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