DATATYPE en PYTHON

link de GITHUB: https://github.com/villacis-cristian/ejercicios_en_clase_python.git

CRISTIAN ANDRES VILLACIS MENDOZA

DESARROLLO DE SOFTWARE

EJERCICIOS

1. Escribe un programa que muestre por pantalla la concatenación de un número y una cadena de caracteres. Para obtener esta concatenación puedes usar uno de los operadores explicados en este tema. Ejemplo: dado el número 3 y la cadena 'abc', el programa mostrará la cadena '3abc'.
2. Escribe un programa que muestre por pantalla un valor booleano que indique si un número entero *N* está contenido en un intervalo semiabierto *[a,b)*, el cual establece una cota inferior *a* (inclusive) y una cota superior *b* (exclusive) para *N*.
3. Escribe un programa que, dado dos strings *S1* y *S2* y dos números enteros *N1* y *N2*, determine si el substring que en *S1* se extiende desde la posición *N1* a la *N2* (ambos inclusive) está contenido en *S2*.
4. Dada una *lista* con elementos duplicados, escribir un programa que muestre una nueva *lista* con el mismo contenido que la primera pero sin elementos duplicados.
5. Escribe un programa que, dada una *lista* de strings *L*, un string *s* perteneciente a *L* y un string *t*, reemplace *s* por *t* en *L*. El programa debe mostrar la lista resultante por pantalla.
6. Escribe un programa que defina una *tupla* con elementos numéricos, reemplace el valor del último por un valor diferente y muestre la *tupla* por pantalla. Recuerda que las *tuplas* son inmutables. Tendrás que usar objetos intermedios.
7. Dada la lista [1,2,3,4,5,6,7,8] escribe un programa que, a partir de esta lista, obtenga la lista [8,6,4,2] y la muestre por pantalla.
8. Escribe un programa que, dada una tupla y un índice válido *i*, elimine el elemento de la tupla que se encuentra en la posición *i*. Para este ejercicio sólo puedes usar objetos de tipo tupla. No puedes convetir la *tupla* a una *lista*, por ejemplo.
9. Escribe un programa que obtenga la mediana de una *lista* de números. Recuerda que la mediana *M* de una lista de números *L* es el número que cumple la siguiente propiedad: la mitad de los números de *L* son superiores a *M* y la otra mitad son inferiores. Cuando el número de elementos de *L* es par, se puede considerar que hay dos medianas. No obstante, en este ejercicio consideraremos que únicamente existe una mediana.

EJERCICIO 01

In [3]:
```
numero = 3
cadena = 'abcdefhi'

resultado = str(numero) + cadena
print(resultado)
```

```
3abcdefhi
```

EJERCICIO 02

In [5]:
```
# Solicita al usuario ingresar el número a evaluar y lo convierte a entero
N = int(input("Ingrese el número N: "))

# Solicita el límite inferior del intervalo y lo convierte a entero
a = int(input("Ingrese el límite inferior a: "))

# Solicita el límite superior del intervalo y lo convierte a entero
b = int(input("Ingrese el límite superior b: "))

# Evalúa si N está en el intervalo [a, b) (a incluido, b excluido)
esta_en_intervalo = a <= N < b

# Imprime True si está en el intervalo, False si no
print(esta_en_intervalo)
```

```
False
```

EJERCICIO 03

In [8]:
```
# Solicita el primer string al usuario
S1 = input("Ingrese el string S1: ")

# Solicita el segundo string al usuario
S2 = input("Ingrese el string S2: ")

# Solicita la posición inicial para el substring y la convierte a entero
N1 = int(input("Ingrese la posición inicial N1: "))

# Solicita la posición final para el substring y la convierte a entero
N2 = int(input("Ingrese la posición final N2: "))

# Extrae el substring de S1 desde N1 hasta N2 (ambos incluidos)
substring = S1[N1:N2+1]

# Verifica si el substring está contenido en S2
esta_contenido = substring in S2

# Imprime el resultado de la verificación
print(esta_contenido)
```

```
True
```

EJERCICIO 04

In [9]:
```
# Lista original con elementos duplicados
lista_original = [1, 2, 2, 3, 4, 4, 5, 6, 6, 6]

# Convierte la lista a conjunto (elimina duplicados) y luego de vuelta a lista
lista_sin_duplicados = list(set(lista_original))  #set retira los elementos repetidos

# Imprime la lista sin elementos duplicados
print(lista_sin_duplicados)
```

```
[1, 2, 3, 4, 5, 6]
```

EJERCICIO 05

In [10]:
```
# Lista original de strings
L = ["hola", "mundo", "python", "programación"]

# Elemento a buscar y reemplazar
s = "python"

# Nuevo valor que reemplazará al elemento encontrado
t = "coding"

# Busca el índice donde se encuentra el elemento 's' en la lista
indice = L.index(s)

# Reemplaza el elemento en la posición encontrada con el nuevo valor 't'
L[indice] = t

# Imprime la lista modificada
print(L)
```

```
['hola', 'mundo', 'coding', 'programación']
```

EJERCICIO 06

In [11]:
```
# Tupla original (inmutable)
tupla_original = (1, 2, 3, 4, 5)

# Nuevo valor para el último elemento
nuevo_valor = 10

# Convierte la tupla a lista para poder modificarla
lista_temporal = list(tupla_original)

# Modifica el último elemento de la lista temporal
lista_temporal[-1] = nuevo_valor

# Convierte la lista modificada de vuelta a tupla
tupla_modificada = tuple(lista_temporal)

# Imprime la nueva tupla con el último elemento modificado
print(tupla_modificada)
```

```
(1, 2, 3, 4, 10)
```

EJERCICIO 07

In [12]:
```
# Lista original de números
lista_original = [1, 2, 3, 4, 5, 6, 7, 8]

# Primero invierte la lista [::-1], luego toma cada segundo elemento [::2]
lista_resultado = lista_original[::-1][::2]

# Imprime la lista resultante [8,6,4,2]
print(lista_resultado)
```

```
[8, 6, 4, 2]
```

EJERCICIO 08

In [13]:
```
# Tupla original de números
tupla_original = (10, 20, 30, 40, 50)

# Índice del elemento a eliminar
i = 2

# Crea nueva tupla concatenando:
# - Todos los elementos antes del índice i
# - Todos los elementos después del índice i
tupla_modificada = tupla_original[:i] + tupla_original[i+1:]

# Imprime la tupla resultante sin el elemento en posición i
print(tupla_modificada)
```

```
(10, 20, 40, 50)
```

EJERCICIO 09

In [14]:
```
# Función para calcular la mediana
def calcular_mediana(lista):
    # Ordena la lista de menor a mayor
        lista_ordenada = sorted(lista)   #sorted ordena la lista sin modificarla

    # Obtiene la cantidad de elementos
    n = len(lista_ordenada)

    # Si la cantidad es impar
    if n % 2 == 1:
        # La mediana es el elemento central
        mediana = lista_ordenada[n//2]
    else:
        # Si es par, toma el elemento anterior al centro
        mediana = lista_ordenada[n//2 - 1]

    # Devuelve el valor de la mediana
    return mediana

# Lista de ejemplo
numeros = [5, 2, 9, 1, 7, 6, 4, 3]

# Calcula la mediana llamando a la función
mediana = calcular_mediana(numeros)

# Imprime el resultado formateado
print(f"La mediana es: {mediana}")
```

```
La mediana es: 4
```

In [2]:
```
# Definición de una lista llamada 'L' con números desordenados
L = [21, 9, 1, 5, 5, 10, 3, 15]

# Ordena la lista 'L' de menor a mayor usando el método sort()
# Después de esta línea, L será [1, 3, 5, 5, 9, 10, 15, 21]
L.sort()

# Calcula la mediana de la lista:
# - len(L) devuelve la longitud de la lista (8 en este caso)
# - len(L)//2 hace división entera por 2 (resultado: 4)
# - L[4] accede al elemento en la posición 4 de la lista (recordando que en Python los índices comienzan en 0)
# Para listas con número par de elementos, esta fórmula da el elemento justo después del centro
mediana = L[len(L)//2]

# Imprime el valor de la mediana calculada (que será 9 en este caso)
print(mediana)
```

```
9
```

In [ ]:
```

```