Sebastian Oña

Taller nro1

Algoritmo 1

la sumatoria $1+1/2+1/4+1/8... $ tal que el error absoluto $ e_{abs}< 10^{1} $

```
### Algoritmo 1
suma = 0
termino = 1
n = 0
while abs(termino) >= 10**-1:
    suma += termino
    n += 1
    termino = 1 / (2 ** (n+1))
    n += 1

print("La suma de la serie es:", suma)
print("El número de términos sumados es:", n)
```

Algoritmo 2

```
#Algoritmo 2
def bubble_sort(a):
    n = len(a)
    for i in range(n):
        swapped = False
        for j in range(1, n - i):
            if a[j] < a[j - 1]:
                a[j], a[j - 1] = a[j - 1], a[j]
                swapped = True
        if not swapped:
            break
    return a
# V2
arr = [-1, 0, 4, 5, 6, 7]
print(bubble_sort(arr))
```

```
[-1, 0, 4, 5, 6, 7]
```

Para ordenar una lista de 100_000 elementos con valores aleatorios entre -200 y 145

```
import random
import time

# Generar lista aleatoria de 100000 elementos entre -200 y 145
```

```
lista = [random.randint(-200, 145) for _ in range(100000)]

print("Lista original se mostrara los 100 primeros elementos porq se
saturaria la terminal:")
print(lista[:100])

# Algoritmo de Bubble Sort
def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        # Si no hay intercambios en una pasada, la lista ya está
ordenada
        swapped = False
        for j in range(n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True
        if not swapped:
            break

# Medir tiempo de ordenamiento
inicio = time.time()
bubble_sort(lista)
fin = time.time()

print("\nLista ordenada (primeros 100 elementos):")
print(lista[:100])
print(f"\nTiempo de ejecución: {fin - inicio:.2f} segundos")
```

Pregunta 1 del taller

Algoritmo 2 modificado para que cuente el numero de comparaciones realizadas al ordenar la serie 5,4,3,2,1

```
lista = [5, 4, 3, 2, 1]
comparaciones = 0

n = len(lista)
for i in range(n - 1):
    for j in range(n - i - 1):
        comparaciones += 1
        if lista[j] > lista[j + 1]:
            lista[j], lista[j + 1] = lista[j + 1], lista[j]  #
Intercambio

print("Lista ordenada:", lista)
print("Número de comparaciones:", comparaciones)
```

Algoritmo 3

n	fib(n)
0	0
1	1
2	1
3	2
4	3
5	5
6	8
...	...
$ n= 11 | ?= 89) \vee ¿ n= 84 | ?= 1605 .....) \vee ¿ n= 1531 $	?= 4079....

```
#Algoritmo 3
def Fibonacci(n):
    if n == 0:
        return 0
    else:
        x = 0
        y = 1
        for i in range(1, n):
            z = x + y
            x = y
            y = z
        return y

n = int(input("Ingrese el valor de n: "))
resultado = Fibonacci(n)
print(f"Para n = {n}, y = {resultado}")
```

Pregunta 2 del taller

Usando el Algoritmo 3 y la aritmetica de redondeo con 3 cifras

```
import math

# Valor real del número áureo
phi = (1 + math.sqrt(5)) / 2

# Generar términos de Fibonacci con redondeo a 3 cifras
fib = [1.00, 1.00]

# Redondear a 3 cifras significativas
def redondear(x):
    return float(f"{x:.3g}")
```

```
# Crear secuencia con redondeo
for i in range(2, 20):
    nuevo = redondear(fib[-1] + fib[-2])
    fib.append(nuevo)

# Calcular las razones y su error relativo
for i in range(1, len(fib) - 1):
    r = redondear(fib[i + 1] / fib[i])
    error_relativo = abs((r - phi) / phi)
    print(f"i={i:2d}  y_i={fib[i]:7.3f}  r={r:6.3f}
error={error_relativo:.2e}")
    if error_relativo < 1e-5:
        print(f"\n Desde la iteración i={i}, el error relativo es
menor que 1e-5")
        break
```

Graficar Algoritmo 4

- El valor de la serie $fib(n)$
- El valor cociente

$ \phi \rightarrow \frac {fib(n)} {fib(n-1)} \approx 1.618 $ numero áureo

n	$ \frac {fib(n)} {fib(n-1)}$
0	$1/1 =1 $
1	$2/1 =2 $
2	$3/2 =1.5 $
3	$5/3 =1.666 $
4	$8/5 =1.6 $
5	$13/8 =1.625 $
6	$21/13 =1.615 |)…|…)∨¿ \infty $

```
import matplotlib.pyplot as plt

# Generar la serie de Fibonacci
def fibonacci(n):
    fib = [0, 1]
    for i in range(2, n):
        fib.append(fib[-1] + fib[-2])
    return fib

# Cantidad de términos
n = 20
fib = fibonacci(n)

razones = [fib[i] / fib[i - 1] for i in range(2, len(fib))]

indices = list(range(2, len(fib)))
```

```
phi = (1 + 5 ** 0.5) / 2

# Graficar las razones y el número áureo
plt.figure()
plt.plot(indices, razones, marker='o', label='fib(n)/fib(n-1)')
plt.axhline(y=phi, color='r', linestyle='--', label='Número áureo
(φ)')
plt.title('Aproximación al número áureo con la serie de Fibonacci')
plt.xlabel('n')
plt.ylabel('fib(n) / fib(n-1)')
plt.legend()
plt.grid(True)
plt.show()
```