SESION 29

Implementa las listas de Python para en el manejo y control de algoritmos, que requieren el uso de grandes volumenes de datos y de diverso tipo.

1. Introducción:

En el presente indicador, los alumnos aprenderán a manipular los datos usando a las listas como lugar almacenamiento temporal en diversas cantidades y tipos; para luego usarlos en los diversos procesos planteados en un programa o caso a desarrollar.

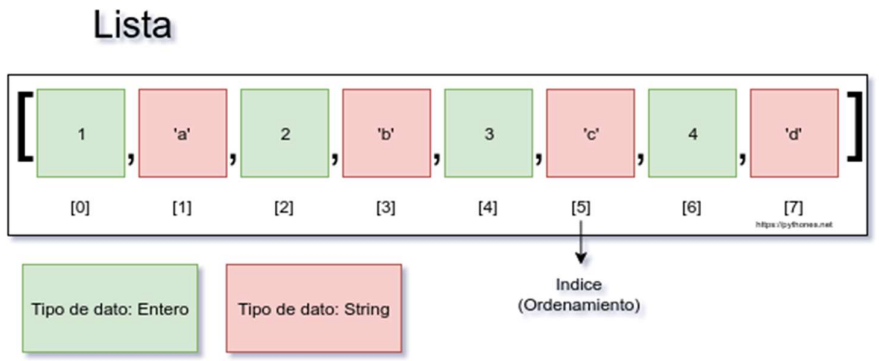
2. Colecciones de datos(a).

a. Las listas y su uso

Las listas en Python son estructuras de datos que usamos cuando requerimos trabajar con grandes cantidades de datos y cuyo valor puede variar a lo largo del tiempo, los tipos de datos que puede almacenar son diversos; al poderse modificar se indica que puede ser mutables, las listas tienen una serie de funciones y métodos que permite manipular de manera adecuada los datos.

Si tuviéramos que manipular las notas de un salón de 60 alumnos y operarlas temporalmente para obtener promedios o mensajes asociados, el registrarlas en variables sería complicado, en ese caso usamos una lista que los almacene temporalmente y se les pueda manipular varias veces; y evitar acceder a la base de datos a cada rato o en su defecto la carga manual.

En la siguiente imagen muestra a una lista con elementos:



Características asociadas:

- Los datos se almacenan en orden.
- La lista tiene dos índices: orden positivo y orden inverso.
- Los índices de las listas siempre inician en cero (0).
- La lista puede almacenar cualquier tipo de datos y se permiten repeticiones.

Actividades básicas sobre las listas:

- La sintaxis y creación de una lista, se indica como una secuencia de valores encerrados entre corchetes y separados por comas, observar:

```
Lst=[]

Lst=[2,45.6,"Hola",[3,7,8],"*"]
```

En el primer caso podemos ver la creación de una lista vacía.

En el segundo caso vemos una lista con elementos, los cuales dicho sea de paso tienen diverso tipo de datos.

- Cuando se desea alterar un valor existente en la lista debemos reasignarle el dato, por medio del índice, tenemos la lista:

Lst	2	45.6	Hola	[3,7,8]	*
	0	1	2	3	4

Codificando la lista:

```
Lst=[2,45.6,"Hola",[3,7,8],"*"]
print(Lst)

Lst[2]="Cambio de dato"

print(Lst)
```

Al ejecutar se muestra lo siguiente:

```
[2, 45.6, 'Hola', [3, 7, 8], '*']
[2, 45.6, 'Cambio de dato', [3, 7, 8], '*']
```

Podemos observar que para imprimir una lista basta con usar la función print().

Para poder cambiar un valor debemos acceder por medio del índice asociado a dicho valor, si la lista está vacía, esta opción no funciona.

- Del mismo modo si se desea copiar valores existentes en una lista, podemos realizar lo siguiente:

```
Lst=[2,45.6,"Hola",[3,7,8],"*"]
print(Lst)

Lst[2]=Lst[4]

print(Lst)
```

Por medio de los índices se pueden copiar los valores, al ejecutar se muestra:

```
[2, 45.6, 'Hola', [3, 7, 8], '*']
[2, 45.6, '*', [3, 7, 8], '*']
```

Observe el dato * aparece copiado en la posición 2 en la segunda impresión.

b. Operaciones sobre listas: acceso y eliminación

Cuando se desea acceder a los valores de una lista debemos hacerlo por medio de los índices o usando alguna estructura de programación que lo permita, observar:

```
Lst=[2,45.6,"Hola",[3,7,8],"*"]

print(Lst[2])
print(Lst[4])

print(Lst)

for dato in Lst:
    print(dato)
```

Al ejecutar se muestra lo siguiente:

```
Hola
*
[2, 45.6, 'Hola', [3, 7, 8], '*']

2
45.6
Hola
[3, 7, 8]
*
```

En el primer caso se imprime los valores independientes por medio del índice.

En el segundo caso se imprime toda la lista pero con su formato ([ ])

En el tercer caso se imprime toda la lista, valor por valor por medio de la instrucción for.

Cuando se desea eliminar un valor de la lista, podemos usar la instrucción **del** indicando el índice a eliminar, observar:

```
Lst=[2,45.6,"Hola",[3,7,8],"*"]
print(Lst)

del(Lst[2])

print(Lst)
```

Al ejecutar se muestra lo siguiente:

```
[2, 45.6, 'Hola', [3, 7, 8], '*']
[2, 45.6, [3, 7, 8], '*']
```

El dato que correspondía a la posición 2 (“Hola”), ya no existe.

c. Uso de índices negativos

En una lista podemos indicar la obtención de datos en base a índices negativos, lo interesante es que en realidad no existen, sino que son manipulados por Python en base a los existentes; observe:

Entonces si codificamos de la siguiente manera, Python comprende lo que deseamos obtener:

```
Lst=[2,45.6,"Hola",[3,7,8],"*"]
print(Lst)

print(Lst[-1])
print(Lst[-3])
```

Al ejecutar el código obtenemos lo siguiente:

```
[2, 45.6, 'Hola', [3, 7, 8], '*']
*
Hola
```

Observación, se tiene los siguientes casos de acceso:

```
Lst=[2,45.6,"Hola",[3,7,8],"*"]
print(Lst)

print(Lst[3][2])
print(Lst[3][0])
print(Lst[-2][1])
```

Al ejecutar el código, se muestra lo siguiente:

```
[2, 45.6, 'Hola', [3, 7, 8], '*']
8
3
7
```

Podrá ver que se está accediendo a la lista interior.

d. Funciones vs métodos

Las funciones requieren de datos (parámetros) para operar, generalmente siempre deben generar un resultado.

Un método es un tipo específico de función: se comporta como una función y se parece a una función, pero difiere en la forma en que actúa y en su estilo de invocación; los métodos pueden cambiar el estado de la entidad donde se aplica.

La forma como se invoca es distinta en ambos casos, observar:

```
Lst=[442,6,18,3,46]
print(Lst)

print(len(Lst))

Lst.sort()
print(Lst)
```

Al ejecutar se muestra:

```
[442, 6, 18, 3, 46]
5
[3, 6, 18, 46, 442]
```

En el primer caso se trata de una función denominada len() y que devuelve el número de elementos de una lista.

En el segundo caso se trata de un método (sort), asociado a las listas, que permite ordenarlas.

e. Operaciones en listas: append(), insert()

Cuando se desea ingresar datos a una lista debemos usar el método append, de ese modo se agregarán elementos, observar:

```
Lst=[]
for i in range(4):
    print("Ingrese un número:")
    N=int(input())
    Lst.append(N)
print(Lst)
```

Al ejecutar el código se muestra:

```
Ingrese un número:
6
Ingrese un número:
3
Ingrese un número:
8
Ingrese un número:
12
[6, 3, 8, 12]
```

Para realizar la inserción de valores en una lista, se hace uso del método insert, de la siguiente manera:

```
Lst=[442,6,18,3,46]
print(Lst)

print("Ingrese un número:")
N=int(input())

Lst.insert(2,N)

print(Lst)
```

Al ejecutar se muestra:

```
[442, 6, 18, 3, 46]
Ingrese un número:
80
[442, 6, 80, 18, 3, 46]
```

Considerar que estamos por insertar el valor 80 en la posición 2, y eso es lo que ocurre finalmente.

Observaciones:

- Si deseamos recorrer la lista, pero por medio de sus índices, debemos trabajar de la siguiente manera:

```
Lst=[442,6,18,3,46]
for i in range(len(Lst)):
    print(Lst[i])
```

Al ejecutar nos muestra lo siguiente:

```
442
6
18
3
46
```

- Cuando deseamos intercambiar valores dentro de una lista, Python nos ofrece una forma bien peculiar, que funciona entre variables comunes del mismo modo, observar:

```
Lst=[2,7,11,56,3,99,45]
print(Lst)

Lst[1],Lst[4]=Lst[4],Lst[1]
print(Lst)
```

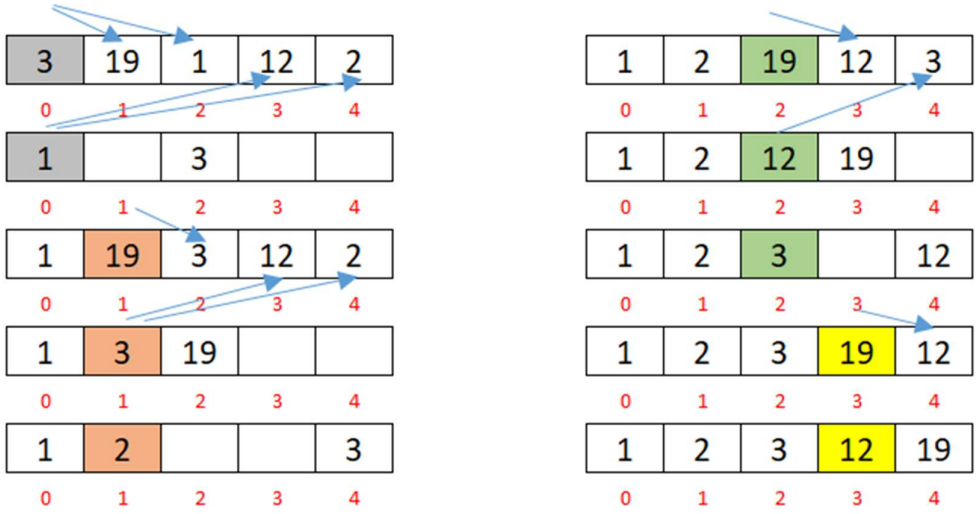
Al ejecutar, se muestra lo siguiente, saque sus conclusiones:

```
[2, 7, 11, 56, 3, 99, 45]
[2, 3, 11, 56, 7, 99, 45]
```

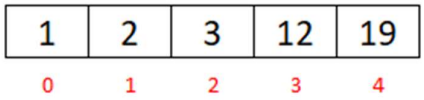
f. Ordenamiento de listas: Método de la burbuja

Cuando se desea ordenar una lista existen muchas formas de codificar, sin embargo, la más tradicional y fácil de comprender es el denominado método de la burbuja, donde se compara cada elemento de la lista con los siguientes existentes hacia la derecha, en caso de ser mayor (o menor, depende como quiera ordenar la lista), deberá realizar un intercambio de valores, este proceso debe ocurrir hasta llegar al penúltimo valor de la lista.

Cuando se finaliza el proceso, la lista deberá estar ordenada, observe:



Lista Ordenada



El código asociado al gráfico es el siguiente:

```
Lst=[3,19,1,12,2]
print(Lst)
for i in range(len(Lst)-1):
    for j in range(i+1,len(Lst)):
        if(Lst[i]>Lst[j]):
            Lst[i],Lst[j]=Lst[j],Lst[i]
print(Lst)
```

El primer for control al valor que compara
El segundo for controla a los valores comparados
En caso de cumplir que los valores comparados con mayores el 1ero al 2do, se realiza el intercambio.
Finalmente, la lista estará ordenada.

Sin embargo, existe una forma directa de realizar todo este código y es usando el método sort asociado a las listas, observar:

```
Lst=[3,19,1,12,2]
print(Lst)

Lst.sort()
print(Lst)
```

Al ejecutar obtenemos:

```
[3, 19, 1, 12, 2]
[1, 2, 3, 12, 19]
```

La idea de explicar el proceso de ordenamiento, es que usted pueda analizar y ver cómo es que se aplica la lógica al momento de generar un código que haga posible ordenar datos en una lista.

Se le sugiere ahora que investigue sobre el método **Shell** de ordenamiento y lo desarrolle.

g. Actividades con listas, generar copias, uso de rodajas.

Cuando operamos con listas, se puede generar varios casos, a continuación, veremos algunos de ellos:

- Unión de listas

```
Lst1=[3,19,1,12,2]
Lst2=[4,55,11,3,35,6]
print(Lst1)
print(Lst2)

print("Uniòn de listas...")
Lst3=[]
Lst3=Lst1+Lst2
print(Lst3)
```

Al ejecutar se obtiene:

```
[3, 19, 1, 12, 2]
[4, 55, 11, 3, 35, 6]
Uniòn de listas...
[3, 19, 1, 12, 2, 4, 55, 11, 3, 35, 6]
```

- Unir listas, pero generando una lista interna:

```
Lst1=[3,19,1,12,2]
Lst2=[4,55,11,3,35,6]
Lst1.append(Lst2)
print(Lst1)
```

Al ejecutar se obtiene:

```
[3, 19, 1, 12, 2, [4, 55, 11, 3, 35, 6]]
```

- Unir listas, pero generando solo listas internas:

```
Lst1=[3,19,1,12,2]
Lst3=[4,55,11,3,35,6]
Lst5=[Lst1,Lst3]
print(Lst5)
```

Al ejecutar se obtiene:

```
[[3, 19, 1, 12, 2], [4, 55, 11, 3, 35, 6]]
```

- Observe los ejercicios mostrados y luego saque sus conclusiones:

```
Lst1=[3,19,1,12,2]
print(Lst1)

Lst2=Lst1[:]
print("Caso 1:",Lst2)

Lst3=Lst1[0:3]
print("Caso 2:",Lst3)

Lst4=Lst1[1:-1]
print("Caso 3:",Lst4)

Lst5=Lst1[1:-2]
print("Caso 4:",Lst5)

Lst6=Lst1[:3]
print("Caso 5:",Lst6)
```

Al ejecutar, se mostrará lo siguiente en cada caso:

```
[3, 19, 1, 12, 2]
Caso 1: [3, 19, 1, 12, 2]
Caso 2: [3, 19, 1]
Caso 3: [19, 1, 12]
Caso 4: [19, 1]
Caso 5: [3, 19, 1]
```

h. Operadores in y not.

Estos operadores nos pueden quitar un gran peso de encima cuando estamos programando, ya que permiten ubicar un elemento en una lista, de ese modo no estamos generando códigos internos de búsqueda.

```
Lst1=[3,19,1,12,2,45,11,7]
if (7 in Lst1):
    print("Si se encuentra...!!!")
else:
    print("NO se encuentra...!!!")
```

Al ejecutar se muestra:

```
Si se encuentra...!!!
```

3. Desarrollo de ejercicios tipo

Los ejercicios que veremos a continuación operan mediante el uso de listas, están nivelados en cuanto a la dificultad para que se comprenda lo mejor posible.

Consideraciones:

Para el desarrollo de los ejercicios en esta parte del curso vamos a usar el módulo random de Python con la función asociada randint().

De esa manera evitaremos ingresar número de manera independiente y los generaremos al azar.

La función randint(), permite generar número al azar indicando un límite inferior y otro superior.

No se usará en todos los programas.

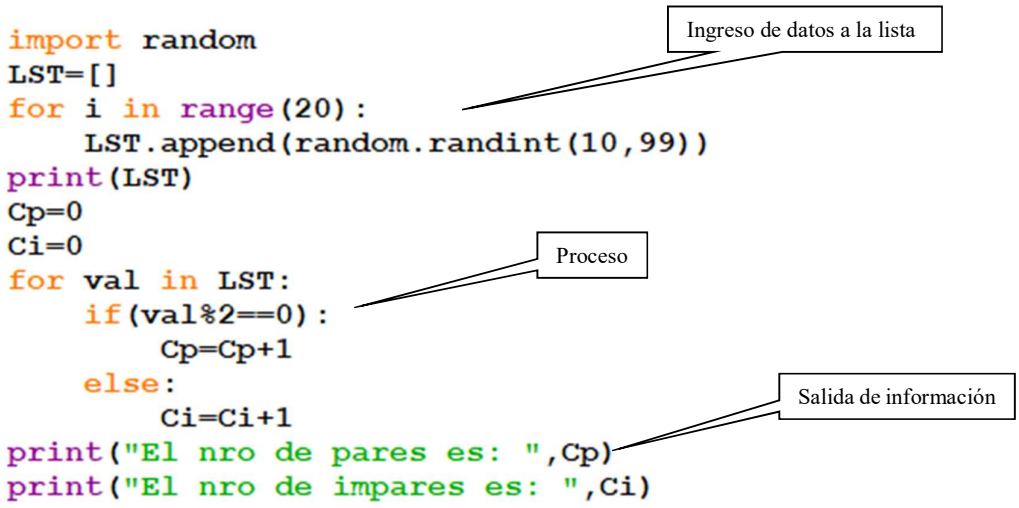
1. Problema Prg1

Desarrolle un algoritmo que permita ingresar 20 números en una lista, para luego verificar de manera particular cuantos fueron pares e impares.

Solución:

Este ejercicio indica ubicar números pares e impares, cuyo concepto ya se manejó anteriormente, la idea es usar listas en estos problemas ahora y ver cómo se tratan los datos. Vamos asumir la generación de números entre 10 y 99.

Programa:




```
import random
LST=[]
for i in range(20):
    LST.append(random.randint(10,99))
print(LST)
Cp=0
Ci=0
for val in LST:
    if(val%2==0):
        Cp=Cp+1
    else:
        Ci=Ci+1
print("El nro de pares es: ",Cp)
print("El nro de impares es: ",Ci)
```

Al ejecutar el programa, se muestra lo siguiente:

```
[78, 69, 65, 72, 49, 93, 88, 19, 45, 43, 45, 41, 91, 2
3, 37, 70, 48, 16, 42, 76]
El nro de pares es:  8
El nro de impares es:  12
```

Para activar la función randint(), que permite autogenerar números, se debe importar el módulo al que pertenece (random), por ello al inicio del código se muestra: `import random`

Por otro lado, podrá observar que al momento de diseñar el código este está separado por bloques de trabajo: entrada de datos, proceso y salida de información, sin embargo, se puede hacer varias cosas al mismo tiempo, en realidad depende como esté organizado su código, observar:

```
import random
Cp=0
Ci=0
LST=[]
for i in range(20):
    N=random.randint(10,99)
    LST.append(N)
    if(N%2==0):
        Cp=Cp+1
    else:
        Ci=Ci+1
print(LST)
print("El nro de pares es: ",Cp)
print("El nro de impares es: ",Ci)
```

2. Problema Prg2

Desarrolle un algoritmo que permita ingresar las notas de 40 alumnos en dos listas (Nota1 y Nota2), para luego hallar el promedio y almacenarlo en otra lista, finalmente indicar cuantos aprobados y desaprobados existen.

	Nota1		Nota2		Prom
0	12	0	19	0	
1	15	1	14	1	
2	17	2	18	2	
3	8	3	18	3	
4	11	4	11	4	
5	7	5	14	5	
6	8	6	11	6	
7	19	7	17	7	
8	16	8	11	8	

Solución:

Este ejercicio requiere ingresar o generar 40 pares de notas, un par para cada alumno, luego se debe halla el promedio y cargarlo en la lista Prom, para finalmente evaluar dicha lista y definir el número de aprobados y desaprobados.

Programa:

```
import random
Ca=0
Cd=0
LsT1=[];LsT2=[];LsTP=[]
for i in range(40):
    LsT1.append(random.randint(0,20))
    LsT2.append(random.randint(0,20))
print(LsT1)
print()
print(LsT2)
print()
for i in range(40):
    LsTP.append((LsT1[i]+LsT2[i])/2)
print(LsTP)
for valor in LsTP:
    if(valor>12.5):
        Ca=Ca+1
    else:
        Cd=Cd+1
print("Aprobados es: ",Ca)
print("Desaprobados es: ",Cd)
```

Al ejecutar el programa, se muestra lo siguiente:

```
[6, 3, 19, 20, 4, 15, 6, 5, 13, 1, 3, 15, 8, 19, 11, 1
7, 19, 12, 5, 7, 18, 0, 14, 19, 13, 10, 5, 15, 15, 13,
0, 7, 6, 10, 18, 13, 9, 15, 10, 13]

[19, 14, 7, 10, 3, 1, 19, 4, 1, 19, 11, 5, 13, 13, 15,
20, 6, 2, 10, 14, 16, 3, 16, 13, 6, 17, 3, 13, 7, 19,
10, 15, 19, 1, 18, 5, 0, 19, 4, 3]

[12.5, 8.5, 13.0, 15.0, 3.5, 8.0, 12.5, 4.5, 7.0, 10.0
, 7.0, 10.0, 10.5, 16.0, 13.0, 18.5, 12.5, 7.0, 7.5, 1
0.5, 17.0, 1.5, 15.0, 16.0, 9.5, 13.5, 4.0, 14.0, 11.0
, 16.0, 5.0, 11.0, 12.5, 5.5, 18.0, 9.0, 4.5, 17.0, 7.
0, 8.0]
Aprobados es:  13
Desaprobados es:  27
```

En la codificación podrá observar que existen funciones print() solas, ello es solo para separar filas de impresión.

3. Problema Prg3

Ingresar 50 número en una lista (estos deberán ser de 2 dígitos positivos), luego deberá ingresar un número positivo y verificar cuantas veces se repite en la lista.

Solución:

En este caso se debe validar el ingreso del número a buscar, ya que nos indica que debe ser positivos, el resto es parte de procesos vistos antes.

Programa:

```
import random
LST=[]
for i in range(50):
    LST.append(random.randint(10,99))
print(LST)
Op=True
while Op:
    VBuscado=int(input("Ing un nùmero + a buscar:"))
    if (VBuscado>0):
        Op=False
c=0
for ValLst in LST:
    if(ValLst==VBuscado):
        c=c+1
print(f"Se duplicò el valor {VBuscado}:{c} veces")
```

Al ejecutar el programa, se muestra lo siguiente:

```
[60, 88, 40, 41, 21, 74, 62, 82, 32, 42, 63, 90, 56, 7
4, 54, 65, 98, 53, 40, 16, 34, 77, 14, 36, 82, 19, 43,
90, 76, 86, 92, 66, 23, 51, 67, 62, 75, 85, 78, 38, 57
, 33, 37, 25, 84, 28, 55, 37, 22, 97]
Ing un nùmero + a buscar:66
Se duplicò el valor 66:1 veces
```

4. Problema Prg4

Se van a ingresar 5 números positivos (No generarlos), luego indicar cuál fue el mayor y el menor de todos ellos, desarrollarlo de dos formas, la 1era usando solo la lógica de programación y en la 2da forma mediante métodos que tienen las listas.

Solución:

Para el ejercicio planteado, la solución lógica implica ubicar un valor en una variable, para luego condicionarlo hacia el resto de elementos de la lista; esto deberá realizarlo tanto para al mayor como para el menor valor.

Programa:
Usando la lógica de programación

```
Lst=[]
for i in range(5):
    Op=True
    while(Op):
        print("Ingrese un Nro positivo:")
        N=int(input())
        if(N>0):
            break
    Lst.append(N)
print(Lst)
My=Lst[0]
Mn=Lst[0]
for i in range(1,5):
    if(My<Lst[i]):
        My=Lst[i]
    if(My>Lst[i]):
        Mn=Lst[i]
print("El mayor es:",My)
print("El menor es:",Mn)
```

Al ejecutar el programa, se muestra lo siguiente:

```
Ingrese un Nro positivo:
58
Ingrese un Nro positivo:
400
Ingrese un Nro positivo:
3
Ingrese un Nro positivo:
500
Ingrese un Nro positivo:
4
[58, 400, 3, 500, 4]
El mayor es: 500
El menor es: 4
```

Usando métodos de las listas:

```
Lst=[]
for i in range(5):
    Op=True
    while(Op):
        print("Ingrese un Nro positivo:")
        N=int(input())
        if(N>0):
            break
    Lst.append(N)
print(Lst)
Lst.sort()
print("El mayor es:",Lst[4])
print("El menor es:",Lst[0])
```

Al ejecutar el programa, se muestra lo siguiente:

```
Ingrese un Nro positivo:
6
Ingrese un Nro positivo:
0
Ingrese un Nro positivo:
2
Ingrese un Nro positivo:
4
Ingrese un Nro positivo:
6
Ingrese un Nro positivo:
10
[6, 2, 4, 6, 10]
El mayor es: 10
El menor es: 2
```

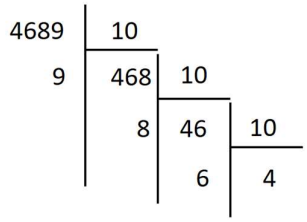
5. Problema Prg5

Desarrollar un algoritmo que permita ingresar 10 números positivos de N cifras en una lista (no generarlos), luego indicar cuantas cifras tiene cada número ingresado.

Solución:

Para el ejercicio planteado, la solución se relaciona al hecho de descomponer cada valor que tiene la lista, recordemos que no podemos usar funciones que no estén relacionadas al tema; por ende, debemos encontrar una solución lógica al problema.

Si realizamos divisiones sucesivas para descomponer (entre 10) podremos saber el nro. de dígitos contando las divisiones.



En el ejemplo vemos que se puede generar divisiones exactas hasta un dígito antes de que se llegue al final; de ese modo, al contador se le debería sumar un valor más para indicar el número de dígitos del número ingresado.

Programa:

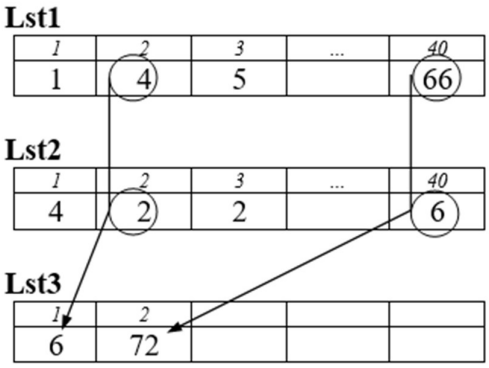
```
Lst=[]
for i in range(5):
    Op=True
    while(Op):
        print("Ingrese un Nro positivo:")
        N=int(input())
        if(N>0):
            break
    Lst.append(N)
print(Lst)
for valor in Lst:
    Op=True
    c=0
    if valor>=10:
        while(Op):
            valor=valor//10
            c=c+1
            if(valor<10):
                c=c+1
                break
    else:
        c=1
    print(f"Número de digitos de {valor}:{c}")
```

Al ejecutar el programa, se muestra lo siguiente:

```
Ingrese un Nro positivo:
564
Ingrese un Nro positivo:
2342423
Ingrese un Nro positivo:
4
Ingrese un Nro positivo:
56
Ingrese un Nro positivo:
7898
[564, 2342423, 4, 56, 7898]
Número de digitos de 5:3
Número de digitos de 2:7
Número de digitos de 4:1
Número de digitos de 5:2
Número de digitos de 7:4
```

6. Problema Prg6

Diseñe un algoritmo que permita ingresar 40 números a cada en sus listas respectivas, luego en un tercer arreglo se deberá cargar la suma solamente de aquellos números que sean múltiplos de 2 siempre en cuando estos se encuentren en la misma posición que también deberán ser pares. Adicionalmente ordenar el vector creado.



Solución:

En este caso vamos a autogenerar los 40 números en cada lista, para luego avaluar el valor (múltiplo de 2) y la posición (múltiplo de 2), ya que debe cumplirse ambos casos para poder generar la tercera lista, y finalmente mostrarla ordenada.

Programa:

```
import random
Lst1=[]
Lst2=[]
for i in range(40):
    Lst1.append(random.randint(10,99))
    Lst2.append(random.randint(10,99))
print(Lst1)
print()
print(Lst2)
print()
Lst3=[]
for i in range(40):
    if(i%2==0):
        if(Lst1[i]%2==0)and(Lst2[i]%2==0):
            Lst3.append(Lst1[i]+Lst2[i])
print(Lst3)
Lst3.sort()
print(Lst3)
```

Al ejecutar el programa, se muestra lo siguiente:

```
[24, 96, 72, 17, 70, 70, 14, 52, 17, 22, 56, 67, 32, 6
1, 41, 56, 61, 90, 30, 83, 77, 62, 31, 44, 33, 41, 31,
35, 36, 93, 31, 68, 65, 78, 25, 53, 48, 52, 16, 69]

[32, 64, 86, 24, 39, 98, 47, 39, 99, 87, 26, 96, 42, 6
7, 23, 75, 26, 16, 67, 69, 50, 11, 16, 26, 81, 11, 50,
44, 43, 66, 44, 18, 68, 13, 31, 94, 68, 97, 57, 65]

[56, 158, 82, 74, 116]
[56, 74, 82, 116, 158]
```

4. Actividad

A) Ejercicios en general

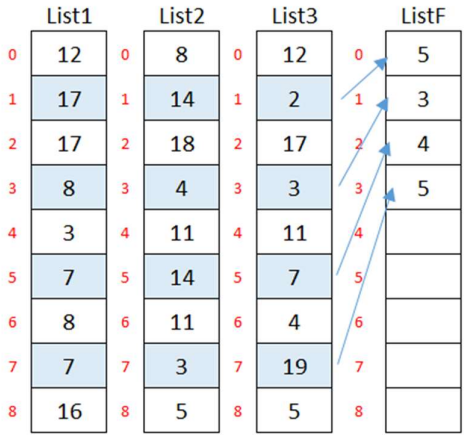
Desarrolle los siguientes ejercicios usando todas las sentencias e instrucciones desarrolladas a lo largo del curso, incluir a las listas en su desarrollo.

a) Desarrolle un algoritmo que permita ingresar 100 números positivos en una lista, luego deberá mostrar la tabla de multiplicar del 1 al 12 solo para los números ingresados que fueron pares.

b) Se ingresan 20 números positivos en una lista, muestre la factorial de cada uno de ellos.

c) Se generan 2 listas, cada una de 10 elementos (estos contienen letras), luego deberá realizar un algoritmo que permita evaluar ambas listas y cuando se cuente 3 vocales en cada caso, el programa deberá concluir; en caso de no encontrar las 3 vocales en ambas listas deberá indicar que no se puede concluir y forzar a finalizar el algoritmo.

d) Se deben generar 3 listas, cada una con 20 número autogenerados de 3 cifras, luego deberá evaluar los valores de las posiciones impares de cada lista, para finalmente en una cuarta lista agregar la suma del número de dígitos que tenía cada valor evaluado en la posición impar



e) Se ingresarán en una lista 400 notas, indicar cuantas son aprobatorias, cuantas son desaprobatorias, el promedio general de los aprobados, el promedio general de los desaprobados, el número de notas que fueron 20 y el número de notas que fueron 0.

5. Fuentes consultadas:

A) https://edube.org/learn/programming-essentials-in-python-part-1-spanish

B) Edison Zavaleta C. (2005). *Fundamentos de Programación*. Perú, Editorial Abaco-Lima.

C) https://pythones.net/programacion-en-python-fundamentos/