Ejercicio 1

Crea una función que reciba los tres coeficientes a, b y c para resolver una ecuación de segundo grado. Muestra la solución por pantalla y ayúdate de la librería Math para acceder a la función raíz cuadrada.

In [22]:
```
import numpy as np
def solucion(a,b,c):
    x1 = (-b + np.sqrt((b**2)-(4*a*c)))/(2*a)
    x2 = (-b - np.sqrt((b**2)-(4*a*c)))/(2*a)
    return(x1,x2)
```

Por ejemplo tenemos:

$$2x^2 - 4x + 1 = 0$$

In [23]:
```
solucion(2,-4,1)
```

Out[23]:
```
(1.7071067811865475, 0.2928932188134524)
```

Ejercicio 2

Crea una función que lea una frase de teclado y nos diga si es o no un palíndromo (frase que se lee igual de izquierda a derecha o al revés como por ejemplo La ruta nos aportó otro paso natural.)

In [2]:
```
texto = str(input("Ingrese una frase: "))
texto = texto.replace(" ", "").lower()
inverso = texto[::-1]

def esPalindromo(texto):
    if texto == inverso:
        return "Es un Palindromo"
    else:
        return "No es un Palindromo"

esPalindromo(texto)
```

```
Ingrese una frase: La ruta nos aporto otro paso natural
```

Out[2]:
```
'Es un Palindromo'
```

Ejercicio 3

Crea un diccionario que tenga por claves los números del 1 al 10 y como valores sus raíces cuadradas

In [1]:
```
cuadrado = {}
for i in range(1,11):
    cuadrado[i]=i**2

print(cuadrado)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

Ejercicio 4

Crea un diccionario que tenga como claves las letras del alfabeto castellano y como valores los símbolos del código morse (los tienes todos en la Wikipedia).

In [46]:
```
alfabeto = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ"
simbolos_morse = ["· –", "– · · ·", "– · – ·", "– · ·","·","· · – ·","– – ·","· · · ·","· ·",
                  "· – – –","– · –","· – · ·","– –","– ·","– – · – –","– – –","· – – ·",
                  "– – · –","· – ·","· · ·","–","· · –","· · · –","· – –","– · · –","– · – –",
                  "– – · ·"]
codigo_morse = {}
for indice, letra in enumerate(alfabeto):
    codigo_morse[letra] = simbolos_morse[indice]

print(str(codigo_morse)+'\n')
print("Por ejemplo el código Morse de la letra P es: " + codigo_morse["P"])
```

```
{'A': '· –', 'B': '– · · ·', 'C': '– · – ·', 'D': '– · ·', 'E': '·', 'F': '· · – ·', 'G': '– – ·', 'H': '· · · ·',
'I': '· ·', 'J': '· – – –', 'K': '– · –', 'L': '· – · ·', 'M': '– –', 'N': '– ·', 'Ñ': '– – · – –', 'O': '– – –',
'P': '· – – ·', 'Q': '– – · –', 'R': '· – ·', 'S': '· · ·', 'T': '–', 'U': '· · –', 'V': '· · · –', 'W': '· – –',
'X': '– · · –', 'Y': '– · – –', 'Z': '– – · ·'}

Por ejemplo el código Morse de la letra P es: · – – ·
```

A continuación crea un programa que lea una frase del teclado y te la convierta a Morse utilizando el diccionario anterior.

In [47]:
```
texto = str(input("Ingrese una frase: "))
texto = texto.upper().replace(" ", "")
morse = [] #lista para guardar los simbolos en codigo morse de la frase
for i in texto:
    morse.append(codigo_morse[i])

print(morse)
```

```
Ingrese una frase: La ruta nos aporto otro paso natural
['· – · ·', '· –', '· – ·', '· · –', '–', '· –', '– ·', '– – –', '· · ·', '· –', '· – – ·', '– – –', '· – ·', '–', '–
– –', '– – –', '–', '· – ·', '– – –', '· – – ·', '· –', '· · ·', '– – –', '– ·', '· –', '–', '· · –', '· – ·', '· –',
'· – · ·']
```

Ejercicio 5

Crea una función que dados dos diccionarios nos diga que claves están presentes en ambos.

In [48]:
```
a = {"a":1,"b":2,"c":3,"d":4,"e":5,"f":6,"g":7}
b = {"a":2,"b":5,"e":4,"f":21}

def clavesComunes(a,b): #a y b son diccionarios
    claves_a = a.keys()
    claves_b = b.keys()
    comunes = [] #lista que almacena las claves que son comunes en los dos diccionarios
    for i in claves_a:
        if i in claves_b:
            comunes.append(i)
    return comunes

print(clavesComunes(a,b))
```

```
['a', 'b', 'e', 'f']
```

Ejercicio 6

Crea una función que dado un número N nos diga si es primo o no (tiene que ir dividiendo por todos los números x comprendidos entre 2 y el propio número N menos uno y ver si el cociente de N/x tiene resto entero o no).

In [3]:
```
n = int(input("Ingrese un número: "))

#funcion que devuelve un valor booleano, True si n es primo y False si no es primo
def esPrimo(n):
    b = False
    for i in range (2,n):
        if n%i == 0:
            b = False
            break
        else:
            b = True
    return b

if esPrimo(n) == True:
    print("Es Primo")
elif esPrimo(n) == False:
    print("No es Primo")
```

```
Ingrese un número: 41
Es Primo
```

Sea n un numero entero, encuentre los numeros primos entre 0 y n

In [6]:
```
n = int(input("Ingrese un número: "))
def nPrimos(n):
    primos = []
    for i in range(2,n+1):
        if esPrimo(i) == True:
            primos.append(i)
    return primos

print("Los numeros primos entre 0 y " + str(n) + " son: " +str(nPrimos(n)))
```

```
Ingrese un número: 60
Los numeros primos entre 0 y 60 son: [3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59]
```

Ejercicio 7

Investiga la documentación de la clase string y crea un método que lea una frase del teclado y escriba la primera letra de cada palabra en Mayúscula.

In [72]:
```
texto = str(input("Ingrese una frase: "))
lista_texto = texto.split()
filtro = [] #lista que almacena las primeras letras de cada palabra en mayuscula
for i in lista_texto:
    filtro.append(i[0].upper())
print(filtro)
```

```
Ingrese una frase: Eso no lo se
['E', 'N', 'L', 'S']
```

Ejercicio 8

Crea una función que calcule el máximo común divisor de dos números introducidos por el usuario por teclado.

In [24]:
```python
print("Ingrese dos numeros: ")
n1 = int(input())
n2 = int(input())

#funcion para calcular los divisores de un numero
def divisores(n):
    div = []
    for i in range(1,n+1):
        if n%i==0:
            div.append(n/i)
    return div

d1 = divisores(n1) #divisores de n1
d2 = divisores(n2) #divisores de n2
dComunes = list(set(d1) & set(d2)) #Divisores comunes de n1 y n2
mcd = np.max(dComunes) #calculo del maximo de los divisores comunes
print("El mcd de "+str(n1)+" y "+str(n2)+" es: "+str(mcd))
```

```
Ingrese dos numeros:
30
45
El mcd de 30 y 45 es: 15.0
```

Ejercicio 9

Investiga el Cifrado del César y crea una función que lo reproduzca en Python. Cada letra del mensaje original se desplaza tres posiciones en el alfabeto estándar. La A se convierte en la D, la B se convierte en la E, la C se convierte en la F... y cuando se acaba el alfabeto se le vuelve a dar la vuelta: la X se convierte en la A, la Y en la B y la X en la C. Los números no sufren ninguna modificación.

In [9]:
```
alfabeto = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZABC".lower()
alfabeto_cod = "DEFGHIJKLMNÑOPQRSTUVWXYZABC".lower()

texto = str(input("Ingrese una frase: ")).lower()
texto = texto.split()
texto_cesar = [] #lista para guardar las palabras codificadas

def codCesar(texto):
    for i in texto:
        indice = alfabeto.index(i) #devuelve el indice del elemento i
        texto = texto.replace(i,alfabeto_cod[indice])
    return texto

for i, palabra in enumerate(texto):
    texto_cesar.append(codCesar(palabra))

texto_cesar = " ".join(texto_cesar) #concatena las palabras de la lista texto_cesar
print("La frase codificada es: " + texto_cesar)
```

```
Ingrese una frase: La transformacion se puede representar alineando dos alfabetos
La frase codificada es: ñd wudpvluurdflup vh sxhgh uhvuhvhpwdu gñlphgpgr grv dñidhhwrv
```

Ejercicio 10

Dado una lista de nombres de persona, escribe un algoritmo que los ordene de tres formas diferentes:

- De forma alfabética
- forma alfabética invertida
- De nombre más corto al más largo.

In [43]:
```
nombres = ['Ana', 'Miguel', 'Carlos', 'Bianca', 'Eduardo', 'Gabriela', 'Margarita',"Raul"]

#primera forma
nombres.sort()
print("Orden Alfabetico: " + str(nombres))
#segunda forma
nombres.sort(reverse=True)
print("Orden Alfabetico Inverso: " + str(nombres))
#tercera forma
nombres.sort(key=len)
print("Orden por el tamaño de cada palabra: " + str(nombres))
```

```
Orden Alfabetico: ['Ana', 'Bianca', 'Carlos', 'Eduardo', 'Gabriela', 'Margarita', 'Miguel', 'Raul']
Orden Alfabetico Inverso: ['Raul', 'Miguel', 'Margarita', 'Gabriela', 'Eduardo', 'Carlos', 'Bianca', 'Ana']
Orden por el tamaño de cada palabra: ['Ana', 'Raul', 'Miguel', 'Carlos', 'Bianca', 'Eduardo', 'Gabriela', 'Margarit
a']
```