Tarea 5: Estructuras de datos en Python

Daniel Eduardo Macias Estrada

13/3/2021

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.

```
import math
import numpy as np
def eq2(a,b,c):
    d = math.sqrt(b**2 - 4*a*c)
    s1, s2 = (-b + d)/(2*a), (-b - d)/(2*a)
    return s1, s2

eq2(2,3,1)
```

```
## (-0.5, -1.0)
```

Ejercicio 2

Crea una función que lea una frase del teclado y nos diga si es o no un palindromo

```
def palindromo(frase):
    frase = frase.lower()
    frase = frase.replace(' ', '')
    accents = {
        'á':'a',
        'é':'e',
        'í':'i',
        'ó':'o',
        'ú':'u'
    }
    for key in accents.keys():
        frase = frase.replace(key, accents[key])
    return frase == frase[::-1]

palindromo("La ruta nos aportó otro paso natural")
```

```
## True
```

Ejercicio 3

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

```
d = dict(zip(range(1,11), np.sqrt(range(1,11))))
d
```

{1: 1.0, 2: 1.4142135623730951, 3: 1.7320508075688772, 4: 2.0, 5: 2.23606797749979, 6: 2.449489742783178, 7: 2.6457513110645907, 8: 2.8284271247461903, 9: 3.0, 10: 3.1622776601683795}

Ejercicio 4

Crea un diccionario que tenga como claves las letras del alfabeto castellano y como valores los símbolos del código morse. Crea un programa que lea una frase del teclado y te la convierta a Morse, con el diccionario anterior

```
abc = {
    "A" : "._",
    "B" : "_...",
    "C" : "_._.",
    "CH" : "____",
    "D" : "_..",
    "E" : ".",
    "F" : ".._.",
    "G" : "__.",
    "H" : "....",
    "I" : "..",
    "J" : ".___",
    "K" : "_._",
    "L" : "._..",
    "M" : "__",
    "N" : "_.",
    "Ñ" : "__.__",
    "O" : "___",
    "P" : ".__.",
    "Q" : "__._",
    "R" : "._.",
    "S" : "...",
    "T" : "_",
    "U" : ".._",
    "V" : "..._",
    "W" : ".__",
    "X" : "_.._",
    "Y" : "_.__",
    "Z" : "__..",
    " " : "/"
}

accents = {
    'Á':'A',
    'É':'E',
    'Í':'I',
    'Ó':'O',
```

```
    'Ú':'U'
}

def morse(frase):
    nfrase = ""

    frase = frase.upper()
    for key in accents.keys():
        frase = frase.replace(key, accents[key])
    l = list(frase);

    for let in l:
        nfrase = nfrase + abc[let] + " "
    print(nfrase)

morse("Soy yo Dániel")
```

```
## ... ___ _·__ / _·__ ___ / _·· ·_ _· ·· · ·_··
```

Ejercicio 5

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

```
def claves(d1,d2):
    keys_c = []
    keys1 = list(d1)
    keys2 = list(d2)
    for i in range(0,len(keys1)):
        if keys1[i] in keys2:
            keys_c.append(keys1[i])
    return keys_c

dic = dict(A=1, B=2)
dic2 = dict(C=5, A=4)

claves(dic,dic2)
```

```
## ['A']
```

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)

```
def primo(n):
    for i in range(2,n):
        if n % i == 0:
            return False
    return True

primo(23)
```

```
## True
```

```
primo(11)
```

```
## True
```

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.

```
def capitalLetters(frase):
    nueva = ""
    lista = frase.split(" ")
    for let in lista:
        pal = let[0].upper() + let[1:]
        nueva = nueva + pal + " "
    return nueva

capitalLetters("Si tengo mucho sueño")
```

```
## 'Si Tengo Mucho Sueño '
```

Ejercicio 8

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

```
def MCD(a,b):
    '''Obtener el máximo común divisor con el algoritmo de Euclides.
    La variable a es el mayor y la variable b el menor'''
    while a % b != 0:
        a, b = b, a % b
    return b

MCD(60,48)
```

```
## 12
```

```
MCD(56,42)
```

```
## 14
```

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 nunguna modificación

```
abc = ['A','B','C','D','E','F','G','H','I','J','K','L','M','N','Ñ','O','P','Q','R','S','T','U','V','W',

def cifradoCesar(frase):
    newfrase = ""
    frase = frase.upper()
    lista = list(frase)
    for let in lista:
        if let == " ":
            newfrase = newfrase + " "
            continue
        d = abc.index(let)
        newfrase = newfrase + abc[d+3]
    return newfrase

cifradoCesar("Daniel es mi nombre")
```

```
## 'GDPLHÑ HV OL PROEUH'
```

Ejercicio 10

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

A. De forma alfabética B. De forma alfabética invertida C. De nombre más corto a más largo

```
def ordenar(lista):
    alf = sorted(lista)
    alf_inv = sorted(lista, reverse=True)
    length = sorted(lista, key = lambda x: len(x))
    return [alf, alf_inv, length]

ordenar(['Diego', 'Miguel', 'Yanel', 'Gaby', 'Daniel'])
```

[[‘Daniel’, ‘Diego’, ‘Gaby’, ‘Miguel’, ‘Yanel’], [‘Yanel’, ‘Miguel’, ‘Gaby’, ‘Diego’, ‘Daniel’], [‘Gaby’, ‘Diego’, ‘Yanel’, ‘Miguel’, ‘Daniel’]]