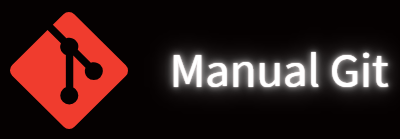
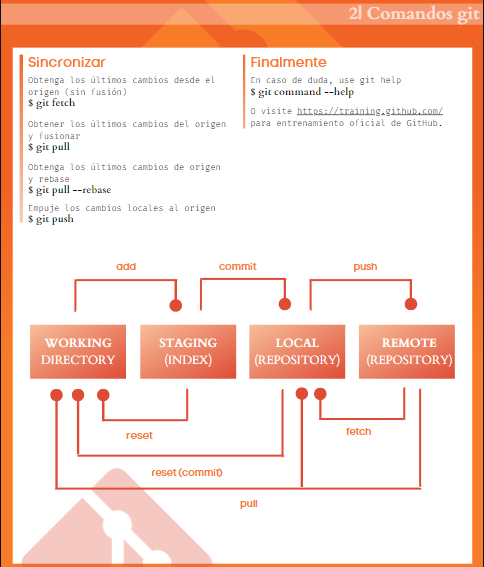
TALLER DE CODIGO

Escuela DE CODIGO

 PILARES

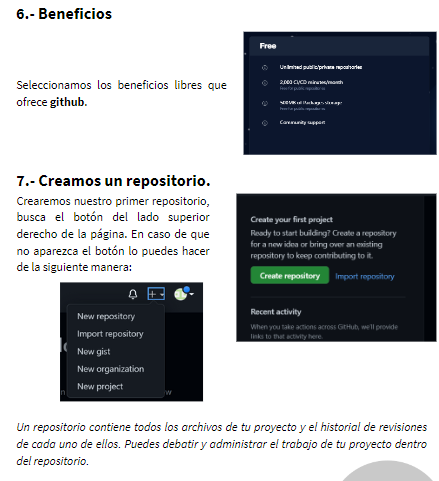
Recursos

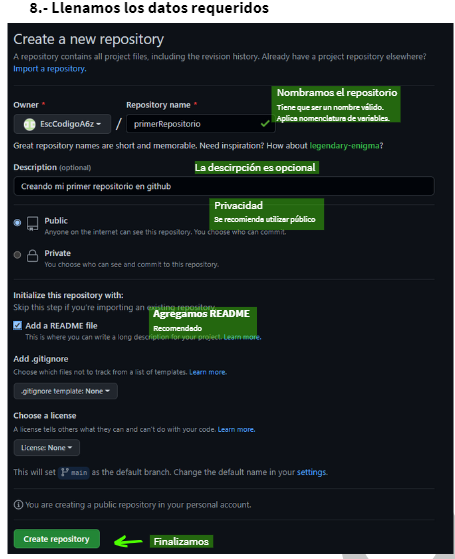




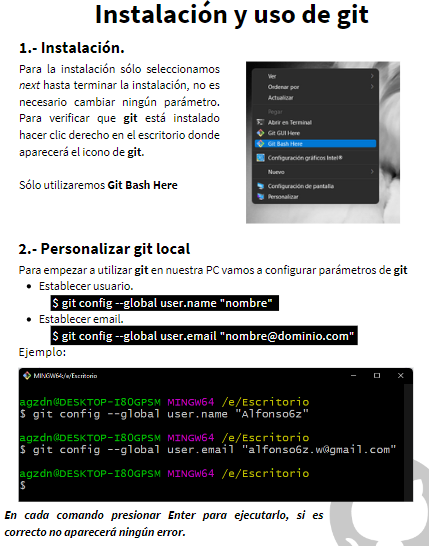


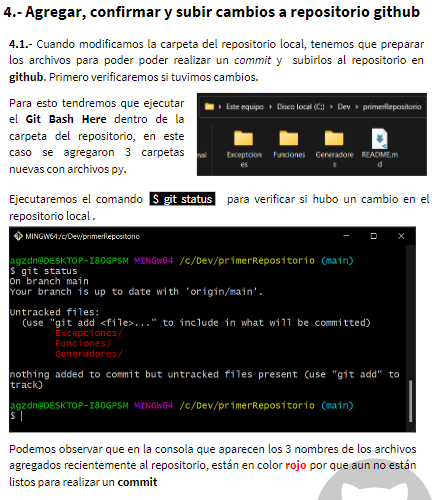




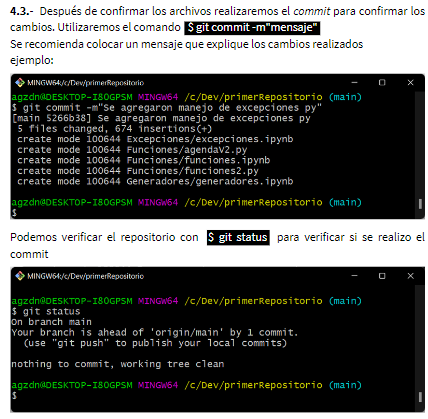
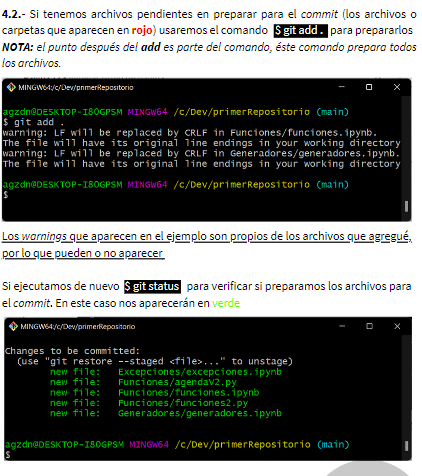


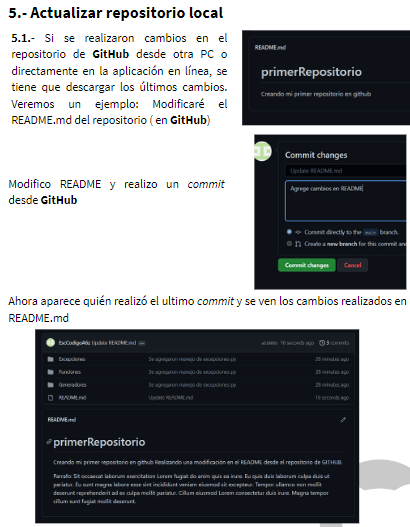


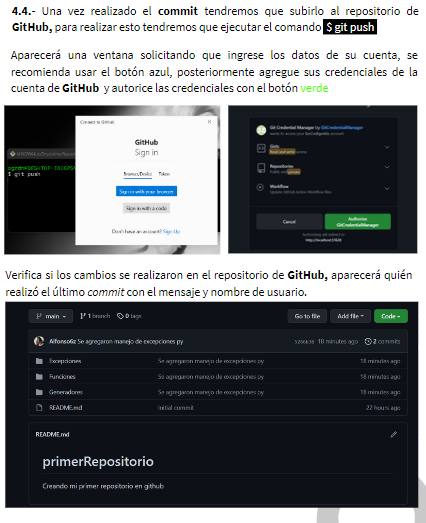


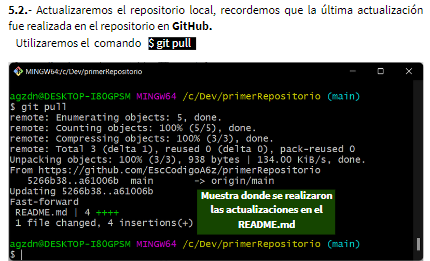












METODOS DE ORDENACIÒN

Los métodos de ordenación son algoritmos que se utilizan para ordenar elementos en una lista o arreglo en un orden específico. Hay diferentes métodos de ordenación disponibles, pero los dos tipos principales son los métodos de ordenación directos y los métodos de ordenación logarítmicos.

Métodos de ordenación directos:

1. Ordenamiento por selección (Selection sort): Este método de ordenación selecciona el elemento mínimo en cada iteración y lo mueve a la posición correcta en la lista. Este proceso se repite hasta que todos los elementos estén ordenados.

La ordenación por selección selecciona repetidamente el elemento mínimo restante y lo coloca al comienzo de la lista ordenada. La sublista de elementos restantes se reduce después de cada iteración.

Pseudo código:

ini

Para i en rango (0, longitud de la lista):

min\_idx = i

Para j en rango (i+1, longitud de la lista):

Si lista**[j]** < lista**[min\_idx]**:

min\_idx = j

Intercambiar lista**[i]** y lista**[min\_idx]**

Python:

python

def ordenacion\_seleccion(lista):

n = len(lista)

for i in range(n):

min\_idx = i

for j in range(i+1, n):

if lista[j] < lista[min\_idx]:

min\_idx = j

lista[i], lista[min\_idx] = lista[min\_idx], lista[i]

return lista

Ejemplo:

python

lista = [64, 34, 25, 12, 22, 11, 90]

print(ordenacion\_seleccion(lista)) # salida: [11, 12, 22, 25, 34, 64, 90]

Métodos de ordenación por logaritmos, también conocidos como métodos de ordenación eficientes, son aquellos que se basan en el uso de estructuras de datos complejas para ordenar rápidamente grandes conjuntos de datos. Uno de los algoritmos más populares es la ordenación rápida.

1. Ordenamiento por inserción (Insertion sort): Este método de ordenación toma un elemento de la lista y lo compara con los elementos que le preceden. Si el elemento es menor, se desplaza a la izquierda hasta que encuentre su posición correcta.
2. Ordenamiento por burbuja (Bubble sort): Este método de ordenación compara elementos adyacentes y los intercambia si están en el orden incorrecto. Este proceso se repite varias veces hasta que la lista esté ordenada.

Ordenación de Burbuja:

La ordenación de burbuja compara repetidamente los elementos adyacentes de una lista y los intercambia si están en el orden incorrecto. La lista se recorre varias veces hasta que no hay intercambios adicionales necesarios, lo que indica que la lista está ordenada.

Pseudo código:

less

Para i en rango (0, longitud de la lista):

Para j en rango (0, longitud de la lista-i-1):

Si lista[j] > lista[j+1]:

Intercambiar lista[j] y lista[j+1]

Python:

python

def ordenacion\_burbuja(lista):

n = len(lista)

for i in range(n):

for j in range(0, n-i-1):

if lista[j] > lista[j+1]:

lista[j], lista[j+1] = lista[j+1], lista[j]

return lista

Ejemplo:

python

ista = [64, 34, 25, 12, 22, 11, 90]

print(ordenacion\_burbuja(lista)) # salida: [11, 12, 22, 25, 34, 64, 90]

Métodos de ordenación logarítmicos:

1. Ordenamiento rápido (Quicksort): Este método de ordenación divide la lista en dos sub-listas y ordena cada una de ellas recursivamente. El algoritmo selecciona un pivote, que divide la lista en dos sub-listas, una con elementos menores que el pivote y otra con elementos mayores. Luego, las sub-listas se ordenan recursivamente.
2. Ordenamiento por mezcla (Merge sort): Este método de ordenación divide la lista en dos sub-listas de tamaño similar y las ordena recursivamente. Luego, las sub-listas ordenadas se mezclan para formar una lista ordenada completa.
3. Ordenamiento de montículo (Heapsort): Este método de ordenación utiliza una estructura de datos llamada montículo para ordenar los elementos en la lista. El montículo es una estructura de árbol binario en la que cada nodo es mayor que sus hijos. Los elementos se agregan al montículo y luego se eliminan en orden para formar una lista ordenada.

Ciclo For

Ordenar  
colocar los elementos de una lista bajo un criterio

Para recorrerla (elemento a elemento, de dos en dos tres en tres, de adelante para atrás. E.t.c) se usa el ciclo For

Pseudo código: para cada elemento de la lista realizar una acción con cada elemento.

Ejemplos.

* Para cada elemento de los números del 1 al 10 sumar 1 +1 a cada elemento

Del 1 + 1 = 2 del 10, 1= 11

Resultado 2,3,4,5,6,7,8,9,10,11

* Con restricciones (en caso de no querer usar todos los elementos)

Para cada elemento de los números de 1 al 10 seleccionar desde el segundo hasta el penúltimo y sumar uno a cada uno de estos elementos

Sumamos 1 del elemento 2 al 9, y el primer y último quedan igual

1,3,4,5,6,7,8,9,10,10

* Del principio al fin o del fin al pricipio

Para cada elemento de los números del 1 al 10 desde el ultimo hasta el primero, sumar uno (para recorrerlo al revés)

Métodosde ordenación

* **Internos (de arreglos)**

Interno ya que los componentes se encuentran en la memoria RAM o interna

* + - * **Directos**

Cantidad de datos pequeña. (n^2)

Ordenación por:

* **Intercambio:**
* Método de ordenación interno directo de intercambio

Buscar cada vez el menor elemento del conjunto y ponerlo al principio

En cada paso se compara el primer elemento del conjunto x[ i ] con los demás elementos del mismo

x[ j ] ( j=i+1 . . n) y cuando x[ i ] es mayor que x[ j ] se intercambian sus valores cuando se termina de recorrer el arreglo, el proceso nos garantiza que en x[ i ] está el menor elemento del conjunto

independientemente de los datos almacenados NO existe un peor, mejor, o caso promedio y su complejidad siempre será 0(n^2)

* Métodos de Intercambio
* De la Burbuja: comparar 2 elementos adyacentes para en forma ya sea descuente o asenté colocar de mayor a menor

Método burbuja 1

1. Generar un ciclo que inicie desde 1 hasta el número de elementos de arreglos
2. Generar un segundo ciclo dentro del anterior que inicie desde 0 hasta el numero de elementos menos 2.
3. Dentro del segundo ciclo debe existir una comparación que determine el tipo de ordenamiento (ascendente, decente) entre el primer elemento (posición generada por el segundo ciclo ) y segundo elemento ( el que le sigue ) si la respuesta a la condición es verdadera se realiza un intercambio entre los dos elementos.
4. Para realizar el intercambio se genera un almacenamiento temporal el cual guarda el dato del primer elemento, el segundo elemento toma el lugar del primero y en el lugar del segundo se coloca lo que contiene el almacenamiento temporal.

Una vez que los ciclos terminan la estructura debe quedar ordenada.

Se considera el peor de los casos ya que si la lista es de 100 números se requieren 9900 comparaciones

Ciclo 1.- desde 1 hasta n

Ciclo 2.- desde 0 hasta n – 2

Dentro de Ciclo 2 si ꓥ ꓦ comparar E1C2 y E2C2 si es V

Método burbuja 2

1. Generar un ciclo que inicie desde 0 hasta el número de elementos menos 2
2. Generar un ciclo desde el valor del ciclo anterior mas 1 hasta el numero de elementos menos 1
3. Dentro del segundo ciclo debe existir una comparación que determina el tipo de ordenamiento (ascendente, descendente) entre el primer elemento (posición generada por el primer ciclo) y el segundo elemento (posición generada por el segundo ciclo), si la respuesta a la condición es verdadera se realiza un intercambio entre los 2 elementos.
4. Para realizar el intercambio se realiza un almacenamiento temporal el cual guarda el dato del primer elemento

* Quicksort:
* Shellsort
* Radix
* **Inserción:**
* Método de ordenación interno directo de inserción
* **Selección:**
* Método de ordenación interno directo de
  + - * **Logarítmicos**

Complejos difíciles de entender números grandes

(n \* log n )

Externos (de archivos)