = 0

      if (n < 0):

          n = n \* -1

      pow = countdig(n)

      while( n > 0):

            rem = n % 10 #get last digit

            asn = asn + (rem \*\* pow) #sum up the power of digits

            n = n // 10 #remove digit

      if(temp < 0):

            asn = asn \* -1

      if (temp == asn):

            return True

      else:

            return False

n = int(input("enter number:"))

temp = n

res = armstrong(n)

print(res)

15.Write a program to print all the Armstrong numbers of user defined range

def countdig(n):

    count = 0

    while (n > 0):

          n = n // 10

          count = count + 1

    return count

def armstrong(n):

      temp = n

      asn = 0

      if (n < 0):

          n = n \* -1

      pow = countdig(n)

      while( n > 0):

            rem = n % 10 #get last digit

            asn = asn + (rem \*\* pow) #sum up the power of digits

            n = n // 10 #remove digit

      if(temp < 0):

            asn = asn \* -1

      if (temp == asn):

            return True

      else:

            return False

sr = int(input("enter the start value"))

er = int(input("enter the end value"))

if (sr > er):

    print("invalid input")

else:

    print("Armstrong numbers :")

    for i in range(sr, er + 1):

        flag = armstrong(i)

        if flag:

             print(i, end=" ")

16. First n Armstrong numbers

def countdig(n):

    count = 0

    while (n > 0):

          n = n // 10

          count = count + 1

    return count

def armstrong(n):

      temp = n

      asn = 0

      if (n < 0):

          n = n \* -1

      pow = countdig(n)

      while( n > 0):

            rem = n % 10 #get last digit

            asn = asn + (rem \*\* pow) #sum up the power of digits

            n = n // 10 #remove digit

      if(temp < 0):

            asn = asn \* -1

      if (temp == asn):

            return True

      else:

            return False

count = int(input("enter number:"))

num = 1

while(count > 0):

      flag = armstrong(num)

      if flag:

            print(num,end =" ")

            count -= 1

      num += 1

17. Armstrong numbers and non – Armstrong numbers in separate

def countdig(n):

    count = 0

    while (n > 0):

          n = n // 10

          count = count + 1

    return count

def armstrong(n):

      temp = n

      asn = 0

      if (n < 0):

          n = n \* -1

      pow = countdig(n)

      while( n > 0):

            rem = n % 10 #get last digit

            asn = asn + (rem \*\* pow) #sum up the power of digits

            n = n // 10 #remove digit

      if(temp < 0):

            asn = asn \* -1

      if (temp == asn):

            return True

      else:

            return False

sr = int(input("enter the start value:"))

er = int(input("enter the end value:"))

if (sr > er):

    print("invalid input")

print("Armstrong numbers :")

for i in range(sr, er + 1):

      flag = armstrong(i)

      if flag:

            print(i, end=" ")

print("\nNon Armstrong numbers :")

for i in range(sr, er + 1):

      flag = armstrong(i)

      if not flag:

            print(i, end=" ")

18. Write a program to print reversal of a number

n = int(input("Enter the number"))

temp = n

rev = 0

if (n < 0):

    n = n \* -1

while(n > 0):

    rem = n % 10

    rev = (rev \* 10) + rem

    n = n // 10

if (temp < 0):

    rev = rev \* -1

print(f"the reversal of {n} is {rev}")

using customized function

def reversal(n):

    temp = n

    rev = 0

    if (n < 0):

        n = n \* -1

    while(n > 0):

        rem = n % 10

        rev = (rev \* 10) + rem

        n = n // 10

    if (temp < 0):

        rev = rev \* -1

    return rev

n = int(input("Enter the number:"))

res = reversal(n)

print(f"the reversal of {n} is {res}")

19. Write a program to reverse all the individual number of user defined range(ASS)

def reversal(n):

    temp = n

    rev = 0

    if (n < 0):

        n = n \* -1

    while(n > 0):

        rem = n % 10

        rev = (rev \* 10) + rem

        n = n // 10

    if (temp < 0):

        rev = rev \* -1

    return rev

sr = int(input("Enter the start range:"))

er = int(input("Enter the end range:"))

if(sr > er):

    print("Invalid input")

for i in range(sr,er+1):

    res = reversal(i)

    print(f"The reversal of {i} is {res} ")

20. Write a program to print all the integer palindrome of a number

def intpalindrome(n):

    temp = n

    rev = 0

    if (n < 0):

        n = n \* -1

    while(n > 0):

        rem = n % 10

        rev = (rev \* 10) + rem

        n = n // 10

    if (temp < 0):

        rev = rev \* -1

    return temp == rev

n = int(input("Enter the number:"))

flag = intpalindrome(n)

if flag:

    print(f"the number {n} is a palindrome")

else:

    print(f"the number {n} is not a palindrome")

20. Write a program to print all the integer palindromes in user defined range

#Write a program to print all the integer palindromes in user defined range

def palindrome(n):

    temp = n

    rev = 0

    if (n < 0):

        n = n \* -1

    while (n > 0):

        rem = n % 10

        rev = (rev \* 10) + rem

        n = n // 10

    if (temp < 0):

        rev = rev \* -1

    return temp == rev

sr = int(input("Enter the start range:"))

er = int(input("Enter the end range:"))

if(sr > er):

    print("Invalid input")

for i in range(sr,er+1):

    flag = palindrome(i)

    if flag:

        print(i,end=" ")

21. Write a program to print all the palindrome and non palindromic num separately in user defined range

#program to print all the palindromic and non palindromic num separately in user defined range

def palindrome(n):

    temp = n

    rev = 0

    if(n < 0):

        n = n \* -1

    while(n > 0):

        rem = n % 10

        rev = (rev \* 10) + rem

        n = n // 10

    if (temp < 0):

        rev = rev \* -1

    return (temp == rev)

sr = int(input("Enter the start range:"))

er = int(input("Enter the end range:"))

if(sr > er):

    print("Invalid input")

print("Palindromic numbers:")

for i in range(sr,er+1):

    flag = palindrome(i)

    if flag:

        print(i,end=" ")

print("\nNon palindromic numbers:")

for i in range(sr,er+1):

    flag = palindrome(i)

    if not flag:

        print(i,end=" ")

22. Write a pro to print first n palindromic numbers

#program to print first n palindromic numbers

def palindrome(n):

    temp = n

    rev = 0

    if(temp < 0):

        n = n \* -1

    while(n > 0):

        rem = n % 10

        rev = (rev \* 10) + rem

        n = n // 10

    if(temp < 0):

        rev = rev \* -1

    return (temp == rev)

count = int(input("Enter the number:"))

num = 1

while (count > 0):

    flag = palindrome(num)

    if flag:

        print(num,end=" ")

        count -= 1

    num += 1

23. Write a pro to print first n non-palindromic numbers

#program to print first n non-palindromic numbers

def palindrome(n):

    temp = n

    rev = 0

    if(temp < 0):

        n = n \* -1

    while(n > 0):

        rem = n % 10

        rev = (rev \* 10) + rem

        n = n // 10

    if(temp < 0):

        rev = rev \* -1

    return (temp == rev)

count = int(input("Enter the number:"))

num = 1

while (count > 0):

    flag = palindrome(num)

    if not flag:

        print(num,end=" ")

        count -= 1

    num += 1

24. Write a program to print all the factors of given number

Factors Divisibility or Deduction (Modulus)

* Note: A value is said to be a factor of a number on;y if the value can completely divide and reduce the num to 0
* All the factors of a num will be in range of 1 to the num itself
* Every num will have minimum of two factors 1 and num itself

def factors(n):

    for i in range(1,n+1):

        if(n % i == 0):

            print(i,end=" ")

n = int(input("Enter the number:"))

print(f"The factors of {n} are:")

factors(n)

26. Write a program to print all the factors of user defined range

o/p: the factors of 1 is 1

def factors(n):

    count = 0

    for i in range(1, n + 1):

        if n % i == 0:

            print(i, end=" ")

            count += 1

    return count

sr = int(input("Enter start range:"))

er = int(input("Ente end range:"))

if (sr > er):

    print("Invalid input")

for i in range(sr, er + 1):

    print(f"\nFactors of {i} are: ", end="")

    res = factors(i)

    print(f"\nThe count is : {res}")

27. Write a program to count the number of factors for given input number

def factors(n):

    for i in range(1,n+1):

        if(n % i == 0):

            print(i,end=" ")

def countfac(n):

    countfact = 0

    for i in range(1,n+1):

        if (n % i == 0):

            countfact += 1

    return countfact

n = int(input("Enter the number:"))

print(f"The factors of {n} are:")

factors(n)

res = countfac(n)

print(f"\nThe count of {n} is {res}")

Optimized logic for printing factors

All the factors of a number can be printed within the direct square root or the lower nearest square root of given number

28. Write a program to print factors of a number

n = int(input("Enter the number:"))

i = 1

while(i \* i <= n):

    if ((n % i) == 0):

        print(i,end=" ")

        if (i != (n // i)):

            print((n // i),end=" ")

    i += 1

29. Write a program to count the number of factors of given numbers

def countfactors(n):

    i = 1

    count = 0

    cyccount = 0

    while(i \* i <= n):

        if (n % i == 0):

            print(i, end=" ")

            count = count + 1

            if (i != (n // i)):

                print(n // i, end=" ")

                count = count + 1

        i += 1

    cyccount += 1

    return cyccount,count

n = int(input("Enter the number:"))

res1,res2 = countfactors(n)

print(f"The count {n} of  is {res2}")

print(f"\nThe countcycle of  is {res1}")

30. Write a program to check wheather a given number is prime or not

No -tive prime numbers

The number which have exactly 2 factors are called prime numbers

def prime(n):

    i = 1

    count = 0

    while(i \* i <= n):

        if (n % i == 0):

            print(i, end=" ")

            count = count + 1

            if (i != (n // i)):

                print(n // i, end=" ")

                count = count + 1

        i += 1

    return count == 2

n = int(input("Enter the number:"))

flag = prime(n)

if flag:

    print(f"\n{n} is prime")

else:

    print(f"\n{n} is not prime")

ASS

Write a program to check wheather number is prime or not

Write a program to check wheather number is prime or not using customized function

Write a program to print the prime numbers of user defined range

#program to print the prime numbers of user defined range

def primerange(n):

    i = 1

    count = 0

    while(i \* i <= n):

        if (n % i == 0):

            count = count + 1

            if (i != (n // i)):

                count = count + 1

        i += 1

    return count == 2

sr = int(input("Enter the start range:"))

er = int(input("Enter the end range:"))

primes = []

for i in range(sr, er + 1):

    flag = primerange(i)

    if flag:

        primes.append(i)

if primes:

    print(f"The prime numbers from {sr} to {er} are:", \*primes)

Write a program to print all prime and non prime numbers

def primerange(n):

    i = 1

    count = 0

    while(i \* i <= n):

        if (n % i == 0):

            count = count + 1

            if (i != (n // i)):

                count = count + 1

        i += 1

    return count == 2

sr = int(input("Enter the start range:"))

er = int(input("Enter the end range:"))

primes = []

nonprimes = []

for i in range(sr, er + 1):

    flag = primerange(i)

    if flag:

        primes.append(i)

    else:

        nonprimes.append(i)

if primes:

    print(f"Prime numbers from {sr} to {er} are:", \*primes)

    print(f"Non prime numbers from {sr} to {er} are:", \*nonprimes)

Write a program to print first n prime and non prime numbers

#program to print first n prime numbers

def isprime(n):

    i = 1

    count = 0

    while(i \* i <= n):

        if (n % i == 0):

            count = count + 1

            if(i != (n//i)):

                count = count + 1

        i = i + 1

    return count == 2

count = int(input("Enter the number"))

num = 2

while count > 0:

    if isprime(num):

        print(num,end=" ")

        count -= 1

    num += 1

#program to print first n non prime numbers

def isprime(n):

    i = 1

    count = 0

    while(i \* i <= n):

        if (n % i == 0):

            count = count + 1

            if(i != (n//i)):

                count = count + 1

        i = i + 1

    return count == 2

count = int(input("Enter the number"))

num = 2

while (count > 0):

    if not isprime(num):

        print(num,end=" ")

        count -= 1

    num += 1

GCD or HCF of 2 given numbers

1. Write a program to display the HCF or GCD of given 2 numbers(Greatest common divisor)

The largest common factor that divides the given 2 input numbers

Note: The smaller number will not have a factor in the higher range of larger number

n1 = int(input("Enter the first number:"))

n2 = int(input("Enter the second number:"))

lower = n1

if (n2 < n1):

    lower = n2

hcf = 1

for i in range(2, lower+1):

    if (n1 % i == 0 and n2 % i == 0):

        hcf = i

print(f"The HCF of {n1} and {n2} is:{hcf}")

using function

def hcf(n1,n2):

    lower = n1

    if (n2 < n1):

        lower = n2

    hcf = 1

    for i in range(2,lower+1):

        if (n1 % i == 0 and n2 % i == 0):

            hcf = i

    return hcf

n1 = int(input("Enter the first number:"))

n2 = int(input("Enter the second number:"))

res = hcf(n1,n2)

print(f"The HCF of {n1} and {n2} is:{res}")

Fibonacci

The initial 2 values of the series will always be 0 and 1

The basic logic is sum the previous 2 position values

def decrewhile(pos):

    n1 = 0

    n2 = 1

    temp = 1

    while(pos > 0):

        print(n1,end=" ")

        temp = n1 + n2

        n1 = n2

        n2 = temp

        pos -= 1

pos = int(input("Enter your position:"))

decrewhile(pos)

#Incrementing while

def increwhile(pos):

    n1 = 0

    n2 = 1

    temp = 1

    i = 0

    while(pos > i):

        print(n1,end=" ")

        temp = n1 + n2

        n1 = n2

        n2 = temp

        i += 1

pos = int(input("Enter your position:"))

increwhile(pos)

#Incrementing for loop

def increfor(pos):

    n1 = 0

    n2 = 1

    temp = 1

    for i in range(pos):

        print(n1, end=" ")

        temp = n1 + n2

        n1 = n2

        n2 = temp

pos = int(input("Enter your position:"))

increfor(pos)

#Decrementing for loop

def decrefor(pos):

    n1 = 0

    n2 = 1

    temp = 1

    for i in range(pos, 0, -1):

        print(n1, end=" ")

        temp = n1 + n2

        n1 = n2

        n2 = temp

pos = int(input("Enter your position:"))

decrefor(pos)

Stack is temporary memory where the current executing variable

Python compiler

* It checks for syntactical mistakes
* It includes the compulsory line of code required by the PVM to execute the logic
* It converts the HLL to intermediate code/ Byte code

Return

* It is a pre-defined keyword
* It is the last executable line of code in a function
* Uses:
* It helps to return val or val’s from called function
* It returns the execution of PVM from called function back to function call
* It returns the memory that was provided during function call to PVM

**Recursion:**

The function calling itself repeatedly for certain number of times to execute the same logic

Syn: Without return value

def func\_name(parameters):

#base cond

#logic

func\_name(arguments) #recursive fun call

func\_name(initial value for parameters) #initial function call

Syn: With return value

def func\_name(parameters):

#base cond

#logic

return func\_name(arguments) #recursive fun call

func\_name(initial value for parameters) #initial function call

def printnum(n):

    if (n <= 0):

        return

    print(n,end=" ")

    printnum(n-1)

n = int(input("Enter the number:"))

printnum(n)

WAL to print

I/P: n = 4

Expected O/P : 4 3 2 1 1 2 3 4

def printnum(n):

    if (n <= 0):

        return

    print(n,end=" ")

    printnum(n-1)

    print(n,end=" ")

#n = int(input("Enter the number:"))

printnum(4)

WAL to print

I/P: n = 4

Expected O/P : 1 2 3 4 4 3 2 1

def printnum(n):

    if (n <= 0):

        return

    print(n,end=" ")

    printnum(n-1)

    print(n,end=" ")

def display(i,n):

    if (i > n):

        return

    print(i, end=" ")

    display((i+1),n)

    print(i, end=" ")

n = int(input("Enter the number:"))

printnum(n)

print("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

display(1,n)

Recursive function call

It will be included within the recursive function definition (body of the fun)

It passes the updated value that are required by the call

Normal function call

It will be included outside function to initiate the function execution

It passes the first value of parameters

1. Write a program to count the digits using recursion

def countdig(n,count):

    if (n  <= 0):

        return count

    n = n // 10

    count += 1

    return countdig(n,count)

n = int(input("Enter the number:"))

res = countdig(n,0)

print(f"The count of dig  {n} is:{res}")

2. Write a program to Armstrong using recursion

def countdig(n,count):

    if (n  <= 0):

        return count

    n = n // 10

    count += 1

    return countdig(n,count)

def armstrong(n,pow,asn,temp):

    if (n <= 0):

        return temp == asn

    base = n % 10

    asn = asn + (base\*\*pow)

    n = n // 10

    return armstrong(n,pow,asn,temp)

n = int(input("Enter the number:"))

pow = countdig(n,0)

flag = armstrong(n,pow,0,n)

if flag:

    print(f"{n} is armstrong number")

else:

    print(f"{n} is not armstrong")

3. Write a program for reversal of a number using recursion

#Reversal of a number

def reversal(n,rev):

    if (n <= 0):

        return rev

    rem = n % 10

    rev = (rev \* 10) + rem

    n = n // 10

    return reversal(n,rev)

n = int(input("Enter n:"))

res = reversal(n,0)

print(f"The reversal of {n} is:{res}")

4. Write a program to check palindrome of a number using recursion

def palindrome(n,rev,temp):

    if (n <= 0):

        return temp == rev

    rem = n % 10

    rev = (rev \* 10) + rem

    n = n //10

    return palindrome(n,rev,temp)

n = int(input("Enter n:"))

flag = palindrome(n,0,n)

if flag:

    print(f"{n} is palindrom")

else:

    print(f"{n} is not palindrome")

5 Write a program to find factors using recursion

def printfac(n,i):

    if (i > n):

        return

    if (n % i == 0):

        print(i,end=" ")

    return printfac(n, i + 1)

n = int(input("Enter n:"))

printfac(n,1)

def countfac(n,i,count):

    if (i \* i > n):

        return count

    if (n % i == 0):

        count +=1

        if (i != (n//i)):

            count += 1

    return countfac(n,(i + 1),count)

n = int(input("Enter n:"))

res = countfac(n,1,0)

print(f"the count of{n} is: {res}")

def factorial(n,i):

    if (i \* i >= n):

        return

    if ((n % i) == 0):

        print(i,end=" ")

        if (i != (n // i)):

            print((n // i),end=" ")

    return factorial(n,(i +1))

n = int(input("Enter n:"))

factorial(n,1)

def faccount(n,i,count):

    if (i > n):

        return count

    if (n % i == 0):

        count += 1

    return faccount(n,i+1,count)

n = int(input("Enter n:"))

res = faccount(n,1,0)

print(f"The factor count of {n} is:{res}")

6. Write a program to check the number is prime or not using recursion

def isprime(n,i,count):

    if (i \* i > n):

        return count == 2

    if (n % i == 0):

        count +=1

        if (i != (n//i)):

            count += 1

    return isprime(n,(i + 1),count)

n = int(input("Enter n:"))

flag = isprime(n,1,0)

if flag:

    print(f"{n} is prime")

else:

    print(f"{n} is not prime")

7. Write a program to print the GCD of the given 2 numbers using recursion

def gcd(n1,n2,i,hcf,lower):

    if (i > lower):

        return hcf

    if(n1 % i == 0 and n2 % i == 0):

        hcf = i

    return gcd(n1,n2,i+1,hcf,lower)

n1 = int(input("Enter n1:"))

n2 = int(input("enter n2:"))

lower = n1

if (n2 > n1):

    lower = n2

res = gcd(n1,n2,2,1,lower)

print(f"The hcf of {n1} and {n2} is :{res}")

The above logic the number of cycle count. To reduce the cycle the below logic is better

def gcd(n1,n2):

    if n1 <= 0:

        return n2

    if (n2 > n1):

        n1, n2 = n2, n1

    #return gcd((n1 - n2), n2) or

    return gcd((n1 % n2), n2)

n1 = int(input("Enter n1:"))

n2 = int(input("enter n2:"))

res = gcd(n1,n2)

print(f"The hcf of {n1} and {n2} is :{res}")

The largest number of complete integer range is 2147483648

The smallest number of complete integer range is -2147483648

8. Write a program to pint Fibonacci series using recursion

def fibonacci(pos,n1,n2):

    if (pos <= 0):

        return

    print(n1,end=" ")

    temp = n1 + n2

    #n1 = n2

    #n2 = temp

    return fibonacci(pos-1,n1 = n2,n2 = temp)

pos = int(input("enter your position:"))

fibonacci(pos,0,1)

9. Write a program to print factorial of a number

def printfac(n,i):

    if (i > n):

        return

    if (n % i == 0):

        print(i,end=" ")

    return printfac(n, i + 1)

n = int(input("Enter n:"))

printfac(n,1)

def factorial(n,i):

    if (i \* i >= n):

        return

    if ((n % i) == 0):

        print(i,end=" ")

        if (i != (n // i)):

            print((n // i),end=" ")

    return factorial(n,(i +1))

n = int(input("Enter n:"))

factorial(n,1)

Factorial count

def countfac(n,i,count):

    if (i \* i > n):

        return count

    if (n % i == 0):

        count +=1

        if (i != (n//i)):

            count += 1

    return countfac(n,(i + 1),count)

n = int(input("Enter n:"))

res = countfac(n,1,0)

print(f"the count of{n} is: {res}")

def faccount(n,i,count):

    if (i > n):

        return count

    if (n % i == 0):

        count += 1

    return faccount(n,i+1,count)

n = int(input("Enter n:"))

res = faccount(n,1,0)

print(f"The factor count of {n} is:{res}")

10. Write a program to check a number is happy number or not

def happy(n):

    if n == 1:

        return True

    elif n == 4:

        return False

    sum = 0

    while n > 0:

        base = n % 10

        sum += base \* base

        n = n // 10

    return happy(sum)

n = int(input("Enter a number: "))

res = happy(n)

print(res)

Patterns

#Square

n = int(input("Enter n:"))

for i in range(1,n+1):

    for j in range(1,n+1):

        print("\*",end="")

    print()

#LHS right angle triangle

n = int(input("Enter n:"))

for i in range(1,n+1):

    for j in range(1,i+1):

        print("\*",end="")

    print()

#Inverted right angle triangle

n = int(input("Enter a number:"))

for i in range(n, (1-1),-1):

    for j in range(1,(i+1)):

        print("\*",end="")

    print()

#Inverted right angle triangle opposite

n = int(input("Enter n"))

for i in range(1,(n+1)):

    for k in range(n,i,-1):

        print(" ",end="")

    for j in range(1,i+1):

        print("\*",end="")

    print()

#Pyramid pattern

n = int(input("Enter n:"))

for i in range(1,(n+1)):

    for k in range(n,i,-1):

        print(" ",end="")

    for j in range(1,i+1):

        print("\* ",end="")

    print()

#RHS inverted right angle tringle

n = int(input("Enter n:"))

for i in range(n,(1-1),-1):

    for k in range(n,i,-1):

        print(" ",end="")

    for j in range(1,i+1):

        print("\*",end="")

    print()

# Triangular patterns

#Inverted pyramid

n = int(input("Enter n:"))

for i in range(n,1-1,-1):

    for k in range(n,i,-1):

        print(" ",end="")

    for j in range(1,i+1):

        print("\* ",end="")

    print()

#Odd right angle triangle

odd = 1

for i in range(1,n+1):

    for j in range(1,odd+1):

        print("\*",end="")

    odd = odd + 2

    print()

#Using explicit spacing

odd = 1

for i in range(1,n+1):

    for k in range(n,i,-1):

        print(" ",end="")

    for j in range(1,odd+1):

        print("\*",end="")

    odd = odd + 2

    print()

#Inverted odd pyramid

odd = (n \* 2) - 1

for i in range(n,1-1,-1):

    for k in range(n,i,-1):

        print(" ",end="")

    for j in range(1,odd+1):

        print("\*",end="")

    odd = odd - 2

    print()

#In straight triangles the order of i and j will be same

#In inverted triangles the order of i and j will be opposite

Assignment  
n = 4 Parallellogram

n = int(input("Enter n:"))

for i in range(1,n+1):

    for j in range(1,n+1):

        print("\*",end="")

    print()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

for i in range(1,n+1):

    for k in range(1,i,+1):

        print(" ",end="")

    for j in range(1,n+1):

        print("\*",end="")

    print()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

for i in range(1,n+1):

    for k in range(n,i,-1):

        print(" ",end="")

    for j in range(1,n+1):

        print("\*",end="")

    print()

Combinational Patterns

n = int(input("Enter n:"))

# K pattern

noc = n

for i in range(1, n \* 2):

    for j in range(noc):

        print("\*", end=" ")

    print()

    if i < n:

        noc -= 1

    else:

        noc += 1

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

# #X pattern

noc = n

for i in range(1, n \* 2):

    for k in range(n, noc, -1):

        print(" ", end="")

    for j in range(noc):

        print("\*", end=" ")

    print()

    if i < n:

        noc -= 1

    else:

        noc += 1

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

#inverted K pattern

noc = n

for i in range(1, n \* 2):

    for k in range(n-noc):

        print(" ", end=" ")

    for j in range(noc):

        print("\*", end=" ")

    print()

    if i < n:

        noc -= 1

    else:

        noc += 1

Enter n:4

\* \* \* \*

\* \* \*

\* \*

\*

\* \*

\* \* \*

\* \* \* \*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\* \* \* \*

\* \* \*

\* \*

\*

\* \*

\* \* \*

\* \* \* \*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\* \* \* \*

\* \* \*

\* \*

\*

\* \*

\* \* \*

\* \* \* \*

n = int(input("Enter n:"))

noc = 1

for i in range(1,(n\*2)):

    for j in range(1,noc+1):

        print("\*",end="")

    if i < n:

        noc +=1

    else:

        noc -=1

    print()

#Opposite of above

noc = 1

for i in range(1,(n\*2)):

    for k in range(n,noc,-1):

        print(" ",end="")

    for j in range(1,noc+1):

        print("\*",end="")

    if i < n:

        noc +=1

    else:

        noc -=1

    print()

#Diamond pattern

noc = 1

for i in range(1,(n\*2)):

    for k in range(n,noc,-1):

        print(" ",end="")

    for j in range(1,noc+1):

        print("\* ",end="")

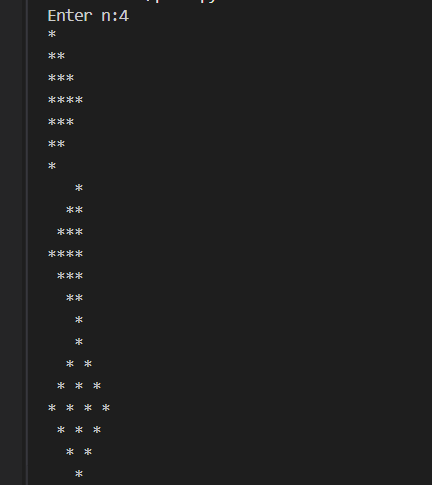
    if i < n:

        noc +=1

    else:

        noc -=1

    print()



Butterfly pattern

n = int(input("Enter n:"))

noc = 1

nor = (n\*2)-1

for i in range(1,2\*n):

    for j in range(1,2\*n):

        if (j <= noc) or (j >= nor):

            print("\*",end="")

        else:

            print(" ",end="")

    if i < n:

        noc += 1

        nor -= 1

    else:

        noc -= 1

        nor += 1

    print()



HOLLOW PATTERNS

Square patterns

n = int(input("Enter n:"))

for i in range(1,n+1):

    for j in range(1,n+1):

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

            print("\*",end=" ")

        else:

            print(" ",end=" ")

    print()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

if n % 2 == 0:

    n += 1

for i in range(1,n+1):

    for j in range(1,n+1):

        if (i == 1 or i == n or j == 1 or j == n or i==j or (i+j == n+1)):

            print("\*",end=" ")

        else:

            print(" ",end=" ")

    print()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

if n % 2 == 0:

    n += 1

for i in range(1,n+1):

    for j in range(1,n+1):

        if (i == 1 or i == n or i==j or (i+j == n+1)):

            print("\*",end=" ")

        else:

            print(" ",end=" ")

    print()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

if n % 2 == 0:

    n += 1

for i in range(1,n+1):

    for j in range(1,n+1):

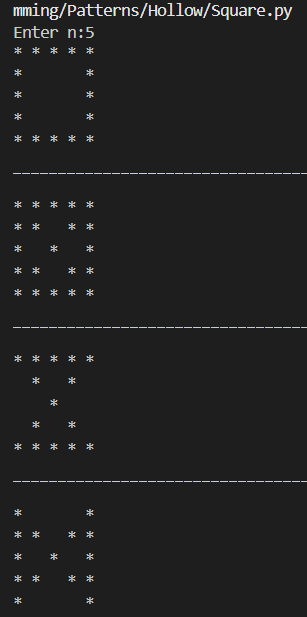
        if (j == 1 or j == n or i==j or (i+j == n+1)):

            print("\*",end=" ")

        else:

            print(" ",end=" ")

    print()



Hollow patterns problems

n = int(input("Enter n:"))

#LHS Right angle triangle

for i in range(1,n+1):

    for j in range(1,n+1):

        if (i== n or j == 1 or i==j ):

            print("\*",end=" ")

        else:

            print(" ",end=" ")

    print()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

#RHS Right angle triangle

for i in range(1,n+1):

    for k in range(n,i,-1):

        print(" ",end="")

    for j in range(1,i+1):

        if (i== n or j == 1 or i==j ):

            print("\*",end="")

        else:

            print(" ",end="")

    print()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

#Pyramid

for i in range(1,n+1):

    for k in range(n,i,-1):

        print(" ",end="")

    for j in range(1,n+1):

        if (i== n or j == 1 or i==j ):

                print("\*",end=" ")

        else:

            print(" ",end=" ")

    print()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

# Inverted LHS Right angle triangle

for i in range(n,1-1,-1):

    for j in range(1,n+1):

        if (i== n or j == 1 or i==j ):

            print("\*",end="")

        else:

            print(" ",end="")

    print()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

#Inverted RHS Right angle triangle

for i in range(n,1-1,-1):

    for k in range(n,i,-1):

        print(" ",end="")

    for j in range(1,n+1):

        if (i== n or j == 1 or i==j ):

            print("\*",end="")

        else:

            print(" ",end="")

    print()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

#Inverted pyramid

for i in range(n,(1-1),-1):

    for k in range(n,i,-1):

        print(" ",end="")

    for j in range(1,n+1):

        if (i== n or j == 1 or i==j ):

                print("\* ",end="")

        else:

            print("  ",end="")

    print()

#Combinational patterns

noc = 1

for i in range(1,(n\*2)):

    for j in range(1,noc+1):

        if (j == 1 or j == noc):

            print("\*",end="")

        else:

            print(" ",end="")

    if i < n:

        noc +=1

    else:

        noc -=1

    print()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

noc = 1

for i in range(1,(n\*2)):

    for k in range(n,noc,-1):

        print(" ",end="")

    for j in range(1,noc+1):

        if (j == 1 or j == noc):

            print("\*",end="")

        else:

            print(" ",end="")

    if i < n:

        noc +=1

    else:

        noc -=1

    print()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

#Diamond

noc = 1

for i in range(1,(n\*2)):

    for k in range(n,noc,-1):

        print(" ",end="")

    for j in range(1,noc+1):

        if (j == 1 or j == noc):

            print("\* ",end="")

        else:

            print("  ",end="")

    if i < n:

        noc +=1

    else:

        noc -=1

    print()

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

#K pattern

noc = n

for i in range(1,n\*2):

    for j in range(1,noc+1):

        if ( i == 1 or i == n or i == (n \*2)-1 or j == 1 or j == noc):

            print("\*",end="")

        else:

            print(" ",end="")

    print()

    if i < n:

        noc -=1

    else:

        noc +=1

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

#RHS K pattern

noc = n

for i in range(1,n\*2):

    for k in range(n,noc,-1):

        print(" ", end="")

    for j in range(1,noc+1):

        if ( i == 1 or i == (n \*2)-1 or j == 1 or j == noc):

            print("\*",end="")

        else:

            print(" ",end="")

    print()

    if i < n:

        noc -=1

    else:

        noc +=1

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

#X pattern

noc = n

for i in range(1,n\*2):

    for k in range(n,noc,-1):

        print(" ", end="")

    for j in range(1,noc+1):

        if ( i == 1 or i == (n \*2)-1 or j == 1 or j == noc):

            print("\* ",end="")

        else:

            print("  ",end="")

    print()

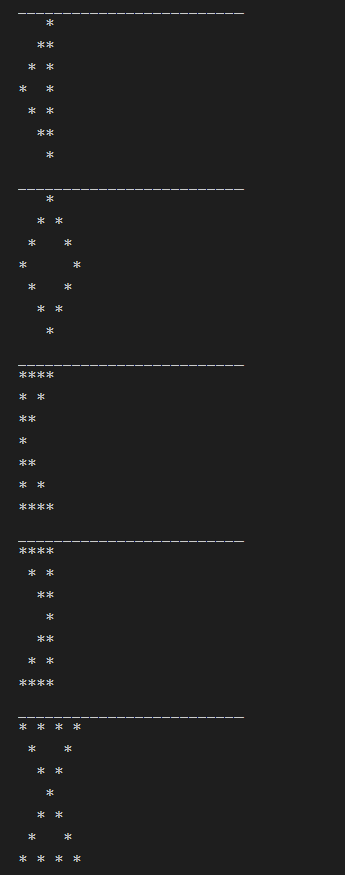
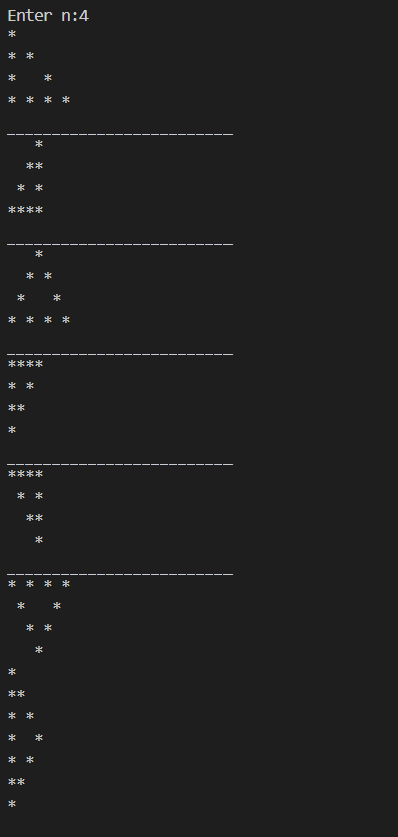
    if i < n:

        noc -=1

    else:

        noc +=1

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")



NUMBER PATTERNS

n = int(input("enter n:"))

for i in range(1,(n+1)):

    for j in range(1,(n+1)):

        print(i,end=" ")

    print()

# enter n:4

# 1 1 1 1

# 2 2 2 2

# 3 3 3 3

# 4 4 4 4

for i in range(n,(1-1),-1):

    for j in range(n,(i-1),-1):

        print(i,end=" ")

    print()

#O/p

# enter n:4

# 1 1 1 1

# 2 2 2 2

# 3 3 3 3

# 4 4 4 4

# 4

# 3 3

# 2 2 2

# 1 1 1 1

for i in range(1,n+1):

    for j in range(n,(i-1),-1):

        print(j,end=" ")

    print()

# enter n:4

# 4 3 2 1

# 4 3 2

# 4 3

# 4

for i in range(n,(1-1),-1):

    for j in range(n,(1-1),-1):

        print(i,end=" ")

    print()

#O/p

# 4 4 4 4

# 3 3 3 3

# 2 2 2 2

# 1 1 1 1

for i in range(1,n+1):

    for k in range(1,i):

        print(" ",end="")

    for j in range(i,(n+1)):

        print(j,end="")

    print()

#O/P

# enter n:4

# 1234

#  234

#   34

#    4

for i in range(1,(n+1)):

    for j in range(i,(n+i)):

        print(j,end="")

    print()

# enter n:4

# 1234

# 2345

# 3456

# 4567

n = int(input("Enter n:"))

noc = n

for i in range(1,(n\*2)):

    for j in range(1,noc+1):

        print(j,end="")

    print()

    if i < n:

        noc -= 1

    else:

        noc +=1

#O/P

# Enter n:4

# 1234

# 123

# 12

# 1

# 12

# 123

# 1234

noc = n

for i in range(1,(n\*2)):

    for j in range(noc,0,-1):

        print(j,end="")

    print()

    if i < n:

        noc -= 1

    else:

        noc +=1

#O/P

# Enter n:4

# 4321

# 321

# 21

# 1

# 21

# 321

# 4321

1234

2234

3334

4444

n = int(input("Enter n:"))

for i in range(1,(n+1)):

    for j in range(1,(i+1)):

        if ((i + j) % 2 == 0):

            print("1",end=" ")

        else:

            print("0",end=" ")

    print()

# Enter n:4

# 1

# 0 1

# 1 0 1

# 0 1 0 1

for i in range(1,(n+1)):

    for j in range(1,(n+1)):

        if ((i + j) % 2 == 0):

            print("0",end=" ")

        else:

            print("1",end=" ")

    print()

# Enter n:4

# 0 1 0 1

# 1 0 1 0

# 0 1 0 1

# 1 0 1 0

for i in range(1,(n+1)):

    for j in range(1,(n+1)):

        if (i < j):

            print(i,end=" ")

        else:

            print(j,end=" ")

    print()

# Enter n:4

# 1 1 1 1

# 1 2 2 2

# 1 2 3 3

# 1 2 3 4

for i in range(n,(1-1),-1):

    for j in range(n,(1-1),-1):

        if (i > j):

            print(j,end=" ")

        else:

            print(i,end=" ")

    print()

# Enter n:4

# 4 3 2 1

# 3 3 2 1

# 2 2 2 1

# 1 1 1 1

for i in range(1,(n+1)): #Need to modify

    for j in range(1,(n+1)):

        if(j > i):

            print(j,end=" ")

        else:

            print(i,end=" ")

    print()

# Enter n:5

# 1 2 3 4 5

# 2 2 3 4 5

# 3 3 3 4 5

# 4 4 4 4 5

# 5 5 5 5 5

if (n % 2 == 0):

    n = n + 1

for i in range(1,(n+1)):

    for j in range(1,(n+1)):

        if (i == j or (i+j == n+1)):

            print(i,end=" ")

        else:

            print("\*",end=" ")

    print()

# Enter n:4

# 1 \* \* \* 1

# \* 2 \* 2 \*

# \* \* 3 \* \*

# \* 4 \* 4 \*

# 5 \* \* \* 5

if (n % 2 == 0):

    n = n + 1

for i in range(1,(n+1)):

    for j in range(1,(n+1)):

        if (i == j or (i+j == n+1)):

            print(j,end=" ")

        else:

            print("\*",end=" ")

    print()

# Enter n:4

# 1 \* \* \* 5

# \* 2 \* 4 \*

# \* \* 3 \* \*

# \* 2 \* 4 \*

# 1 \* \* \* 5

temp = 1

for i in range(1,(n+1)):

    for j in range(1,(n+1)):

        print(temp,end=" ")

        temp = temp + 1

    print()

# # Enter n:4

# # 1 2 3 4

# # 5 6 7 8

# # 9 10 11 12

# # 13 14 15 16

tempincre = 1

for i in range(1,(n+1)):

    for j in range(1,(n+1)):

        if (i % 2 != 0):

            print(tempincre,end=" ")

            tempdecre = tempincre

        else:

            print(tempdecre,end=" ")

            tempdecre = tempdecre - 1

        tempincre = tempincre + 1

    print()

# # Enter n:4

# # 1 2 3 4

# # 4 3 2 1

# # 9 10 11 12

# # 12 11 10 9

tempincre = 1

for i in range(1,(n+1)):

    for j in range(1,(i+1)):

        if (i % 2 != 0):

            print(tempincre,end=" ")

            tempdecre = tempincre

        else:

            print(tempdecre,end=" ")

            tempdecre = tempdecre - 1

        tempincre = tempincre + 1

    print()

    tempdecre = tempincre + i

# Enter n:4

# 1

# 3 2

# 4 5 6

# 10 9 8 7

Alphabetic patterns

print(ord("A"))

print(ord(" "))

print(ord("a"))

#print(ord("")) --- Error

#print(ord("ab")) --- Error

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

print(chr(122))

print(chr(0))

print(chr(2222))

#print(chr(-22)) --- Error

Logic:

When ever capital letters 🡪 (64 + val)

Small letters 🡪 (96+val)

# for i in range(1,(n+1)):

#     for j in range(1,(i+1)):

#         print(chr(96+i),end="")

#     print()

# Enter n:4

# a

# bb

# ccc

# dddd

# for i in range(n,(1-1),-1):

#     for j in range(1,(i+1)):

#         print(chr(64+j),end="")

#     print()

# Enter n:4

# ABCD

# ABC

# AB

# A

# for i in range(1,(n+1)):

#     for j in range(1,(n+1)):

#         if (i % 2 != 0):

#             print(chr(64+j),end="")

#         else:

#             print(chr(96+j),end="")

#     print()

# Enter n:4

# ABCD

# abcd

# ABCD

# abcd

# for i in range(n,(1-1),-1):

#     for j in range(1,(i+1)):

#         if (j % 2 == 0):

#             print(chr(64+j),end="")

#         else:

#             print(chr(96+j),end="")

#     print()

# Enter n:4

# aBcD

# aBc

# aB

# a

noc = 1

for i in range(1,(n\*2)):

    for j in range(noc,(n+1)):

        print(chr(96+j),end="")

    print()

    if i < n:

        noc += 1

    else:

        noc -= 1

# abcd

# bcd

# cd

# d

# cd

# bcd

# abcd

temp = 1

for i in range(1,(n+1)):

    for j in range(1,(n+1)):

        if(temp % 2 != 0):

            print(chr(64+temp),end="")

        else:

            print(chr(96+temp),end="")

        temp = temp +1

    print()

# Enter n:4

# AbCd

# EfGh

# IjKl

# MnOp

**ARRAYS**

Array is a data structure that can hold multiple individual elements of same or different datatypes into the same shared memory location

DATA structure :- WE can store multiple individual elements the same shared memory location in a particular organized (organized or unorganized) memory

EX: arrays, linked lists, stack, queue, trees, graphs

* It stores elements into contiguous memory locations
* One shared memory loc is further sub-divided and each sub-division has it’s whole called as index, to access ech element individually
* It stores data of homogeneous (same datatype) or heterogeneous
* Array are represented 🡪 “[]”, where elements are segregated with “,”
* It can hold duplicates
* It is a indexed data structure
* Index:
* It is awhole number that starts from 0 to positive integer
* First index 🡪 0, last index 🡪 len(array)-1
* Index is assigned by PVM during execution time
* Manually if we try to access the index that is not available in the created list the PVM will through an “INDEERROR”
* Len():
* It is a predefined function which returns the count of num if elements available in a created iterable object
* Syn:
* len(created\_iterable\_obj)
* Returns a whole number
* ele = input("Enter val:")
* print(ele)
* print(type(ele))
* print("============================")
* try:
* ele = int(input("Enter val:"))
* print(ele)
* print(type(ele))
* except Exception as e:
* print("Invalid I/O")
* #Write a program to create a dynamic integer array with user define values
* def creatarr():
* l1 = []
* while True:
* try:
* n = int(input("enter num:"))
* l1.append(n)
* except Exception as e:
* return l1
* arr = creatarr()
* print(arr)
* #Write a program to create a dynamic character array with user define values
* def stringarray():
* l1 = []
* while True:
* try:
* n = input("enter char:")
* if n == "":
* return l1
* l1.append(n)
* except Exception as e:
* return l1
* arr = stringarray()
* print(arr)
* def chararray():
* l1 = []
* while True:
* n = input("enter char:")
* if n == "":
* return l1
* l1.append(n[0])
* arr = chararray()
* print(arr)

1. Write a program to find to find the number in given array using linear search

def createarray():

    l1=[]

    while True:

        try:

            n=int(input("enter a num: "))

            l1.append(n)

        except Exception as e:

            return l1

print("enter values into an array")

arr=createarray()

target=int(input("enter a target element to be searched: "))

flag, index= False, -1

for i in range(0, len(arr)):

    if target==arr[i]:

        flag=True

        index=i

        break

if flag:

    print(f"the {target} element is found at:{index}")

else:

    print(f"the {target} element is not found")

#Using function

def createarr():

    l1 = []

    while True:

        try:

            n = int(input("Enter n: "))

            l1.append(n)

        except Exception as e:

            return l1

def linearsearch(arr,target):

    for i in range(0, len(arr)):

        if target == arr[i]:

            return True,i

    return False, i

print("Enter the values into the array:")

arr = createarr()

target = int(input("Enter the number to be searched"))

flag, index = linearsearch(arr,target)

if flag:

    print(f"{target} found at index {index}")

else:

    print(f"{target} not found")

2. Write a program to find the largest and smallest element in given array along with its index

#Maximum number of the given array

def creatarr():

    l1 = []

    while True:

        try:

            n = int(input("ENter num:"))

            l1.append(n)

        except Exception as e:

            return l1

def findmax(arr):

    max = (-2\*\*31)-1

    maxind = -1

    for i in range(0, len(arr)):

        if arr[i] > max:

            max = arr[i]

            maxind = i

    return max, maxind

arr = creatarr()

max, maxind = findmax(arr)

print(f"The largest is {max} found at {maxind}")

#Minimum of given array

def findmin(arr):

    min = (2\*\*31)

    minind = -1

    for i in range(0, len(arr)):

        if arr[i] < min:

            min = arr[i]

            minind = i

    return min, minind

arr = creatarr()

min, minind = findmin(arr)

print(f"The smallest is {min} found at {minind}")

3. Write a program to find the 2nd maximum and minimum element along with its index

#Program to find second maximum number

def creatarr():

    l1 = []

    while True:

        try:

            n = int(input("ENter num:"))

            l1.append(n)

        except Exception as e:

            return l1

def secmax(arr):

    max = ((-2\*\*31)-1)

    maxind = -1

    sec = ((-2\*\*31)-1)

    secind = -1

    for i in range(0, len(arr)):

        if max < arr[i]:

            sec = max

            secind = maxind

            max = arr[i]

            maxind = i

        elif (max != arr[i] and sec < arr[i]):

            sec = arr[i]

            secind = i

    return [max, maxind, sec, secind]

arr = creatarr()

res = secmax(arr)

print(f"The second max number is {res[2]} found at {res[3]}")

#Program to find second minimun number

def secmin(arr):

    min = (2\*\*31)

    minind = -1

    sec = (2\*\*31)

    secind = -1

    for i in range(0, len(arr)):

        if arr[i] < min:

            sec = min

            secind = minind

            min = arr[i]

            minind = i

        elif (min != arr[i] and sec > arr[i]):

            sec = arr[i]

            secind = i

    return [min, minind, sec, secind]

arr = creatarr()

res = secmin(arr)

print(f"The second min number is {res[2]} found at {res[3]}")