Open in Colab

Basic programings in python

Print Hello world!

In [ ]:
```
print("Hello", "World", sep="_", end="!")
```

```
Hello_World!
```

Add Two numbers

In [ ]:
```
def add(n, m):
  return n+m
num1 = int(input("Enter 1st number:"))
num2 = int(input("Enter 2nd number:"))
result = add(num1, num2)
print("Addition of num1 and num2 are: {}".format(result))
```

```
Enter 1st number:4
Enter 2nd number:66
Addition of num1 and num2 are:70
```

Find square root

In [ ]:
```
def square(n):
  return n*n
def cube(n):
  return n**3

number = int(input("Enter number to find square root:"))
print("Square root of {1} is {0}".format(square(number), number))
print("Cube root of {num} is {num_cube}".format(num_cube=cube(number), num=number))
```

```
Enter number to find square root:5
Square root of 5 is 25
Cube root of 5 is 125
```

Check a number is +ve, -ve or 0

In [ ]:
```
number= int(input("Enter a number to check: "))
if number > 0:
  print("{} is a Positive number".format(number))
elif number < 0:
  print("{} is a Negetive number".format(number))
else:
  print("{} is simply a Zero.".format(number))
```

```
Enter a number to check: +3455
3455 is a Positive number
```

largest among 3 numbers

In [ ]:
```
a = int(input("A: "))
b = int(input("B: "))
c = int(input("C: "))

if a>b and a>c:
  print(f"{a} is a largest number.")
elif b>c:
  print(f"{b} is a largest number.")
else:
  print(f"{c} is a largest number.")
```

```
A: 3
B: 6
C: 9
9 is a largest number.
```

Multiplication table

In [ ]:
```
table = int(input("Enter which number table do you want: "))
upto= 10

for i in range(1, upto+1):
  print(f"{i} * {table} = {i*table}")
```

```
Enter which number table do you want: 2
1 * 2 = 2
2 * 2 = 4
3 * 2 = 6
4 * 2 = 8
5 * 2 = 10
6 * 2 = 12
7 * 2 = 14
8 * 2 = 16
9 * 2 = 18
10 * 2 = 20
```

Simple calculator

In [ ]:

```
operation = input("Enter calculator symbol + - * / // %:")

def add(num):
    result = 0
    for i in num:
        result += i
    return result
def sub(num):
    return num[0] - num[1]
def mul(num):
    result = 1
    for i in num:
        result*= i
    return result
def div(num):
    return num[0] // num[1]
def fdiv(num):
    return num[0]/num[1]
def mod(num):
    return num[0]%num[1]

if operation == '+' or operation == '-' or operation == '*' or operation =='/' or operation =='//' or operation =='%':
    operands = list(map(int, input("Enter the numbers seperated by space:").split(" ")))
    if operation =='+':
        print("Addition:", add(operands))
    elif operation == '-':
        print('Subtraction:', sub(operands))
    elif operation =='*':
        print('Multiplication:', mul(operands))
    elif operation =='//':
        print('Division:', div(operands))
    elif operation =='/':
        print("Floor division:", fdiv(operands))
    else:
        print('Modulus:', mod(operands))
else:
    print("Enter invalid operation...")
```

```
Enter calculator symbol + - * / // %:-
Enter the numbers seperated by space:456 56
Subtraction: 400
```

Swap two variables

In [ ]:
```
a = 10
b = 20
print(f"Before swapping a has {a} and b has {b}")
a, b = b, a
print(f"After swapping a has {a} and b has {b}")
```

```
Before swapping a has 10 and b has 20
After swapping a has 20 and b has 10
```

In [ ]:
```
x = 100
y = 150
print(f"Before swapping x has {x} and y has {y}")
x = x+y
y = x-y
x = x-y
print(f"After swapping x has {x} and y has {y}")
```

```
Before swapping x has 100 and y has 150
After swapping x has 150 and y has 100
```

In [ ]:
```
p = 5
q = 15
print(f"Before swapping p has {p} and q has {q}")
p = p ^ q
q = p^q
p = p^ q
print(f"After swapping p has {p} and q has {q}")
```

```
Before swapping p has 5 and q has 15
After swapping p has 15 and q has 5
```

Calculate the area of triangle

In [ ]:
```
selection = int(input("Apply 1 if it is right angle triangle else 0: "))
def right_angled(b, h):
  area = 0.5 * b * h
  return area
def equilateral(side):
  area = ((3**0.5)/4) * side *side
  return area
if selection == 1:
  height = int(input("Enter Height of the triangle:"))
  base = int(input("Enter Base of the triangle:"))
  print(f"Area of right angled triangle is: {right_angled(base, height)}")
else:
  side = int(input("Enter side of the Equilateral triangle: "))
  print("Area of Equilateral triangle is {}".format(equilateral(side)))
```

```
Enter base of the triangle:5
Enter height of the triangle:10
Area of triangle is 25.0
```

In [ ]:
```
try:
  base = int(input("Enter base of the triangle:"))
  height = int(input("Enter height of the triangle:"))
  area = 0.5 * base * height
except:
  area = ((3**0.5)/4) * base *base
finally:
  print(f"Area of triangle is {area}")
```

Solve Quadratic equation

In [ ]:
```
a = int(input())
b = int(input())
c = int(input())

if a == 0:
  print("Invalid Quadrilateral equation.")
else:
  discriminant = b * b - 4 * a * c
  square_dis = discriminant **0.5
  denominator = 2*a
  if discriminant > 0:
    # If b*b > 4*a*c, then roots are real and different; roots of x2 - 7x - 12 are 3 and 4
    print("Real and different roots")
    print("Roots: ", -b-square_dis/denominator, " and ", -b+square_dis/denominator )


  elif discriminant == 0:
    # b*b == 4*a*c, then roots are real and both roots are same; roots of x2 - 2x + 1 are 1 and 1
    print("Real and same roots")
    print("Roots are: ", -b/denominator)

  else:
    # If b*b < 4*a*c, then roots are complex; x2 + x + 1 roots are -0.5 + i1.73205 and -0.5 - i1.73205
    print("Complex roots")
    print("Roots are: ", -b/denominator,"+i and ", -b/denominator,"-i")
```

```
1
1
1
Complex roots
Roots are:  -0.5 +i and  -0.5 -i11
```

convert Kilometers to miles

In [ ]:
```
#1 kilometer = 0.6215 miles
kilometers = float(input("Enter kilometer to convert into miles: "))
miles = kilometers / 1.609
print(f"{kilometers} kilometers are equal to {miles} miles")
```

```
Enter kilometer to convert into miles: 1
1.0 kilometers are equal to 0.6215040397762586 miles
```

In [ ]:
```
##1mile = 1.609 kilometers
miles = float(input("Enter miles to convert into kilometers: "))
kilometers = miles * 1.609
print(f"{miles} miles are equal to {kilometers} kilometers")
```

```
Enter miles to convert into kilometers: 5
5.0 miles are equal to 8.045 kilometers
```

Convert Celsius to Fahrenheit

In [ ]:
```
#(C*1.8) + 32 = F

celsius = float(input("Enter celsius to convert: "))
fahrenheit = (celsius * 1.8) + 32
print(f"Fahrenheit : {round(fahrenheit, 2)}")
```

```
Enter celsius to convert of Fahrenheit: 37
Fahrenheit : 98.6
```

In [ ]:
```
# C = (F-32)/1.8

fahrenheit = float(input("Enter Fahrenhei: "))
celsius = (fahrenheit - 32) / 1.8
print(f"Celsius : {round(celsius, 2)}")
```

```
Enter Fahrenhei: 98
Celsius : 36.67
```

Check a number is Even or Odd

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

def isEven(n):
  return n%2 == 0
  # return 1 if n%2 == 0 else 0
  # return True if n%2 == 0 else False

if isEven(num):
  print("Even number!")
else:
  print("Odd number!")
```

```
Enter a number:2
Even number!
```

Check Leap Year or not

In [ ]:
```
year = int(input("Enter year: "))

def leap(year):
  return year%4==0
print("Leap Year" if leap(year) else "Non leap year")
```

```
Enter year: 2023
Non leap year
```

Sum of natural numbers

Sum of natural numbers

In [ ]:
```
number = int(input('Number upto: '))
sum = 0
for i in range(number+1):
  sum += i
print("Sum:", sum)
```

```
Number upto: 15
Sum:  120
```

In [ ]:
```
number = int(input('Enter number upto: '))
sum = 0
while number > 0:
  sum += number
  number-=1
print('Sum:', sum)
```

```
Enter number upto: 15
Sum: 120
```

In [ ]:
```
number = int(input('Enter number:'))

def sum(n):
  if n == 1:
    return 1
  else:
    return n + sum(n-1)
print("sum of {} natural numbers are: {}".format(number, sum(number)))
```

```
Enter number:15
sum of 15 natural numbers are: 120
```

Factorial of a number

In [ ]:
```
number = int(input('Enter number: '))
factorial = 1
for i in range(1, number+1):
  factorial *= i
print(f"factorial of {number} is {factorial}")
```

```
Enter number: 5
factorial of 5 is 120
```

In [ ]:
```
def factorial(n):
  # return 1 if n ==0 else n*factorial(n-1)
  if n == 1:
    return 1
  else:
    return n * factorial(n-1)

if __name__ == "__main__":
  number = int(input())
  print(factorial(number))
```

```
5
120
```

Factors of a number

In [ ]:
```
number = int(input("number:"))

for i in range(1,number+1):
    if number%i== 0:
        print(i, end= ' ')
```

```
number:18
1 2 3 6 9 18
```

Prime number

In [ ]:
```
number = int(input("Any number:"))
def isPrime(number):
  for i in range(2, (number//2)+1):
    if number%i == 0:
      return False
      break
  return True

if isPrime(number):
  print("{} is a prime number".format(number))
else:
  print("{} is not a prime number".format(number))
```

```
Any number:21
21 is not a prime number
```

Prime number in an interval

In [ ]:
```
start = int(input("Enter start:"))
stop = int(input("Enter stop:"))

def isPrime(n):
  for j in range(2, (n//2)+1):
    if i%j == 0:
      return False
  return True

for i in range(start, stop+1):
  if isPrime(i):
    print(i, end=" ")
```

```
Enter start:2
Enter stop:100
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

Fabinacci series

In [ ]:
```
upto = int(input('Fabinacci sequence upto: '))

print("Fabinacci sequence:", end= " ")
n = 0
m = 1
if upto ==0:
  print(n)
elif upto == 1:
  print(n,m)
else:
  print(n, m, end=' ')
  for i in range(2, upto+1):
    o = m
    m = m + n
    n = o
    print(m, end = ' ')
```

```
Fabinacci sequence upto: 1
Fabinacci sequence: 0 1 1
```

In [ ]:
```
num = int(input("Enter the number: "))
def fabbinocci(n):
    if n < 2:
        return n
    else:
        return fabbinocci(n-1)+fabbinocci(n-2)
print("Fabbinocci sequence :", end =" ")
for i in range(num):
    print(fabbinocci(i), end=" ")
```

```
Enter the number: 9
Fabbinocci sequence : 0 1 1 2 3 5 8 13 21
```

Armstrong number

In [ ]:
```
number = int(input())
copy = number
armstrong = 0
count = len(str(number))

while number >0:
  remainder = number % 10
  armstrong += remainder**count
  number //= 10

if armstrong == copy:
  print(f"{copy} is an Armstrong number.")
else:
  print("not an armstrong number")
```

```
153
153 is an Armstrong number.
```

Armstrong number in an interval

In [ ]:
```
start = int(input("Enter first number: "))
stop = int(input('Enter final number:'))

def isArmstrong(n):
  str_n = str(n)
  count = len(str_n)
  armstrong = 0
  for j in str_n:
    armstrong += int(j)**count
  if armstrong == n:
    return n
for i in range(start, stop+1):
  if isArmstrong(i):
    print(i, end = " ")
```

```
Enter first number: 1
Enter final number:500
1 2 3 4 5 6 7 8 9 153 370 371 407
```

Power of 2 by anonumous function

In [ ]:
```
number= int(input("Enter number"))
a = list(map(lambda x: 2**x, range(number+1)))
for i in range(number+1):
  print(f'2 power {i} is {a[i]}')
```

```
Enter number5
2 power 0 is 1
2 power 1 is 2
2 power 2 is 4
2 power 3 is 8
2 power 4 is 16
2 power 5 is 32
```

Find numbers divisible by another number

In [ ]:
```
number = int(input("Enter number: "))
upto = int(input('enter a number upto we have to check: '))
divisible = list(filter(lambda y: y%number ==0, list(range(1,upto+1))))
print(f"The list of numbers which are divisible by {number} are {divisible}")
```

```
Enter number: 5
enter a number upto we have to check: 80
The list of numbers which are divisible by 5 are [0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80]
```

Find HCF or GCD

In [ ]:
```
hcf1 = int(input('HCf_1:'))
hcf2 = int(input('HCF_2:'))

def gcd(hcf1, hcf2):
    small = hcf1 if hcf2 > hcf1 else hcf2
    for i in range(small+1,1,-1):
        if ((hcf1 % i == 0) and (hcf2 % i == 0)):
            return i

print(gcd(hcf1, hcf2))
```

```
HCf_1:24
HCF_2:54
6
```

In [ ]:
```
nums = [int(x) for x in list(input().split(','))]

def gcd(nums):
    nums = sorted(nums)
    for j in range(nums[0],1,-1):
        boolean = False
        for i in nums:
            if i%j ==0:
                boolean = True
            else:
                boolean = False
        if boolean== True:
            return j
print(gcd(nums))
```

```
10,5,65,40,20
5
```

In [ ]:
```
##Euclidean algorithm
n = 27
m = 24
def Euclidean(m, n):
    while n:
        m,n = n, m%n
    return m
print(Euclidean(m,n))
```

```
3
```

LCM

In [ ]:
```
m = int(input('Primary:'))
n = int(input("Second:"))
large = ref = m if m>n else n
small = n if m>n else m
count = 2
while large % small != 0:
    large = ref
    large *= count
    count +=1
print("LCM", large)
```

```
Primary:4
Second:6
LCM 12
```

In [ ]:
```
m = m1 = int(input('Primary:'))
n = n1 = int(input("Second:"))
while n:
    m, n = n, m%n
lcm = (m1*n1)//m
print("LCM:", lcm)
```

```
Primary:4
Second:6
LCM: 12
```

Reverse a Number

In [ ]:
```
num = 12345
rev_num = 0
while num > 0:
    rem = num % 10
    rev_num = rem + (rev_num * 10)
    num //=10
print(rev_num)
```

```
54321
```

In [ ]:
```
num = 12345
print(int(str(num)[::-1]))
```

```
54321
```

Reverse a Number

In [ ]:
```
num = 12345
rev_num = 0
while num > 0:
    rem = num % 10
    rev_num = rem + (rev_num * 10)
    num //=10
print(rev_num)
```

54321

In [ ]:
```
num = 12345
print(int(str(num)[::-1]))
```

54321

Count number of digits in a number

In [ ]:
```
num = 123
count = 0
while num:
    num//=10
    count +=1
print(count)
```

3

In [ ]:
```
num=1234
print(len(str(num)))
```

4

Compute the power of a number

In [ ]:
```
num = 4
pwr = 2
def power(num, pwr):
    return num ** pwr
print(power(num,pwr))
```

16

In [ ]:
```
print(int(pow(4,3)))
```

64

Decimal to Binary, Octal Hexadecimal

Decimal to binary

In [ ]:
```
decimal = int(input("Decimal: "))

print('With functions:', bin(decimal).replace("0b",""))

binary = ''
while decimal>0:
  binary = str(decimal%2) + binary
  decimal //=2
print(f'binary is {binary} without functions')
```

```
Decimal: 15
With functions: 1111
 binary is 1111 without functions
```

Decimal to octal

In [ ]:
```
decimal = int(input("Decimal:"))

print("with functions", oct(decimal).replace("0o", ''))

octal = ''
while decimal>0:
  octal = str(decimal%8) + octal
  decimal //= 8
print('Octal without functions:', octal)
```

```
Decimal:12
with functions 14
Octal without functions: 14
```

Decimal to hexadecimal

In [ ]:
```
decimal = int(input('Decimal: '))
print('with functions:', hex(decimal).replace('0x', ''))
hexadecimal = ''
hexa = '0123456789abcdef'
while decimal>0:
  hexadecimal = hexa[decimal%16] + hexadecimal
  decimal //= 16
print("Hexadecimal without function:", hexadecimal)
```

```
Decimal: 15
with functions: f
Hexadecimal without function: f
```

Binary to decimal

In [ ]:
```
binary = input("Binary:")
print("Decimal:", int(binary,2))
decimal = 0
for i in binary:
  decimal = decimal * 2 + int(i)
print('Decimal without function:', decimal)
```

```
Binary:1011
Decimal: 11
Decimal without function: 11
```

Octal to decimal

In [ ]:
```
octal = input("Octal:")
print('Decimal:', int(octal, 8))

decimal = 0
for i in octal:
  decimal = decimal*8 + int(i)
print("Decimal without function:", decimal)
```

```
Octal:17
Decimal: 15
Decimal without function: 15
```

Hexadecimal to decimal

In [ ]:
```
hexa = input("Hexadecimal:")
print("decimal:", int(hexa, 16))

decimal = 0
hex_map = {'0':0, '1':1, '2':2, '3':3, '4':4,'5':5,'6':6, '7':7, '8':8, '9':9, 'a':10, 'b':11, 'c':12, 'd':13, 'e':14, 'f':
for i in hexa:
  decimal = 16*decimal + hex_map[i]
print("deciamal without function:",decimal)
```

```
Hexadecimal:51
decimal: 81
deciamal without function: 81
```

Decimal to Binary with Reccursion

In [ ]:
```
dec = int(input("Enter Decimal number:"))
def decToBin(dec):
    if dec>0:
        decToBin(dec//2)
    print(dec%2, end = "")
decToBin(dec)
```

```
Enter Decimal number:9
01001
```

Creating pyramid patterns

In [ ]:
```
n = 5
for i in range(n):
    for j in range(n):
        print("* ", end="")
    print()
```

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

In [ ]:
```
n = 5
for i in range(n):
    for j in range(i+1):
        print("* ", end="")
    print()
```

```
*
* *
* * *
* * * *
* * * * *
```

In [ ]:
```
n = 5
for i in range(n):
    for j in range(n-i):
        print("* ", end="")
    print()
```

```
* * * * *
* * * *
* * *
* *
*
```

In [ ]:
```
n = 5
for i in range(n):
    for j in range(n-i-1):
        print("  ", end="")
    for k in range(i+1):
        print("* ", end="")
    print()
```

```
      *
    * *
  * * *
* * * *
* * * * *
```

In [ ]:
```
n = 5
for i in range(n):
    for j in range(i):
        print("  ", end= "")
    for k in range(n-i):
        print('* ',end="")
    print()
```

```
* * * * *
  * * * *
    * * *
      * *
        *
```

In [ ]:
```
n = 5
for i in range(n):
    for j in range(n-i-1):
        print(" ", end = "")
    for k in range(i+1):
        print("* ", end="")
    print()
```

```
    *
   * *
  * * *
 * * * *
* * * * *
```

In [ ]:
```
n = 5
for i in range(n):
    for j in range(i):
        print(" ", end="")
    for k in range(n-i):
        print("* ", end = "")
    print()
```

```
* * * * *
 * * * *
  * * *
   * *
    *
```

In [ ]:
```
n = 5
for i in range(n):
    for j in range(n-i-1):
        print(" ", end="")
    for k in range(i+1):
        print("* ", end = "")
    print()
for i in range(n):
    for j in range(i+1):
        print(' ', end = '')
    for k in range(n-i-1):
        print("* ", end = '')
    print()
```

```
    *
   * *
  * * *
 * * * *
* * * * *
 * * * *
  * * *
   * *
    *
```

In [ ]:
```
n = 5
for i in range(n-1):
    for j in range(i):
        print(" ", end = '')
    for k in range(n-i):
        print("* ", end = '')
    print()
for i in range(n):
    for j in range(n-i-1):
        print(" ", end = "")
    for k in range(i+1):
        print('* ', end = '')
    print()
```

```
* * * * *
 * * * *
  * * *
   * *
    *
   * *
  * * *
 * * * *
* * * * *
```

In [ ]:
```
n = 5
for i in range(n-1):
    for j in range(i):
        print(' ', end = "")
    for k in range(n-i):
        if k ==0 or k == n-i-1:
            print('* ', end = '')
        else:
            print("  ",end = '')
    print()
for i in range(n):
    for j in range(n-i-1):
        print(' ', end = '')
    for k in range(i+1):
        if k == 0 or k == i:
            print("* ", end='')
        else:
            print('  ', end='')
    print()
```

```
*       *
 *     *
  *   *
   * *
    *
   * *
  *   *
 *     *
*       *
```

In [ ]:
```
n = 5
for i in range(n):
    for j in range(n):
        if i == 0 or j ==0 or i == n-1 or j == n-1:
            print('* ',end = '')
        else:
            print('  ', end = '')
    print()
```

```
* * * * *
*       *
*       *
*       *
* * * * *
```

In [ ]:
```
n = 5
for i in range(n+1):
    for j in range(i):
        if j == 0 or i ==n or i == j+1:
            print('* ', end = '')
        else:
            print('  ',end = '')
    print()
```

```
*
* *
*   *
*     *
* * * * *
```

In [ ]:
```
n = 5
for i in range(n):
    for j in range(i):
        print('  ',end ='')
    for k in range(n-i):
        if i == 0 or k == 0 or k ==n-i-1:
            print('* ',end='')
        else:
            print('  ',end = '')
    print()
```

```
* * * * *
  *     *
    *   *
      * *
        *
```

In [ ]:
```
n = 5
for i in range(n):
    for j in range(n-i-1):
        print(' ',end='')
    for k in range(i+1):
        if k == 0 or k ==i or i ==n-1:
            print('* ', end = '')
        else:
            print('  ',end='')
    print()
```

```
    *
   * *
  *   *
 *     *
* * * * *
```

In [ ]:
```
n = 5
for i in range(n):
    for j in range(i):
        print(' ',end='')
    for k in range(n-i):
        if i == 0 or k == 0 or k == n-i-1:
            print('* ', end='')
        else:
            print('  ',end='')
    print()
```

```
* * * * *
 *     *
  *   *
   * *
    *
```

In [ ]:
```
n = 5
for i in range(n-1):
    for j in range(n-i-1):
        print(" ",end='')
    for k in range(i+1):
        if k == 0 or k == i:
            print("* ",end='')
        else:
            print('  ',end='')
    print()
for i in range(n):
    for j in range(i):
        print(' ',end='')
    for k in range(n-i):
        if k == 0 or k ==n-i-1:
            print('* ',end='')
        else:
            print('  ',end='')
    print()
```

```
    *
   * *
  *   *
 *     *
*       *
 *     *
  *   *
   * *
    *
```

Check if list is empty

In [ ]:
```
list1 = [21,3,9,54,3,24]
list2=[]
if len(list1) == 0:
    print('List1 is empty')
else:
    print(f'List1 has {len(list1)} elements')

if not list2:
    print('List is Empty')
```

```
List1 has 6 elements
List is Empty
```

Get the last Element of the list

In [ ]:
```
my_list = [1,2,3,4,5,6]
print(my_list[len(my_list)-1])
print(my_list[-1])
```

```
6
6
```

Index of list using For loop

In [ ]:
```
num = ['zero','one','two','three']
for i in num:
    ind = num.index(i)
    print(ind, 'index of', i)
```

```
0 index of zero
1 index of one
2 index of two
3 index of three
```

In [ ]:
```
num = ['zero','one','two','three']
for i,j in enumerate(num):
    print(i,'index of', j)
```

```
0 index of zero
1 index of one
2 index of two
3 index of three
```

Count the occurance of an Item in a List

In [ ]:
```
List = [1,2,3,4,2,4,1,4,5,2,6,72,7,8]
search = 2
print(List.count(search))
```

```
3
```

In [ ]:
```
search = 1
count = 0
for i in List:
    if search == i:
        count +=1
print(count)
```

```
2
```

Slice lists

In [ ]:
```
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Get the first three items
slice1 = my_list[:3]
print(slice1)  # Output: [1, 2, 3]

# Get the last three items
slice2 = my_list[-3:]
print(slice2)  # Output: [8, 9, 10]

# Get every other item starting from the second item
slice3 = my_list[1::2]
print(slice3)  # Output: [2, 4, 6, 8, 10]

# Reverse the list
slice4 = my_list[::-1]
print(slice4)  # Output: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

```
[1, 2, 3]
[8, 9, 10]
[2, 4, 6, 8, 10]
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Concatenate Two lists

In [ ]:
```
list1 = [1,2,3,4]
list2 = [5,6,7,8]

print(list1+list2)

list1.extend(list2)
print(list1)
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
[1, 2, 3, 4, 5, 6, 7, 8]
```

In [ ]:
```
str1 = "Hello"
str2 = "World"

total = '_'.join((str1,str2))
print(total)
```

```
Hello_World
```

Concatenate Two lists

In [ ]:
```
list1 = [1,2,3,4]
list2 = [5,6,7,8]

print(list1+list2)

list1.extend(list2)
print(list1)
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
[1, 2, 3, 4, 5, 6, 7, 8]
```

In [ ]:
```
str1 = "Hello"
str2 = "World"

total = '_'.join((str1,str2))
print(total)
```

```
Hello_World
```

Split a list into evenly sized chuncks

In [ ]:
```
my_list= [1,2,3,4,5,6,7,8,9,10,11,12,13,14]
size = 2
lists= [my_list[i:i+size] for i in range(0,len(my_list),size)]
print(lists)
```

```
[[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12], [13, 14]]
```

Flattened list from nested list

In [ ]:
```
nested_list = [0.1,0.2,[3.1,4.2,[5,6,[70,80,[900,1000]]]]]
def flattened(num):
    flattened_list = []
    for i in num:
        if isinstance(i, list):
            flattened_list.extend(flattened(i))
        else:
            flattened_list.append(i)
    return flattened_list
print(flattened(nested_list))
```

```
[0.1, 0.2, 3.1, 4.2, 5, 6, 70, 80, 900, 1000]
```

Iterate through two lists in parallel

In [ ]:
```
my_list = [1,2,3,4]
my_list2 = my_list[::-1]
for i, j in zip(my_list, my_list2):
    print(i,j)
```

```
1 4
2 3
3 2
4 1
```

Remove duplicate element from a list

In [ ]:
```
my_list = [1,1,2,7,3,4,4,5,5,3,3,5,4,1,6]
print(list(set(my_list)))
```

```
[1, 2, 3, 4, 5, 6, 7]
```

In [ ]:
```
my_list = [1,1,2,7,3,4,4,5,5,3,3,5,4,1,6]
new_list = []
for i in my_list:
    if i not in new_list:
        new_list.append(i)
new_list.sort()
print(new_list)
```

```
[1, 2, 3, 4, 5, 6, 7]
```

Del, remove,and pop on a list

In [ ]:
```
my_list = [1,2,3,4,5,6,7,8,9]
print('Actual list:', my_list)
del my_list[1] #delete the 1st index item
print('Del index 1:',my_list)
my_list.remove(3) #remove element which we passed as an argument
print(my_list)
popped_item = my_list.pop(5) #pop item in 5th index
print(f'{popped_item} is fetched which locates as 5th index of {my_list}')
```

```
Actual list: [1, 2, 3, 4, 5, 6, 7, 8, 9]
Del index 1: [1, 3, 4, 5, 6, 7, 8, 9]
[1, 4, 5, 6, 7, 8, 9]
8 is fetched which locates as 5th index of [1, 4, 5, 6, 7, 9]
```

Add 2 matrices

In [ ]:
```
m,n= map(int,input('Enter matrix structure in M*N:').split())
print('Enter the elements:')
matrix1 = []
for i in range(m):
    row = []
    for j in range(n):
        k = int(input())
        row.append(k)
    matrix1.append(row)

print('Enter elements of second matrix:')
matrix2 = [[int(input()) for i in range(m)] for j in range(n)]

major = []
for i in range(m):
    minor = []
    for j in range(n):
        k = matrix1[i][j]+matrix2[i][j]
        minor.append(k)
    major.append(minor)

for minor in major:
    print(minor)
for i in range(m):
    for j in range(n):
        print(major[i][j], end=' ')
    print()
```

```
Enter matrix structure in M*N:2 2
Enter the elements:
2
4
6
8
Enter elements of second matrix:
8
6
4
2
[10, 10]
[10, 10]
10 10
10 10
```

Transpose a Matrix

In [ ]:
```
# Example matrix
matrix = [    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
transpose = []
for i in range(len(matrix[0])):
    row = []
    for j in range(len(matrix)):
        row.append(matrix[j][i])
    transpose.append(row)

# Using the zip function
# transpose = list(map(list, zip(*matrix)))

print("Transposed matrix:")
for row in transpose:
    print(row)
```

```
Transposed matrix:
[1, 4, 7]
[2, 5, 8]
[3, 6, 9]
```

Multiply two matrices

In [ ]:
```
matrix = [[2, 2, 2],
          [4, 5, 6],
          [7, 8, 9]]
matrix2 = [[1, 2, 3],
           [4, 5, 6],
           [7, 8, 9]]
matrixB = []
for i in range(len(matrix[0])):
    row = []
    for j in range(len(matrix)):
        k = matrix[j][i]
        row.append(k)
    matrixB.append(row)

mul = [[0 for i in range(len(matrixB[0]))]for j in range(len(matrix))]

for i in range(len(matrix)):
    for j in range(len(matrixB[0])):
        for k in range(len(matrixB)):
            mul[i][j]= matrix[i][k]*matrixB[k][j]
for i in mul:
    print(i)
```

```
[4, 12, 18]
[12, 36, 54]
[18, 54, 81]
```

Parse a string to a float or int

In [ ]:
```
string_in = '3.14'
# string_in = '3.14pi'
try:
    float_in = float(string_in)
    print(float_in)
except ValueError as e:
    print("Exception raised on ", e)
```

```
Exception raised on  could not convert string to float: '3.14pi'
```

Create a Long Multiline String

In [ ]:
```
long_string = '''This is a long multiline string. It spans across multiple lines.
You can include line breaks and formatting within the string.

Here's an example of a bulleted list:
- Item 1
- Item 2

Just make sure to enclose the string in triple quotes.'''

print(long_string)
```

```
This is a long multiline string. It spans across multiple lines.
You can include line breaks and formatting within the string.

Here's an example of a bulleted list:
- Item 1
- Item 2

Just make sure to enclose the string in triple quotes.
```

Ascii value of a character

In [ ]:
```
character = input("Enter a character:")
ascii_value = ord(character)
chr_value = chr(ascii_value)
print(f'Unicode value of {chr_value} is {ascii_value}')
```

```
Enter a character:7
Unicode value of 7 is 55
```

String is Palindrome or Not

In [ ]:
```
string = input("Enter String: ")

if string == string[::-1]:
    print("string is palindrome")
else:
    print("Not pallindrome")
```

```
Enter String: madam
string is palindrome
```

In [ ]:
```
string = input("Enter string: ")
rev = ''
for i in string:
    rev = i + rev
if string == rev:
    print("Pallindrome")
else:
    print("Not pallindrome")
```

```
Enter string: haiah
Pallindrome
```

Count the number of each vowel

In [ ]:
```
words = input().upper()
A = E = I = O = U = 0

for i in words:
    if i =='A':
        A += 1
    elif i =='E':
        E += 1
    elif i =='I':
        I += 1
    elif i =='O':
        O += 1
    elif i =='U':
        U += 1
    else:
        pass

print('A:', A , 'E: ', E,' I: ',I,' O: ',O,' U: ',U)
```

```
Sai Praveen
A: 2 E:  2  I:  1  O:  0  U:  0
```

Get Substing of a String

In [ ]:
```
s = 'Ande Sai Praveen'
s1 = s[4:8]
s2= s[5:]
s3 = s[:8:-1]
print('S1:', s1)
print('S2:', s2)
print('S3:', s3)
```

```
S1:  Sai
S2: Sai Praveen
S3: neevarP
```

Sort words with alphabetical order

In [ ]:
```
alphabets = ['hai','i', 'had', 'many', 'hopes', 'on','ttec', 'company', 'for', 'that', 'i', 'had', 'prepared','very', 'much
sorted_one = sorted(alphabets)
print(sorted_one)
```

```
['communication', 'company', 'didn"t', 'due', 'for', 'get', 'had', 'had', 'hai', 'having', 'hopes', 'i', 'i', 'i', 'it', 'man
y', 'much.', 'on', 'pooor', 'prepared', 'skills.', 'still', 'that', 'to', 'ttec', 'very']
```

In [ ]:
```
words = ['apple', 'banana', 'cherry', 'date', 'elderberry']
for i in range(len(words)):
    for j in range(len(words)-i-1):
        if words[j]>words[j+1]:
            words[j], words[j+1]= words[j+1], words[j]
print(words)
```

```
['apple', 'banana', 'cherry', 'date', 'elderberry']
```

Trim whitespace from a string

In [ ]:
```
string = 'Hai Sai Praveen'
trim = ''
for i in string:
    if i != " ":
        trim = trim+i
print(trim)
```

```
HaiSaiPraveen
```

In [ ]:
```
string = '            Hai sai Praveen          '
print(string.strip())
print(string.rstrip())
print(string.lstrip())
```

```
Hai sai Praveen
            Hai sai Praveen
Hai sai Praveen
```

convert bytes to a string

In [ ]:
```
byte_code = b'Hai Praveen'
decode_code = byte_code.decode()
print(decode_code)
```

```
Hai Praveen
```

Check if two strings are Anagram

In [ ]:
```
string1='listen'
string2 = 'silent'
string1 = string1.replace(" ","").lower()
string2 = string2.replace(" ","").lower()
if sorted(string1) == sorted(string2):
    print("Strings are Anagrams")
else:
    print("Not Anagrams")
```

```
Strings are Anagrams
```

Capitalise the First Character of a string

In [ ]:
```
text = "hai my name is praveen. now i am going to create a capitalised text."
words = text.split()
Text = [word[0].upper() + word[1:] for word in words]
text = " ".join(Text)
print(text)
```

```
Hai My Name Is Praveen. Now I Am Going To Create A Capitalised Text.
```

In [ ]:
```
name= 'sai praveen'
print(name[0].upper() + name[1:])
```

```
Sai praveen
```

In [ ]:
```
text = "hai my name is praveen. now i am going to create a capitalised text."
print(text.capitalize())
print(text.title())
```

```
Hai my name is praveen. now i am going to create a capitalised text.
Hai My Name Is Praveen. Now I Am Going To Create A Capitalised Text.
```

Check if a string is a Number(Float)

In [ ]:
```
def is_num(input):
    return isinstance(input, str)

def is_num1(input):
    return input.isdigit()

def is_num2(input):
    return input.isnumeric()

def is_num3(input):
    try:
        float(input)
        return True
    except:
        return False
print(is_num3('5'))
```

```
True
```

Count the number of occurance of a character in a string

In [ ]:
```
text = 'Sai praveen'.lower()
ch = 'a'.lower()
print(text.count(ch))
```

```
2
```

In [ ]:
```
text = 'Sai praveen'
ch = 'e'
count = 0
for i in text:
    if ch == i:
        count += 1
print(count)
```

```
2
```

Remove punctuation from a string

In [ ]:
```
import string
string_input = 'Hai! my name is Praveen: Im working in D.X.C. Technologies!'
new = str.maketrans('','',string.punctuation)
string_input = string_input.translate(new)
print(string_input)
```

```
Hai my name is Praveen Im working in DXC Technologies
```

In [ ]:
```
import string
string_input = 'Hai! my name is Praveen: Im working in D.X.C. Technologies!'
s = ''
for i in string_input:
    if i not in string.punctuation:
        s += i
print(s)
```

Illustrate different set operations

In [ ]:
```
a = {1,2,3,4,5,6}
b = {4,5,6,7,8,9}

union_set = a.union(b) #returns unique elements from both sets
union_set= a | b
print(union_set)

intersection_set = a.intersection(b) # return common elements in both sets
intersection_set=a & b
print(intersection_set)

difference_set = a.difference(b)
difference_set = a - b #returns elements present in A but not in B
print(difference_set)

s
ymmetric_difference_set = a.symmetric_difference(b)
symmetric_difference_set = a ^ b # returns the elements present in either of the sets but not in both
print(symmetric_difference_set)

subset_check = a.issubset(b)
subset_check = a <= b  #returns True if all the elements in the first set are present in the second
print(subset_check)

superset_check = b.issuperset(a)
superset_check = a >= b #returns True if all the elements in the second set are present in the first
print(superset_check)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
{4, 5, 6}
{1, 2, 3}
{1, 2, 3, 7, 8, 9}
False
False
```

Create nested dictionary

In [ ]:
```
Details = {'Name':'Sai Praveen Ande',
           'Age' :22,
           'Qualification': 'B.Tech',
           'Branch' : 'ECE',
           'Address':{'Door No':'1-96',
                      'Street':'Shivalayam Street',
                      'Village':'Chintaparru',
                      'State': 'Andhra Pradhesh',
                      'Pin':534250}}
print(Details)
```

```
{'Name': 'Sai Praveen Ande', 'Age': 22, 'Qualification': 'B.Tech', 'Branch': 'ECE', 'Address': {'Door No': '1-96', 'Street':
'Shivalayam Street', 'Village': 'Chintaparru', 'State': 'Andhra Pradhesh', 'Pin': 534250}}
```

Convert two lists into a Dictionary

In [ ]:
```
list1 = [1,2,3,4,5]
list2 = [1,4,9,16,25]

my_dict = {}
for index,value in enumerate(list1):
    my_dict[value] = list2[index]

my_dict2 = {k:v for k, v in zip(list2, list1) }

my_dict3 = dict(zip(list1, list2))

print(my_dict)
print(my_dict2)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
{1: 1, 4: 2, 9: 3, 16: 4, 25: 5}
```

Iterate over Dictionaries using for loop

In [ ]:
```
my_dict = {'A':'Append', 'B':'BaseClass', 'C':'Cassendra','D':'Database','E':'Efficient'}
for item, value in my_dict.items():
    print(item, 'for', value)
for item in my_dict.keys():
    print(item, end = ' ')
print()
for value in my_dict.values():
    print(value, end = ' ')
```

```
A for Append
B for BaseClass
C for Cassendra
D for Database
E for Efficient
A B C D E
Append BaseClass Cassendra Database Efficient
```

Sort a dictionary by value

In [ ]:
```
my_dict = {2:"pineapple",3: "banana", 4:"orange", 11:'papaya'}

sorted_dict = dict(sorted(my_dict.items(), key=lambda x:x[1]))#sort values
print(sorted_dict)

print(dict(sorted(my_dict.items(), key = lambda x:x[1], reverse = True))) # sort values in reverse order

will_sorted = list(my_dict.items()) #covert dict to list
will_sorted.sort(key=lambda x: x[1])#sort the values
print({key:value for key,value in will_sorted}) #covert list to dict
```

```
{3: 'banana', 4: 'orange', 11: 'papaya', 2: 'pineapple'}
{2: 'pineapple', 11: 'papaya', 4: 'orange', 3: 'banana'}
{3: 'banana', 4: 'orange', 11: 'papaya', 2: 'pineapple'}
```

Check if a key is present in a dictionary

In [ ]:
```
my_dict = {'A':'Append', 'B':'BaseClass', 'C':'Cassendra','D':'Database','E':'Efficient'}
ele = 'F'
if ele in my_dict.keys():
    print(f'{ele} Key exist with {my_dict[ele]}')
else:
    print('Element not found')
```

```
Element not found
```

In [ ]:
```
ele = 'C'
if my_dict.get(ele) is not None:
    print(f'{ele} is present in dictionary with {my_dict.get(ele)}')
else:
    print(f'{ele} not in dictionary')
```

```
C is present in dictionary with Cassendra
```

Delete an Item from a Dictionary

In [ ]:
```
my_dict = {'a':'Append', 'b':'BredthFirstSearch', 'c':'concat', 'd':'Developing'}
del my_dict['b']
my_dict.pop('c')
print(my_dict)
```

```
{'a': 'Append', 'd': 'Developing'}
```

Merge Two Dictionaries

In [ ]:
```
one = {1:'sai Praveen', 2:'vamsi', 3: 'nazir', 4: 'jagadhish'}
two = {3:'Raja Blessing', 6:'peta subhramanyam', 7:'Pavan Putram'}
one.update(two)
merged = {**one, **two}
print(one)
print(merged)
```

```
{1: 'sai Praveen', 2: 'vamsi', 3: 'Raja Blessing', 4: 'jagadhish', 6: 'peta subhramanyam', 7: 'Pavan Putram'}
{1: 'sai Praveen', 2: 'vamsi', 3: 'Raja Blessing', 4: 'jagadhish', 6: 'peta subhramanyam', 7: 'Pavan Putram'}
```

Print output without a new line

In [ ]:
```
print("I am a python", end = " ")
print("Programmer!")
```

```
I am a pyton Programmer!
```

In [ ]:
```
import sys
sys.stdout.write("Hello ")
sys.stdout.write("Praveen!")
```

```
Hello Praveen!
```

Get the class name of an Instance

In [ ]:
```
class MyClass:
    pass
obj = MyClass()

class_name = type(obj).__name__
class_Name = obj.__class__.__name__

print(class_name)
print(class_Name)
```

```
MyClass
MyClass
```

Represent Enum

In [ ]:
```
from enum import Enum

class Color(Enum): #Define Color Enum
    RED = 1
    GREEN = 2
    YELLOW = 3

print(Color.RED)  #Accessing Enum member

print(Color.GREEN.value)  #print Enum value

for colors in Color:    #accessing Enum through iteration
    print(colors)

print(Color.RED == Color.RED) #Comparision Enum values
print(Color.RED == Color.GREEN)
```

```
Color.RED
2
Color.RED
Color.GREEN
Color.YELLOW
True
False
```

Generate a random number random lib

In [ ]:
```
import random as r

print(r.randint(1,90))       #returns an integer between an interval
print(r.random())            #returns a floating number from 0 to 1
print(r.uniform(3,6))        #return a floating number in an interval
print(r.choice("PRAVEN"))    #returns an element in a sequence
print(r.choices("ANDE BHASKARA NAGA SAI PRAVEEN"))   #returns a list of elements in a sequence
```

```
8
1
3.204923201970927
E
['N']
```

Randomly Select an element from the list

In [ ]:
```
import random
lis = [1,2,3,4,5,6,7,8]
print(random.choice(lis))
```

```
1
```

Shuffle deck of cards

In [ ]:
```
import itertools, random
suits = ['Hearts', 'Diamonds', 'Clubs', 'Spades']
ranks= ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'K', 'Q']
cords = [(suit, rank) for suit in suits for rank in ranks]
# chords = list(itertools.product(suits, ranks))
random.shuffle(cords)
for i in range(5):
    print(cords[i][0],'of', cords[i][1])
```

```
Spades of 4
Spades of 10
Diamonds of 10
Spades of 7
Clubs of 2
```

Compute all the permutations of the string

In [ ]:
```
from itertools import permutations
text = 'abc'
letters = permutations(text)
words = [''.join(letter) for letter in letters]
for word in words:
    print(word, end = ' ')
```

```
abc acb bac bca cab cba
```

Copy a File

In [ ]:
```
from shutil import copy
src_file = 'path/source/file.txt'
dst_file = 'path/destination/file.txt'
copy(src_file, dst_file)
```

In [ ]:
```
with('path/source/file.txt', 'rb') as src_file:
    with('path/destination/file.txt', 'wb') as dst_file:
        while True:
            data = src_file.read(1024)
            if not data:
                break
            dst_file.write(data)
```

Append to a File

In [ ]:
```
File_path = 'path/source/file.txt'
with open(File_path, 'a') as file:
    content = "This is the Text that I should have to append to file.txt file.\n"
    file.write(content)
```

Read a File Line by line into a list

In [ ]:
```
path_file = "path/source/file.txt"

with open(path_file, 'r') as file:
    lines = [line.strip() for line in file.readlines()]
print(lines)
```

Get Line Count of a File

In [ ]:
```
file_path= 'path/source/file.txt'
with open(file_path, 'r') as file:
    count = 0
    for line in file:
        count += 1
print(count)
```

Get the file name from the file path

In [ ]:
```
import os
file_path = 'path/source/file.txt'
file_name = os.path.basename(file_path)
print(file_name)
```

```
file.txt
```

Extra extension from a file name

In [ ]:
```
import os
file_name = 'file.exe'
file_exe = os.path.splitext(file_name)[1]
print(file_exe)
```

```
.exe
```

Find All File with .txt Extension Present inside a Directory

In [ ]:
```
import os

files= 'path/source/'
for file in files:

    if os.path.splitext(files)[1] == '.txt':
        print(os.path.basename(file))
```

In [ ]:
```
from glob import glob
txt_files = glob("path/source/*.txt")

for file in txt_files:
    print(file)
```

Get the full path of the current working directory

In [ ]:
```
from os import getcwd
print(getcwd())
```

```
/content
```

Check the file size

In [ ]:
```
from os import path

file_path = 'path/to/source/file.txt'
print('File size:', path.getsize(file_path), 'bytes')
```

Display calender

In [ ]:
```
import calendar
yy = int(input('enter year: '))
month = int(input('Enter month 1-12: '))
print()
print(calendar.month(yy, month))
```

```
enter year: 2001
Enter month 1-12: 10

    October 2001
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

Convert String to dateTime

In [ ]:
```
from datetime import datetime
str_time = "22-10-2001 02:30:12"
Dtime = datetime.strptime(str_time, '%d-%m-%Y %H:%M:%S')
print(Dtime)
```

```
2001-10-22 02:30:12
```

Get file creation and modification date

In [ ]:
```
import os, datetime

file_path = 'path/to/source/file.txt'

creation_time = os.path.getctime(file_path)
creation_date = datetime.datetime.fromtimestamp(creation_time)
print(f'File is created at {creation_time} on {creation_date}')

modify_time = os.path.getmtime(file_path)
modify_date = datetime.datetime.fromtimestamp(modify_time)
print(f'File recently modified at {modify_time} on {modify_date}')
```

Create a countdown timer

In [ ]:
```
import time

def timer(minutes):
    seconds = minutes * 60
    while seconds>0:
        minute_remaining = seconds // 60
        second_remaining = seconds % 60
        print(f'{minute_remaining:02d} : {second_remaining:02d}')
        time.sleep(1)
        seconds -=1
    print("Time is UP!")
timer(1)
```

```
01 : 00
00 : 59
00 : 58
00 : 57
```

```
00 : 24
00 : 23
00 : 22
00 : 21
00 : 20
00 : 19
00 : 18
00 : 17
00 : 16
00 : 15
00 : 14
00 : 13
00 : 12
00 : 11
00 : 10
00 : 09
00 : 08
00 : 07
00 : 06
00 : 05
00 : 04
00 : 03
00 : 02
00 : 01
Time is UP!
```

Measure the Elapsed time

In [ ]:

```python
import time
start = time.time()

time.sleep(3)#delay for 3seconds

end = time.time()
elapsed_time = end -start

print(f'{elapsed_time:.2f}')
```

```
3.01
```

Merge mails

In [ ]:

```python
import email
import mailbox

# create a new mailbox file
merged_mailbox = mailbox.mbox('merged.mbox')

# list of email messages to merge
messages = ['message1.eml', 'message2.eml', 'message3.eml']

# loop through each message and add it to the merged mailbox
for msg_file in messages:
    with open(msg_file, 'r') as f:
        msg = email.message_from_file(f)
        merged_mailbox.add(msg)

# close the merged mailbox file
merged_mailbox.close()
```

size of an Image

In [ ]:
```
from PIL import Image

pic = Image.open("picture.jpg")
width, height = pic.size
print('Image resolution: ', width, "*", height)
```

Print color text to the terminal

In [ ]:
```
# install the colorama library first by running pip install colorama

from colorama import Back, Style, Fore
#Fore and Back has BLACK, RED, GREEN, YELLOW, BLUE, MAGENTA, CYAN, WHITE

# Style has NORMAL, BRIGHT, DIM, RESET_ALL

print(Fore.RED + 'Text is in RED color')
print(Back.LIGHTBLACK_EX + 'Text background is in light black')
print(Style.DIM + 'Text style is changed to DIM')
print(Style.RESET_ALL + "Text style can be r")
```

Find hash the file

In [ ]:
```
import hashlib

# open the file to hash
with open('example.txt', 'rb') as f:
    # read the contents of the file
    contents = f.read()

    # generate the hash value
    md5_hash = hashlib.md5(contents)
    sha256_hash = hashlib.sha256(contents)

    # print the hash values
    print("MD5 hash:", md5_hash.hexdigest())
    print("SHA256 hash:", sha256_hash.hexdigest())
```