Ejercicios1

October 30, 2018

0.1 Relación ejercicios 1

0.1.1 1. Escribe una función contar_letras(palabra, letra) que devuelva el número de veces que aparece una letra en una palabra.

```
In [1]: def contar_letras(palabra,letra):
            contador = 0
            for i in palabra:
                if(letra == i):
                    contador += 1
            return contador
```

```
In [2]: contar_letras("HolaMundo", "a")
```

```
Out[2]: 1
```

```
In [3]: contar_letras("HolaMundo", "o")
```

```
Out[3]: 2
```

```
In [4]: contar_letras("CarlosSequi", "S")
```

```
Out[4]: 1
```

0.1.2 2. Escribe una función eliminar_letras(palabra, letra) que devuelva una versión de palabra que no contiene el carácter letra.

```
In [5]: def eliminar_letras(palabra,letra):
            palabra_new = ""
            for i in palabra:
                if(letra != i):
                    palabra_new = palabra_new + i

            return palabra_new
```

```
In [6]: eliminar_letras("HolaMundo", "o")
```

```
Out[6]: 'HlaMund'
```

```
In [7]: eliminar_letras("HolaMundo", "M")
```

```
Out[7]: 'Holaundo'

In [8]: eliminar_letras("HolaMundo", "A")

Out[8]: 'HolaMundo'
```

0.1.3 3. Escribe una función buscar(palabra, sub) que devuelva la posición en la que se puede encontrar sub dentro de palabra o -1 en caso de que no esté.

```
In [150]: def buscar(palabra,sub):
              encontrado = -1
              existe = False
              i = 0
              while (not existe and (i < len(palabra))):
                  if (palabra[i] == sub[0]):
                      if(len(sub) == 1):
                          existe = True
                          encontrado = i
                      elif((len(palabra) - i) > len(sub)):
                          z = 1
                          ii = i + 1
                          existe2 = True
                          while(existe2 and z < len(sub)):
                              if (palabra[ii] != sub[z]):
                                  existe2 = False
                              z+=1
                              ii+=1
                          if(existe2):
                              existe = True
                              encontrado = i
                  i +=1

              return encontrado

In [151]: buscar("HolaMundo", "a")

Out[151]: 3

In [152]: buscar("HolaMundo", "MU")

Out[152]: -1

In [153]: buscar("HolaMundo", "Mun")

Out[153]: 4

In [154]: buscar("HolaMundo", "z")

Out[154]: -1
```

0.1.4 4. Escribe una función num_vocales(palabra) que devuelva el número de vocales que aparece en la palabra.

```
In [155]: # Preguntar si hay que contar mayus y min
          def num_vocales(palabra):
              contador = 0
              palabra_min = palabra.lower()
              for i in palabra_min:
                  if(i == 'a' or i == 'e' or i == 'i' or i == 'o' or i == 'u'):
                      contador +=1
              return contador

In [156]: num_vocales("HOLAmundo")

Out[156]: 4

In [157]: num_vocales("cdfgc")

Out[157]: 0
```

0.1.5 5. Escribe una función vocales(palabra) que devuelva las vocales que aparecen en la palabra.

```
In [158]: #Preguntar si tenemos que eliminar repetidos y si tenemos que devolver en mayus o min
          def vocales(palabra):
              vocales = []
              palabra_min = palabra.lower()
              for i in palabra_min:
                  if(i == 'a' or i == 'e' or i == 'i' or i == 'o' or i == 'u'):
                      vocales.append(i)

              return vocales

In [159]: vocales("HOLAmundo")

Out[159]: ['o', 'a', 'u', 'o']
```

0.1.6 6. Escribe una función es_inversa(palabra1, palabra2) que devuelve True si una palabra es la misma que la otra pero con los caracteres en orden inverso. Por ejemplo 'absd' y 'dsba'

```
In [160]: #Preguntar si se puede usar break
          def es_inversa(palabra1,palabra2):
              inversa = True
              if(len(palabra1) == len(palabra2)):
                  i = 0
                  z = len(palabra2) - 1
                  while((i < len(palabra1)) and inversa):
                      if(palabra1[i] != palabra2[z]):
                          inversa = False
```

```
                z -= 1
                i+=1
        else:
            inversa = False

        return inversa
```

```
In [161]: es_inversa("absd","dsba")
```

```
Out[161]: True
```

```
In [162]: es_inversa("absd","dsza")
```

```
Out[162]: False
```

```
In [163]: es_inversa("absd","dsbaa")
```

```
Out[163]: False
```

0.1.7 7. Escribe una función comunes(palabra1, palabra2) que devuelva una cadena formada por los caracteres comunes a las dos palabras.

```
In [172]: #La función devolverá los caractéres que existan en ambas palabras pero sin repetidos
          def comunes(palabra1,palabra2):
              comunes = ""
              for i in palabra1:
                  if ( (buscar(palabra2,i) != -1) and (buscar(comunes,i) == -1)):
                      comunes += i

              return comunes
```

```
In [173]: comunes("hola","mundo")
```

```
Out[173]: 'o'
```

```
In [174]: comunes("invierno","primavera")
```

```
Out[174]: 'iver'
```

```
In [175]: comunes("pollo","polla")
```

```
Out[175]: 'pol'
```

0.1.8 8. Escribe una función eco_palabra(palabra) que devuelva una cadena formada por palabra repetida tantas veces como sea su longitud. Por ejemplo 'hola' -> 'holaholaholahola'

```
In [176]: def eco_palabra(palabra):
              palabra_new = ""
              for i in range(len(palabra)):
                  palabra_new += palabra

              return palabra_new
```

```
In [177]: eco_palabra("Hola")

Out[177]: 'HolaHolaHolaHola'

In [178]: eco_palabra("HolaMundo")

Out[178]: 'HolaMundoHolaMundoHolaMundoHolaMundoHolaMundoHolaMundoHolaMundoHolaMundoHolaMundo'

In [179]: eco_palabra("")

Out[179]: ''
```

0.1.9 9. Escribe una función palindromo(frase) que determine si frase es un palíndromo. Es decir, que se lea igual de izquierda a derecha que de derecha a izquierda (sin considerar espacios).

```
In [180]: def palindromo(frase):
              z = len(frase) -1
              palindromo = True
              i = 0
              while(palindromo and i < len(frase)):
                  if(frase[i] != frase[z]):
                      palindromo = False
                  z-=1
                  i+=1
              return palindromo

In [181]: palindromo("an a")

Out[181]: False

In [182]: palindromo("cana")

Out[182]: False

In [183]: palindromo("ana")

Out[183]: True
```

0.1.10 10. Escribe una función orden_alfabetico(palabra) que determine si las letras que forman palabra aparecen en orden alfabético. Por ejemplo: 'abejo'

```
In [184]: def orden_alfabetico(palabra):
              ordenada = True
              i = 0
              while(ordenada and i < (len(palabra)-1)):
                  if(palabra[i] > palabra[i+1]):
                      ordenada = False
                  i+=1
              return ordenada
```

```
In [185]: orden_alfabetico("abejo")
```

```
Out[185]: True
```

```
In [186]: orden_alfabetico("gallina")
```

```
Out[186]: False
```

0.1.11 11. Escribe una función trocear(palabra, num) que devuelva una lista con trozos de tamaño num de palabra.

```
In [187]: def trocear(palabra,num):
              longitud = len(palabra)
              lista = list()
              if(num != 0):
                  longitud = longitud//num
                  indice = 0
                  for i in range(longitud):
                      trozo = ""
                      for z in range(num):
                          trozo+=palabra[indice]
                          indice +=1

                      lista.append(trozo)
              else:
                  lista.append(palabra)
              return lista
```

```
In [188]: trocear("holaaA",2)
```

```
Out[188]: ['ho', 'la', 'aA']
```

```
In [189]: trocear("hola",2)
```

```
Out[189]: ['ho', 'la']
```

```
In [190]: trocear("hola",3)
```

```
Out[190]: ['hol']
```

```
In [191]: trocear("hola",4)
```

```
Out[191]: ['hola']
```

```
In [192]: trocear("hola",0)
```

```
Out[192]: ['hola']
```

```
In [193]: trocear("hola",1)
```

```
Out[193]: ['h', 'o', 'l', 'a']
```

0.1.12 12. Un anagrama de una palabra pal1 es una palabra formada con las mismas letras que pal1 pero en orden distinto. Escribe una función anagrama(palabra1, palabra2) que determine si es una anagrama. Ejemplo: marta – trama.

```
In [194]: def anagrama(palabra1, palabra2):
              is_anagrama = True
              i = 0
              while(is_anagrama and i < len(palabra1)):
                  if(contar_letras(palabra1,palabra1[i]) != contar_letras(palabra2,palabra1[i]))
                      is_anagrama = False
                  i+=1
              return is_anagrama

In [195]: anagrama("marta", "trama")

Out[195]: True

In [196]: anagrama("marta", "carlos")

Out[196]: False
```