

Tarea 5

March 26, 2020

0.1 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 [4]: 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 [4]: solucion(2,-4,1)
```

```
Out[4]: (1.7071067811865475, 0.2928932188134524)
```

0.2 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 [49]: 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 aporoto otro paso natural
Es un Palindromo
```

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

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

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).

{ 'A': 'û ', 'B': ' û û û', 'C': ' û û', 'D': ' û û', 'E': 'û', 'F': 'û û û', 'G': ' û', 'H': 'û û' }

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

Ingrese una frase: La ruta nos aportó otro paso natural

['u u u', 'u ', 'u u', 'u u ', ' ', 'u ', ' u', ' ', 'u u u', 'u ', 'u u', ' ', 'u u', ']

0.5 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']
```

0.6 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 [44]: 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 [64]: 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: 50
Los numeros primos entre 0 y 50 son: [3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]

0.7 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']

0.8 Ejercicio 8

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

```
In [24]: 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

0.9 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 Z en la C. Los números no sufren ninguna modificación.

```
In [9]: alfabeto = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".lower()
        alfabeto_cod = "DEFGHIJKLMNOPQRSTUVWXYZABC".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 wudpvlurdf lup vh sxhgh uhvuhvhpwdu gñlphgpggr grv dñidhhwrv

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

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