Ejercicios Listas, tuplas, cadenas de caracteres, conjuntos y diccionarios

October 7, 2017

0.1 ALEJANDRO SANTORUM VARELA - EJERCICIOS LISTAS, TUPLAS, CADENAS DE CARACTERES, CONJUNTOS Y DICCIONARIOS

EJERCICIO 1 - Dada la lista $L = [3, 5, 6, 8, 10, 12]$ se pide: a) Averiguar la posición del número 8. b) Cambiar el valor 8 por el 9 sin reescribir toda la lista. c) Intercambiar el 5 y el 9. d) Intercambian cada valor por el que ocupa su posición simétrica, es decir, dar la vuelta a la lista. e) Crear la lista resultante de concatenar a la original con el resultado anterior.

```
In [1]: L = [3,5,6,8,10,12]
        LL = list(L)
        a = L.index(8)
        print("posicion del 8: ")
        print(a)
        print(" ")
        L[a] = 9
        print("Lista apartado b:")
        print(L)
        print(" ")
        cinco = L.index(5)
        nueve = L.index(9)
        L[cinco] = 9
        L[nueve] = 5
        print("Lista apartado c: ")
        print(L)
        print(" ")
        L.reverse()
        print("Lista apartado d:")
        print(L)
        print(" ")
        print("Lista apartado e: ")
        print(LL + L)
```

```
posicion del 8:
3

Lista apartado b:
[3, 5, 6, 9, 10, 12]
```

```
Lista apartado c:
[3, 9, 6, 5, 10, 12]

Lista apartado d:
[12, 10, 5, 6, 9, 3]

Lista apartado e:
[3, 5, 6, 8, 10, 12, 12, 10, 5, 6, 9, 3]
```

EJERCICIO 2 - La orden prime_range(100, 1001) genera la lista de números primos entre 100 y 1000. Se pide, siendo primos=prime_range(100, 1001): a) Averiguar el primo que ocupa la posición central en primos. b) Averiguar la posición, en primos, de 331 y 631. c) Extraer, conociendo las posiciones anteriores, la sublista de primos entre 331 y 631 (ambos incluidos). d) Extraer una sublista de primos que olvide los dos centrales de cada cuatro. e) Extraer una sublista de primos que olvide el tercero de cada tres.

```
In [2]: L = prime_range(100, 1001)
        a = len(L)
        pos = floor(a/2)
        primo_central = L[pos]
        print("a) Primo que ocupa la posicion central: ")
        print(primo_central)
        print(" ")
        pos_331 = L.index(331)
        pos_631 = L.index(631)
        print("b) La posición de 331 y de 631, respectivamente es:")
        print(pos_331, pos_631)
        print(" ")
        subL = L[pos_331:pos_631+1]
        print("c) Sublista desde 331 a 631: ")
        print(subL)
        print(" ")
```

```
a) Primo que ocupa la posicion central:
509

b) La posición de 331 y de 631, respectivamente es:
(41, 89)

c) Sublista desde 331 a 631:
[331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439
```

EJERCICIO 3 - Considérese el número 1000! (factorial(1000)): a) £En cuantos ceros acaba? b) £Se encuentra el número 666 entre sus subcadenas? En caso afirmativo, localizar (encontrar los

índices de) todas las apariciones. c) Encontrar la subcadena más larga de doses consecutivos. Mostrarla con los dos dígitos que la rodean.

```
In [3]: N = factorial(1000)

        #primer apartado
        L = list(factor(N))
        for a,b in L:
            if a == 5:
                print("a) El número de ceros que acaba es:")
                print(b)
                print(" ")

        #segundo apartado
        C = str(N)
        a = C.find('666')
        if a != -1:
            print("b) Sí que aparece el 666 entre sus subcadenas y estos son sus índices (menos
            print(a)
            while a != -1:
                a = C.find('666', a+1)
                print(a)
        else:
            print("b) No aparece 666 entre sus subcadenas")

        print(" ")


        #tercer apartado
        a=0
        cont=0
        maxi = 0
        N = factorial(1000)
        C = str(N)
        longitud = len(C)

        while a<longitud:          #programa 1 - calculamos el mayor número de doses consecuti
            if C[a] == '2':
                cont = cont+1
                a=a+1
            else:
                if cont > maxi:
                    maxi = cont
                a=a+1
                cont = 0

        print("c.1) El mayor número de doses consecutivos es:")#Esta información nos permite
                                                               #mostrar la cadena de doses más la
```

```
        print(maxi)
        print(" ")
```

```
a) El número de ceros que acaba es:
249

b) Sí que aparece el 666 entre sus subcadenas y estos son sus índices (menos el -1):
123
383
-1

c.1) El mayor número de doses consecutivos es:
5
```

```
In [4]: #continuación del tercer apartado
        N = factorial(1000)
        C = str(N)
        a = C.find('22222') #Introducir tantos doses como el apartado c.1 anterior indica.
        print("c.2) La cadena de doses más larga mostrada con los dígitos que la rodean es: ")
        print(C[a-1:a+6])
        print(" ")
```

```
c.2) La cadena de doses más larga mostrada con los dígitos que la rodean es:
0222221
```

EJERCICIO 4 - Si se aplica la función sum() a una lista numérica, nos devuelve la suma de sus elementos. En particular, la composición sum(k.digits()), para un k variable entero, nos devuelve la suma de sus dígitos (en base 10). a) Calcular, con la composición sum(k.digits()), la suma de los dígitos del número k=factorial(1000). b) Calcular lo misma suma sin utilizar el método .digits(). Sugerencia: con siderar la cadena digitos='0123456789', en la que dígitos[0]='0', dígitos[1]='1', ..., dígitos[9]='9'. Sumar los elementos de la lista [j (veces que aparece j en 1000!) : con j = 1,2,3,4,5,6,7,8,9.]

```
In [5]: n = factorial(1000)
        C = str(n)

        #primer apartado
        sum1 = sum(n.digits())
        print("a) La suma de los dígitos de 1000! utilizando .digits() es:")
        print(sum1)
        print(" ")

        #segundo apartado
        a = 0
        sum2 = 0
```

```
        longitud = len(C)

        while a<longitud:
            if C[a] == '1':
                sum2 = sum2+1
            elif C[a] == '2':
                sum2 = sum2+2
            elif C[a] == '3':
                sum2 = sum2+3
            elif C[a] == '4':
                sum2 = sum2+4
            elif C[a] == '5':
                sum2 = sum2+5
            elif C[a] == '6':
                sum2 = sum2+6
            elif C[a] == '7':
                sum2 = sum2+7
            elif C[a] == '8':
                sum2 = sum2+8
            elif C[a] == '9':
                sum2 = sum2+9
            a=a+1

        print("La suma total de sus dígitos por un método alternativo a .digits() es: ")
        print(sum2)
        print(" ")
```

```
a) La suma de los dígitos de 1000! utilizando .digits() es:
10539

La suma total de sus dígitos por un método alternativo a .digits() es:
10539
```

EJERCICIO 5 - A partir de la cadena de caracteres texto='Omnes homines, qui sese student praestare ceteris animalibus, summa ope niti decet ne uitam silentio transeant ueluti pecora, quae natura prona atque uentri oboedientia finxit.' extraer la lista de los caracteres del alfabeto utilizados, sin repeticiones, sin distinguir mayúsculas de minúsculas y ordenada alfabéticamente.

```
In [6]: str = 'Omnes homines, qui sese student praestare ceteris animalibus, summa ope niti de
        A = set(str)
        L = list(A)
        L.sort()
        LL = L[3:]
        print("lista requerida:")
        print(LL)
        print(" ")
```

```
lista requerida:
['a', 'b', 'c', 'd', 'e', 'f', 'h', 'i', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'x']
```

EJERCICIO 6 - Sin utilizar los métodos .divisors() ni la función max(), elabora código que, a partir de dos números a y b, calcule: a) El conjunto Diva = {k N : k | a} b) El conjunto Divb = {k N : k | b} c) El conjunto Diva,b = {k N : k | a y k | b}.

```
In [7]: a = 25 #es un ejemplo, puede cambiar el número
        b = 30 #es un ejemplo, puede cambiar el número

        Diva = set()
        Divb = set()
        Divab = set()

        i=1
        j=1
        k=1

        while i<=a:
            if a%i == 0:
                Diva.add(i)
            i=i+1

        print("a) Divisores de a: ")
        print(Diva)
        print(" ")

        while j<=b:
            if b%j == 0:
                Divb.add(j)
            j=j+1

        print("b) Divisores de b: ")
        print(Divb)
        print(" ")

        if a<=b:
            minimo = a
        else:
            minimo = b

        while k<=minimo:
            if (a%k == 0 and b%k == 0):
                Divab.add(k)
            k=k+1
```

```
        print("c) Divisores de a y de b: ")
        print(Divab)
        print(" ")

a) Divisores de a:
set([1, 5, 25])

b) Divisores de b:
set([1, 2, 3, 5, 6, 10, 15, 30])

c) Divisores de a y de b:
set([1, 5])
```

EJERCICIO 7 - El mínimo común múltiplo, m, de dos números es menor o igual que su producto $m <= a * b$. Sin utilizar la función min(), elabora el código que, a partir de dos números a y b, calcule: a) El conjunto $Mult_{a(b)} = \{k \in \mathbb{N} : a|kyk < a * b\}$ b) El conjunto $Mult_{b(a)} = \{k \in \mathbb{N} : b|kyk < a * b\}$ c) El conjunto $Mult_{a,b} = \{k \in \mathbb{N} : a|k, b|kyk < a * b\}$

```
In [8]: a = 5 #es un ejemplo, puede cambiar el número
        b = 3 #es un ejemplo, puede cambiar el número

        Mult1 = set()
        Mult2 = set()
        Mult3 = set()

        i=1
        j=1
        k=1

        while i<=a*b:
            if i%a == 0:
                Mult1.add(i)
            i=i+1

        A=list(Mult1)
        A.sort()
        mina = A[0]
        AA = set(A)
        print("a)")
        print("Conjunto MultA(B): ")
        print(AA)
        print("Mínimo de MultA(B): ")
        print(mina)
        print(" ")

        while j<=a*b:
```

```
            if j%b == 0:
                Mult2.add(j)
            j=j+1

        B=list(Mult2)
        B.sort()
        minb = B[0]
        BB = set(B)
        print("b)")
        print("Conjunto MultB(A): ")
        print(BB)
        print("Mínimo de MultB(A): ")
        print(minb)
        print(" ")

        while k<=a*b:
            if (k%a == 0 and k%b == 0):
                Mult3.add(k)
            k=k+1

        C=list(Mult3)
        C.sort()
        minc = C[0]
        CC = set(C)
        print("c)")
        print("Conjunto MultA,B: ")
        print(CC)
        print("Mínimo de MultA,B: ")
        print(minc)
        print(" ")
```

```
a)
Conjunto MultA(B):
set([10, 5, 15])
Mínimo de MultA(B):
5

b)
Conjunto MultB(A):
set([9, 3, 12, 6, 15])
Mínimo de MultB(A):
3

c)
Conjunto MultA,B:
set([15])
Mínimo de MultA,B:
15
```

EJERCICIO 8 - Dada una lista de números enteros, construye un conjunto con los factores primos de todos los números de la lista.

```
In [9]: L = [2,3,5,6,7,8,8,10,12,20,21,25,27,403,345,45,65,54] #lista cualquiera
        l = len(L)

        C = set()

        i=0
        j=0
        while i<l:
            a = L[i]
            LL = list(a.factor()) #para cada elemento de la lista
                                  #hayamos su descomposición factorial
            ll = len(LL)
            while j<ll:
                b=LL[j][0] #en la tupla, escogemos el factor y no su multiplicidad
                C.add(b)
                j=j+1
            i=i+1
            j=0

        print(C)
```

```
set([2, 3, 5, 7, 43, 13, 17, 19, 29, 31])
```