

1 Enlace GIB HUB [CODIGOS] :
<https://github.com/fresitahappy/CODIGOS-1-10>



**UNIVERSIDAD NACIONAL DEL
ALTIPLANO**
INGENIERÍA ESTADÍSTICA E INFORMÁTICA

METODOS DE OPTIMIZACION

ACTIVIDAD 02

ESTUDIANTE : Mayta Cama Nataly Keila
CODIGO : [227221]
FECHA DE ENTREGA DE TRABAJO :01/10/2024

PROBLEMA 1 Y ENUNCIADO :

Un algoritmo necesita procesar datos en lotes. Cada lote requiere x MB de memoria, pero la capacidad total de memoria disponible es de 1024 MB. El algoritmo puede procesar un máximo de 8 lotes. El objetivo es maximizar la cantidad de datos procesados, pero cada lote más allá del quinto reduce su eficiencia en un 20%.

FORMULAS :

$$\text{Maximizar: } D(x) = \sum_{i=1}^x (\text{lote}_i) \quad (1)$$

Sujeto a:

- Total de memoria disponible:

$$n \cdot x \leq 1024$$

- Número máximo de lotes:

$$n \leq 8$$

Función del problema:

- Para los primeros 5 lotes: La fórmula es simple, cada lote procesa una cantidad fija de datos, que es x , por lo tanto, la cantidad total de datos procesados para n lotes $5 \cdot x$.
- Para los lotes adicionales (del 6 al 8), la eficiencia es del 80%, es decir, $0.8 \cdot x$ por lote.

Entonces, si procesamos n lotes, donde $n \leq 5$, los datos procesados son $n \cdot x$. Si $n > 5$, los datos procesados son:

$$D(n) = \begin{cases} n \cdot x & \text{si } n \leq 5 \\ 5 \cdot x + (n - 5) \cdot 0.8 \cdot x & \text{si } n > 5 \end{cases}$$

Solución

El algoritmo deberá probar distintos valores de x y n dentro de las restricciones, evaluando en cada caso cuántos datos pueden procesarse de manera óptima.

Ejemplo: $n = 5$

$$5x \leq 1024 \implies x \leq 204.8 \text{ MB por lote}$$

Si $x = 204.8$, se procesarán:

$$5 \cdot 204.8 = 1024 \text{ MB de datos}$$

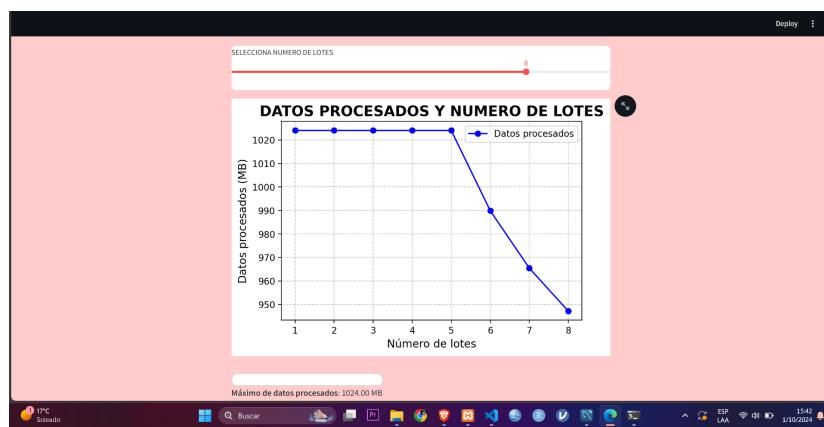
INTERPRETACION DEL PROBLEMA.

La solución óptima es procesar 5 lotes de 204.8 MB, lo que maximiza los datos procesados sin perder eficiencia.

CAPTURA DEL CODIGO

```
Foto Edit Selection View Go Run Terminal Help ⏎ ⌘ Search
[ejercic平py] X
C:\Users\Usuario\Downloads\trabajo 2 final > ejercic平py > ...
41 # Parámetros
42 memoria = 1024
43 eficiencia_re = 0.8
44
45 # Función para calcular los datos procesados
46 def datos_procesados(n, x):
47     if n <= 5:
48         return n * x
49     else:
50         return 5 * x + (n - 5) * eficiencia_re * x
51
52 # Mostrar el gráfico primero
53 st.markdown(<div class="container">, unsafe_allow_html=True)
54
55 # Lado Izquierdo: Gráfico
56 st.markdown(<div class="left-side">, unsafe_allow_html=True)
57 max_lotes_input = st.slider("SELECCIONA NÚMERO DE LOTES", 1, 10, 8)
58 lotes = np.arange(1, max_lotes_input + 1)
59 datos = np.array([datos_procesados(n, memoria / n) for n in lotes])
60
61 fig, ax = plt.subplots(figsize=(6, 4))
62 ax.plot(lotes, datos, marker='o', linestyle='-', color='b', label='Datos procesados')
63 ax.set_xlabel('Número de lotes', fontsize=12, fontweight='bold')
64 ax.set_ylabel('Datos procesados (MB)', fontsize=12, fontweight='bold')
65 ax.set_yticks(np.arange(0, 1000, 100))
66 ax.grid(True, linestyle='--', alpha=0.7)
67
68 st.pyplot(fig)
69 st.markdown(</div>, unsafe_allow_html=True)
70
71 # Lado Derecho: Resultados en un cuadro ordenado
72 st.markdown(<div class="right-side">, unsafe_allow_html=True)
73
74 # Mostrar los resultados: máximo, mínimo y valores en cada punto
75 max_dato = np.max(datos)
76 min_dato = np.min(datos)
77
```

CAPTURA DEL RESULTADO



PROBLEMA 2 Y ENUNCIADO:

Un sistema distribuido tiene 20 nodos. Cada nodo puede procesar x peticiones por segundo. El sistema en su conjunto no puede procesar más de 400 peticiones por segundo debido a limitaciones de red. Maximiza el número de peticiones procesadas sin exceder la capacidad de la red.

$$\text{Maximizar: } P(x) = \sum_{i=1}^x \text{ peticiones} \quad (2)$$

Sujeto a:

$$\begin{aligned} x &\leq 20 \\ \sum_{i=1}^{20} P(x) &\leq 400 \end{aligned}$$

Restricciones que el problema pide:

- Capacidad total de procesamiento:

$$n \cdot x \leq 400$$

- Número de nodos:

$$n = 20$$

Función

Maximizar el número total de peticiones procesadas por el sistema, P , donde:

$$P = n \cdot x$$

Sujeto a las restricciones:

$$n \cdot x \leq 400$$

$$n = 20$$

Solución

Sustituyendo $n = 20$ en la restricción:

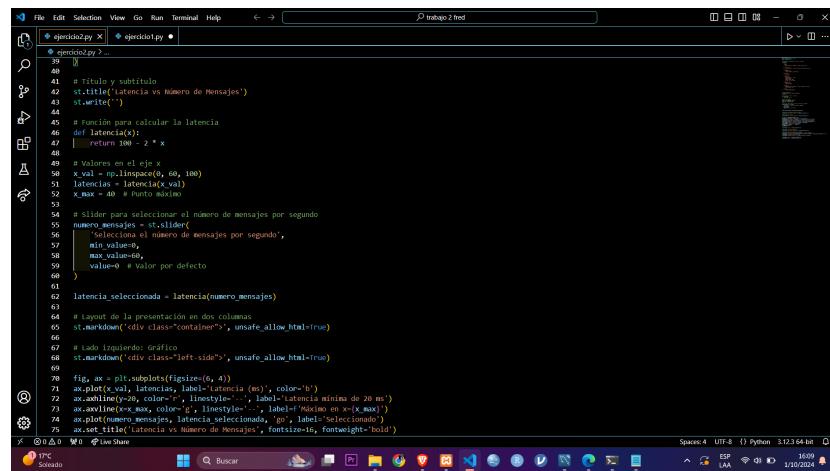
$$20 \cdot x \leq 400$$

$$x \leq \frac{400}{20} = 20 \text{ peticiones/segundo por nodo}$$

INTERPRETACION DEL PROBLEMA :

Cada nodo debe procesar $x = 20$ peticiones por segundo, lo que maximiza el número total de peticiones.

CAPTURA DEL CODIGO

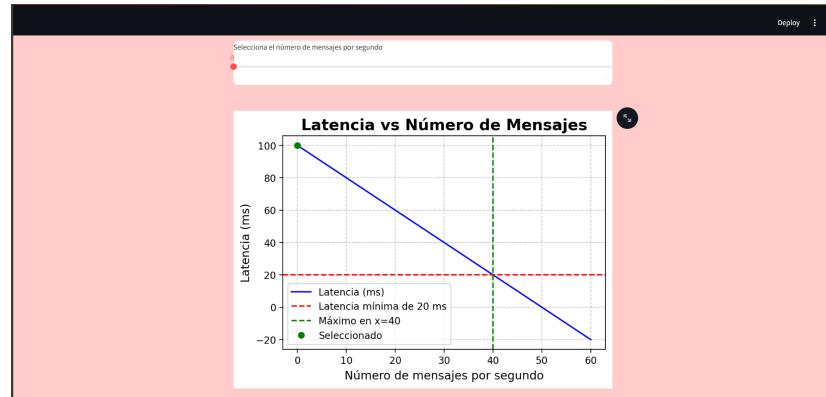


```

File Edit Selection View Go Run Terminal Help ↻ → trabajo 2 fred
ejercicio.py X ejercicio.py ●
ejercicio.py >
39
40
41 # Título y subtítulo
42 st.title('Latencia vs Número de Mensajes')
43 st.write('')
44
45 # Función para calcular la latencia
46 def latencia(x):
47     return 100 - 2 * x
48
49 # Valores en el eje x
50 x_val = np.linspace(0, 60, 100)
51 latencias = latencia(x_val)
52 x_max = 40 # Punto máximo
53
54 # Slider para seleccionar el número de mensajes por segundo
55 numero_mensajes = st.slider(
56     'Selecciona el número de mensajes por segundo',
57     min_value=0,
58     max_value=60,
59     value=40 # Valor por defecto
60 )
61 latencia_seleccionada = latencia(numero_mensajes)
62
63 # Layout de la presentación en dos columnas
64 st.markdown("<div class='container'>", unsafe_allow_html=True)
65
66 # Lado izquierdo: Gráfico
67 st.markdown("<div class='left-side'>", unsafe_allow_html=True)
68
69 fig, ax = plt.subplots(figsize=(4, 4))
70 ax.plot(x_val, latencias, label='latencia (ms)', color='blue')
71 ax.axhline(y=20, color='red', linestyle='--', label='latencia mínima de 20 ms')
72 ax.axvline(x=40, color='green', linestyle='--', label='máximo en x=40')
73 ax.set_xlim(0, 60)
74 ax.set_xlabel('Número de mensajes por segundo')
75 ax.set_title('Latencia vs Número de Mensajes', fontsize=16, fontweight='bold')

```

CAPTURA DEL RESULTADO



PPROBLEMA 3 Y ENUNCIADO :

Un script de Python tarda $5x+2$ segundos en procesar x datos. Por cada dato adicional, el tiempo de ejecución crece linealmente. Sin embargo, el sistema tiene un límite de tiempo de ejecución de 50 segundos. ¿Cuál es el número máximo de datos que puede procesar el script?

$$\text{Maximizar: } D(x) \quad (3)$$

Sujeto a:

$$T(x) = 5x + 2 \leq 50$$

Restricciones del problema :

$$5x + 2 \leq 50$$

Función

Maximizar x , es decir, encontrar el número máximo de datos que se pueden procesar sin exceder el tiempo permitido.

Solución

Despejamos x de la ecuación:

$$\begin{aligned} 5x + 2 &\leq 50 \\ 5x &\leq 50 - 2 \\ 5x &\leq 48 \\ x &\leq \frac{48}{5} = 9.6 \end{aligned}$$

Dado que x debe ser un número entero, el valor máximo de x es 9.

INTERPRETACION DEL RESULTADO :

El número máximo de datos que el script puede procesar sin exceder los 50 segundos es $x = 9$.

CAPTURA DEL CODIGO

```
# ejercicio4.py
# Ejercicio 4: "Tiempo de ejecución del script vs número de datos procesados"
# Slider para límite de tiempo
limite_tiempo = st.slider('Límite máximo de tiempo (segundos)', 10, 100, 50)
st.write("")

# Función para calcular el tiempo de ejecución
def tiempo_ejecucion(x):
    return x * x + 10

# Valores en el eje x
x_val = np.arange(0, 10, 1)
tiempos = tiempo_ejecucion(x_val)

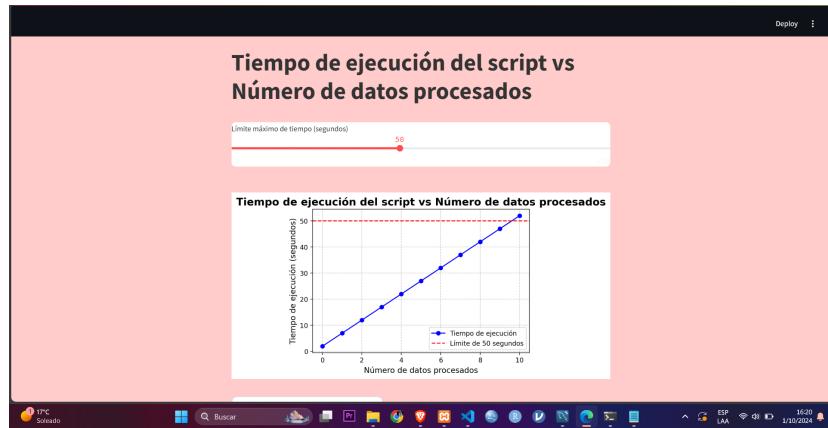
# Layout de la presentación en dos columnas
st.markdown("<div class='container'>", unsafe_allow_html=True)
st.markdown("<div class='left-side'>", unsafe_allow_html=True)
st.markdown("<div class='right-side'>", unsafe_allow_html=True)
st.markdown("</div>", unsafe_allow_html=True)

# Lado izquierdo: Gráfico
st.pyplot(fig)

# Lado derecho: Mostrar resultados
st.markdown("<div class='right-side'>", unsafe_allow_html=True)
st.markdown("Mostrar el límite de tiempo", unsafe_allow_html=True)

# Mostrar el límite de tiempo
st.write(f'Límite máximo de tiempo (segundos) {limite_tiempo}')
```

CAPTURA DEL RESULTADO



PROBLEMA 4 Y ENUNCIADO :

Un servidor web procesa x peticiones por segundo, y el uso de CPU sigue la fórmula $2x^2 + 10x$. La CPU no puede exceder el 80 % de uso. Minimiza el uso de CPU sin caer por debajo del umbral de procesamiento de 10 peticiones por segundo.

Función objetivo:

$$\text{Minimizar } U(x) = 2x^2 + 10x$$

Sujeto a:

$U(x) \leq 80$ (Límite de uso de CPU)
 $x \geq 10$ (Debe procesar al menos 10 peticiones por segundo)

$$\text{Minimizar: } U(x) = 2x^2 + 10x \quad (4)$$

Sujeto a:

$$\begin{aligned} x &\geq 10 \\ U(x) &\leq 80 \end{aligned}$$

Restricciones que pide el problema 4 :

- El uso de CPU no puede superar el 80%:

$$2x^2 + 10x \leq 80$$

- El servidor debe procesar al menos 10 peticiones por segundo:

$$x \geq 10$$

Solución

Resolviendo la ecuación cuadrática:

$$2x^2 + 10x - 80 \leq 0$$

Simplificamos:

$$x^2 + 5x - 40 \leq 0$$

Resolviendo la ecuación cuadrática:

$$x = \frac{-5 \pm \sqrt{5^2 - 4(1)(-40)}}{2(1)}$$

$$x = \frac{-5 \pm \sqrt{185}}{2}$$

$$x_1 = 4.3 \quad (\text{no factible porque } x \geq 10)$$

$$x_2 = -9.3 \quad (\text{no factible porque } x \geq 10)$$

Como las soluciones no cumplen la restricción de $x \geq 10$, evaluamos directamente en $x = 10$:

$$U(10) = 2(10)^2 + 10(10) = 300$$

El uso de CPU excede el límite del 80%, por lo que no existe una solución factible que cumpla con ambas restricciones.

INTERPRETACION DEL PROBLEMA:

Este ejercicio no tiene una solución factible que satisfaga ambas restricciones de procesamiento mínimo y límite de uso de CPU.

CAPTURA DEL CODIGO

```
Fle Edit Selection View Go Run Terminal Help ↶ ↷ tabaco 2 fred

• ejercicios.py • ejerciciosipy • ejercicio4py • ejercicio10py •

• ejercicio10py >

56 USO_CPU_VALUES = uso.cpu_val()

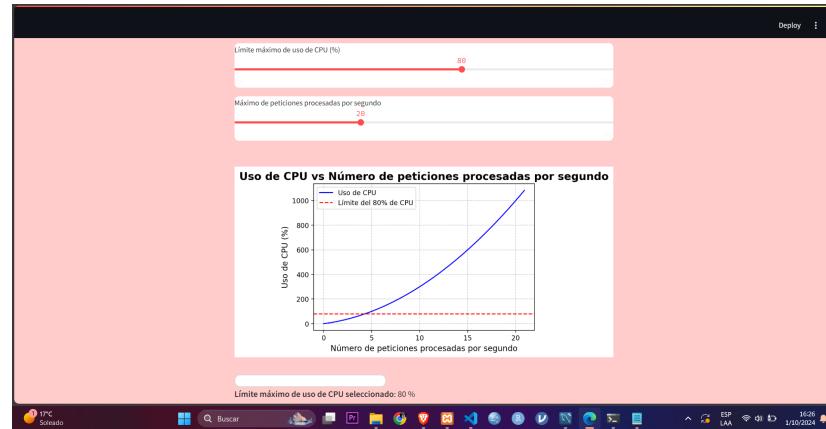
57
58 # Layout de la presentación en dos columnas
59 st.markdown(<div class="container">, unsafe_allow_html=True)
60
61 # Lado izquierdo: Gráfico
62 st.markdown(<div class="left-side">, unsafe_allow_html=True)

63 fig, ax = plt.subplots(figsize=(6, 3))
64 ax.set_xlabel('val', loc='cpu_value', label='uso de CPU', color='b')
65 ax.set_xlim(0, USO_CPU_VALUES, color='r', linestyle='--', label='límite del (uso_cpu) % de CPU')
66 ax.set_title('uso de CPU vs numero de peticiones procesadas por segundo', fontsize=16, fontweight='bold')
67 ax.set_ylabel('uso de CPU (%)', color='g', fontweight='bold')
68 ax.grid(True, linestyle='--', alpha=0.7)
69 ax.legend(fontsize=10)
70 st.pyplot(fig)
71
72 st.markdown(</div>, unsafe_allow_html=True)

73
74 # Lado derecho: Mostrar resultados
75 st.markdown(<div class="right-side">, unsafe_allow_html=True)

76
77 # Mostrar el límite de uso de CPU
78 st.markdown(f'**límite máximo de uso de CPU seleccionado**: {USO_CPU_VALUES} %')
79
80 # Mostrar el máximo de peticiones procesadas por segundo
81 st.markdown(f'**máximo de peticiones procesadas por segundo**: {max_peticiones}')
82
83 # Mostrar los valores del uso de CPU para cada punto
84 for x in np.arange(0, max_peticiones + 1):
85     | st.markdown(f'Número de peticiones procesadas**: [{x}], **Uso de CPU**: {uso_cpu(x)} %')
86
87 st.markdown(</div>, unsafe_allow_html=True)
88 st.markdown(</div>, unsafe_allow_html=True)
89
90
91
```

CAPTURA DEL RESULTADO



PROBLEMA 5 Y ENUNCIADO :

Durante el entrenamiento de un modelo de machine learning, el batch size x afecta el tiempo de entrenamiento $T(x) = \frac{1000}{x} + 0.1x$. El tamaño del lote debe estar entre 16 y 128. Encuentra el batch size que minimiza el tiempo de entrenamiento.

Función objetivo:

$$T(x) = 1000 \underset{x+0.1x \text{ Sujeto a: } 16 \leq x \leq 128}{\text{(Rango del batch size)}}$$

$$\text{Minimizar: } T(x) = \frac{1000}{x} + 0.1x \quad (5)$$

Sujeto a:

$$16 \leq x \leq 128$$

Restricciones

El tamaño del lote debe estar entre 16 y 128:

$$16 \leq x \leq 128$$

Función

$$\text{Minimizar } T(x) = \frac{1000}{x} + 0.1x$$

Solución

Para encontrar el valor de x que minimiza $T(x)$, tomamos la derivada de $T(x)$:

$$T'(x) = -\frac{1000}{x^2} + 0.1$$

Igualamos a cero:

$$\begin{aligned} -\frac{1000}{x^2} + 0.1 &= 0 \\ \frac{1000}{x^2} &= 0.1 \\ x^2 &= 10000 \\ x &= \sqrt{10000} = 100 \end{aligned}$$

Evaluación en los extremos

Evaluamos en los extremos del intervalo:

- Para $x = 16$:

$$T(16) = \frac{1000}{16} + 0.1(16) = 64.1$$

- Para $x = 128$:

$$T(128) = \frac{1000}{128} + 0.1(128) = 20.61$$

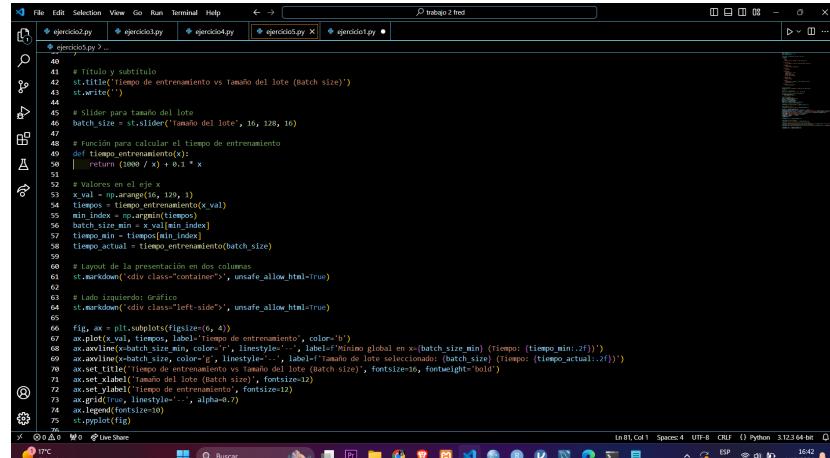
- Para $x = 100$:

$$T(100) = \frac{1000}{100} + 0.1(100) = 20$$

INTERPRETACION DEL PROBLEMA:

El tamaño del lote que minimiza el tiempo de entrenamiento es $x = 100$, con un tiempo mínimo de entrenamiento de $T(100) = 20$ segundos.

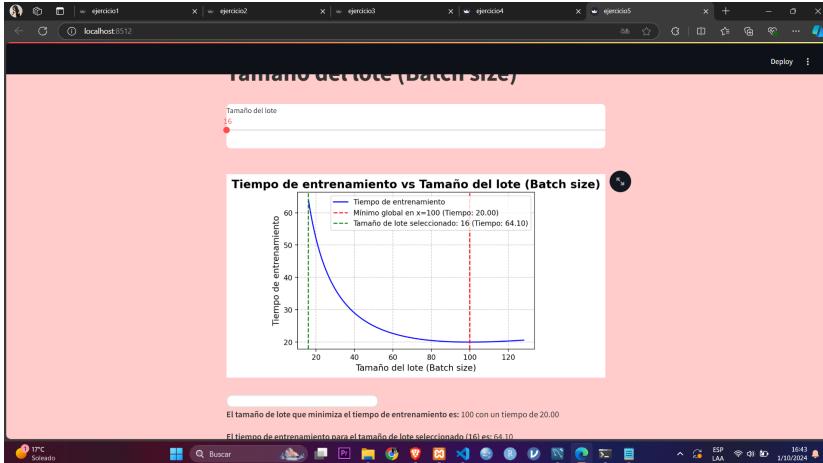
CAPTURA DEL CODIGO



The screenshot shows a terminal window titled "Intabgo 2 fred" with the following Python code:

```
File Edit Selection View Go Run Terminal Help ↵ [ejercicio2.py] [ejercicio1.py] [ejercicio4.py] [ejercicio3.py] [ejercicio1.py] •
40
41 # Título y subtítulo
42 st.title("Tiempo de entrenamiento vs tamaño del lote (batch size)")
43 st.write(" ")
44
45 # Slider para tamaño del lote
46 batch_size = st.slider("Tamaño del lote", 16, 128, 16)
47
48 # Función para calcular el tiempo de entrenamiento
49 def tiempo_entrenamiento(x):
50     return (1000 / x) + 0.1 * x
51
52 # Valores en el eje x
53 x_val = np.arange(16, 128, 1)
54 tiempos = tiempo_entrenamiento(x_val)
55 min_index = np.argmin(tiempos)
56 batch_size_min = x_val[min_index]
57 tiempo_min = tiempos[min_index]
58 tiempo_actual = tiempo_entrenamiento(batch_size)
59
60 # Layout de la presentación en dos columnas
61 st.markdown(<div class="container">, unsafe_allow_html=True)
62
63 # Lado izquierdo: Gráfico
64 st.markdown(<div class="left-side">, unsafe_allow_html=True)
65
66 fig, ax = plt.subplots(figsize=(6, 4))
67 ax.plot(x_val, tiempos, label="Tiempo de entrenamiento", color='r')
68 ax.set_xlabel("Tamaño del lote (batch size)", color='r', linestyle='--', label="Minimo global en x:[batch_size_min] (Tiempo: [tiempo_min:r])")
69 ax.annotate(str(batch_size_min), xy=batch_size_min, xytext=(batch_size_min, tiempo_min), arrowprops=dict(facecolor='black', shrink=5))
70 ax.set_title("Tiempo de entrenamiento vs tamaño del lote (batch size)", fontsize=10, fontweight="bold")
71 ax.set_ylabel("Tiempo de entrenamiento", fontsize=10)
72 ax.set_xlabel("Tamaño del lote (batch size)", fontsize=10)
73 ax.grid(True, linestyle='--', alpha=0.5)
74 ax.legend(fontsize=10)
75 st.pyplot(fig)
```

CAPTURA DEL RESULTADO



PROBLEMA 6 Y ENUNCIADO .

Un sistema de transmisión de datos tiene un ancho de banda total de 1000 Mbps. Cada archivo que se transmite utiliza x Mbps. El sistema puede transmitir un máximo de 50 archivos a la vez, y cada archivo adicional más allá de 30 reduce el ancho de banda disponible en un 5 %. Maximiza el número de archivos transmitidos.

FORMULAS

$$\text{Maximizar: } N(x) = \begin{cases} x & \text{si } x \leq 30 \\ 30 + 0.95(x - 30) & \text{si } x > 30 \end{cases}$$

$$\text{Sujeto a: } x \leq 50$$

Restricciones que pide el problema:

Existen dos casos para las restricciones del problema:

- Si $n \leq 30$: No hay penalización en el ancho de banda, por lo que la restricción es:

$$n \times x \leq 1000$$

- Si $n > 30$: Se aplica una penalización del 5% por cada archivo adicional más allá de 30. El ancho de banda disponible es:

$$A = 1000 \times (1 - 0.05 \times (n - 30))$$

En este caso, la restricción se vuelve:

$$n \times x \leq A$$

Función

Queremos maximizar el número de archivos n sujetos a las restricciones del ancho de banda.

sujeta a:

$$\begin{aligned} n \times x &\leq 1000, \quad \text{si } n \leq 30 \\ n \times x &\leq 1000 \times (1 - 0.05 \times (n - 30)), \quad \text{si } n > 30 \end{aligned}$$

Solucion

Caso 1: $n \leq 30$

Cuando $n \leq 30$, la restricción es:

$$n \times x \leq 1000$$

Esto implica que:

$$n \leq \frac{1000}{x}$$

Entonces: Si $x = 20$ Mbps (ancho de banda por archivo), tenemos:

$$n \leq \frac{1000}{20} = 50$$

Pero dado que solo podemos transmitir hasta 30 archivos sin penalización, podemos transmitir hasta 30 archivos en este caso.

INTERPRETACION DEL PROBLEMA :

Si $x = 20$ Mbps, el número máximo de archivos que se pueden transmitir es 35 antes de que el ancho de banda disponible se reduzca demasiado.

CAPTURA DEL CODIGO

```
# Ejercicio 1
# Función para calcular ancho de banda
def calcular_ancho_banda(archivos, ancho_banda_total=1000):
    if archivos_transmitidos > 100:
        if archivos_extra == 0 / 100 * archivos_extra
            ancho_banda_disponible = ancho_banda_total * (1 - reducción)
        else:
            ancho_banda_disponible = ancho_banda_total
        return max(0, ancho_banda_disponible)

# Función para maximizar archivos
def maximizar_archivos(x, ancho_banda_total=1000):
    for i in range(1, 101):
        if ancho_banda_disponible > calcular_ancho_banda(archivos, ancho_banda_total):
            if archivos * x < ancho_banda_disponible:
                return archivos, ancho_banda_disponible
    return 0, 0

# Título de la aplicación
st.title("Maximización de Transmisión de Archivos con Gráfico")

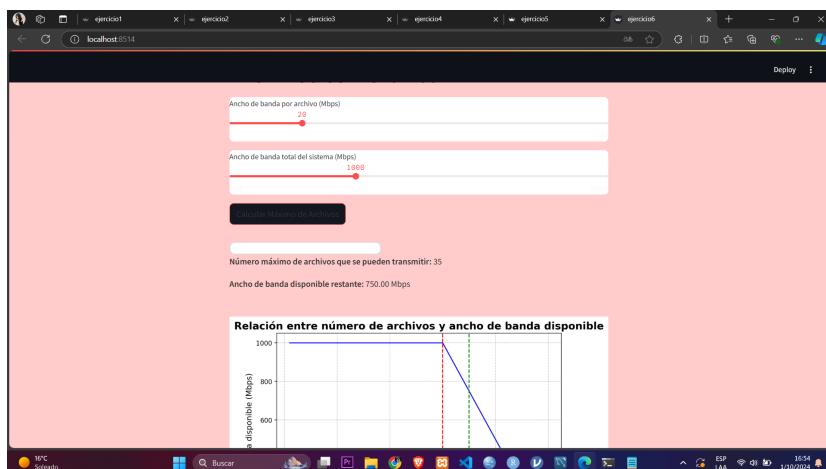
# Sliders para ancho de banda por archivo y total
x = st.slider("Ancho de banda por archivo (Mbps)", min_value=1, max_value=100, value=20)
ancho_banda_total = st.slider("Ancho de banda total del sistema (Mbps)", min_value=500, max_value=2000, value=1000)

# Botón para realizar el cálculo
if st.button("Calcular Número de Archivos"):
    archivos_max, ancho_banda_restante = maximizar_archivos(x, ancho_banda_total)

# Layout de la presentación en dos columnas
st.markdown(<div class="row"><div class="col-6">Nº de Archivos que se pueden transmitir:</div><div class="col-6">Ancho de banda disponible restante: 750.00 Mbps</div></div>, unsafe_allow_html=True)

# Lado izquierdo: Resultados
# Lado derecho: Gráfico
```

CAPTURA DEL RESULTADO



PROBLEMA 7 Y ENUNCIADO :

Un sistema de colas procesa x trabajos por segundo. La función del tiempo de respuesta $T(x) = \frac{100}{x} + 2x$. Minimiza el tiempo de respuesta del sistema, considerando que el sistema debe procesar al menos 5 trabajos por segundo.
FORMULAS .

$$\text{Minimizar: } T(x) = \frac{100}{x} + 2x$$

Sujeto a: $x \geq 5$

Restricciones

El sistema debe procesar al menos 5 trabajos por segundo:

$$x \geq 5$$

Función

Minimizar el tiempo de respuesta $T(x)$:

$$T(x) = \frac{100}{x} + 2x$$

Solución:

Para encontrar el valor de x que minimiza $T(x)$, tomamos la derivada de $T(x)$ respecto a x :

$$T'(x) = -\frac{100}{x^2} + 2$$

Igualamos la derivada a cero para encontrar los puntos críticos:

$$-\frac{100}{x^2} + 2 = 0$$

$$\frac{100}{x^2} = 2$$

$$x^2 = \frac{100}{2} = 50$$

$$x = \sqrt{50} = 5\sqrt{2} = 7.07$$

El valor que minimiza el tiempo de respuesta es $x = 7.07$. Verificamos que este valor cumple con la restricción $x \geq 5$.

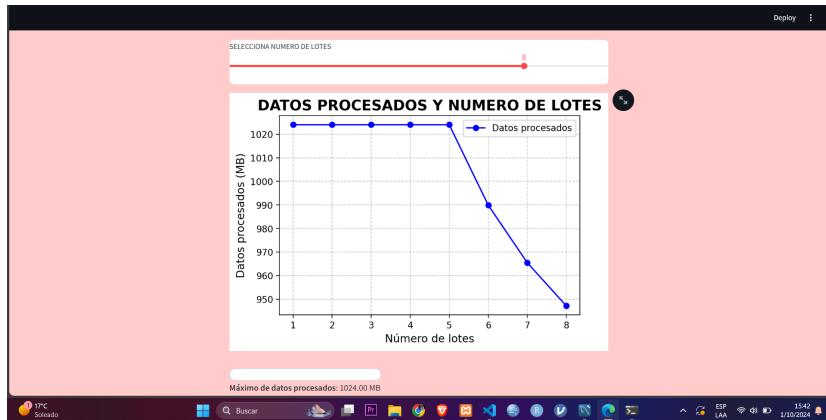
INTERPRETACION DEL PROBLEMA:

El valor óptimo de x que minimiza el tiempo de respuesta del sistema es $x = 7.07$ trabajos por segundo.

CAPTURA DEL CODIGO

```
ejercicio.py
41 # Parametros
42 memoria = 1024
43 eficiencia_re = 0.8
44
45 # Funcion para calcular los datos procesados
46 def datos_procesados(n, x):
47     if x <= 5:
48         return n * x
49     else:
50         return 5 * x + (n - 5) * eficiencia_re * x
51
52 # Mostrar el grafico primero
53 st.markdown(<div class="container">, unsafe_allow_html=True)
54
55 # Lado izquierdo: Grafico
56 st.markdown(<div class="leftside">, unsafe_allow_html=True)
57 max_lotes = int(input("Introduce numero de lotes : "))
58 min_lotes = int(input("Introduce numero de lotes : "))
59 datos = np.array([datos_procesados(n, memoria / n) for n in lotes])
60
61 fig, ax = plt.subplots(figsize=(6, 4))
62 ax.plot(lotes, datos, marker='o', linestyle='-', color='b', label='Datos procesados')
63 ax.set_xlabel('Número de lotes', fontsize=10, fontweight='bold')
64 ax.set_ylabel('Datos procesados (MB)', fontsize=10, fontweight='bold')
65 ax.set_title('DATOS PROCESADOS Y NÚMERO DE LOTES', fontsize=12)
66 ax.grid(True, linestyle='--', alpha=0.5)
67 ax.legend(fontsize=10)
68 st.pyplot(fig)
69 st.markdown(</div>, unsafe_allow_html=True)
70
71 # Lado derecho: Resultados en un cuadro ordenado
72 st.markdown(<div class="right-side">, unsafe_allow_html=True)
73
74 # Mostrar los resultados: maximo, minimo y valores en cada punto
75 max_dato = np.max(datos)
76 min_dato = np.min(datos)
77
```

CAPTURA DEL RESULTADO



PROBLEMA 8 Y ENUNCIADO:

El entrenamiento de un modelo de deep learning en una GPU consume x unidades de energía por lote. El objetivo es maximizar el tamaño del lote x , pero el consumo de energía total no puede exceder las 200 unidades por segundo, y cada lote adicional más allá del 10 reduce el rendimiento en un 10 %.

FORMULA :

Maximizar: x

Sujeto a: $x \leq 10$

Maximizar: $10 + 0.9(x - 10)$ para $x > 10$

Sujeto a: $x \cdot \text{unidades de energía} \leq 200$

Restricciones

1. Consumo de energía total:

$$x \cdot E(x) \leq 200$$

2. Reducción de rendimiento: Si $x > 10$,

Entonces..

El consumo de energía por lote $E(x)$ se modela como sigue:

$$E(x) = \begin{cases} x & \text{si } x \leq 10 \\ x \cdot (1 + 0.1 \cdot (x - 10)) & \text{si } x > 10 \end{cases}$$

Función

$$x \cdot E(x) \leq 200$$

Solución

Caso 1: $x \leq 10$

$$x^2 \leq 200 \Rightarrow x \leq \sqrt{200} \approx 14.14$$

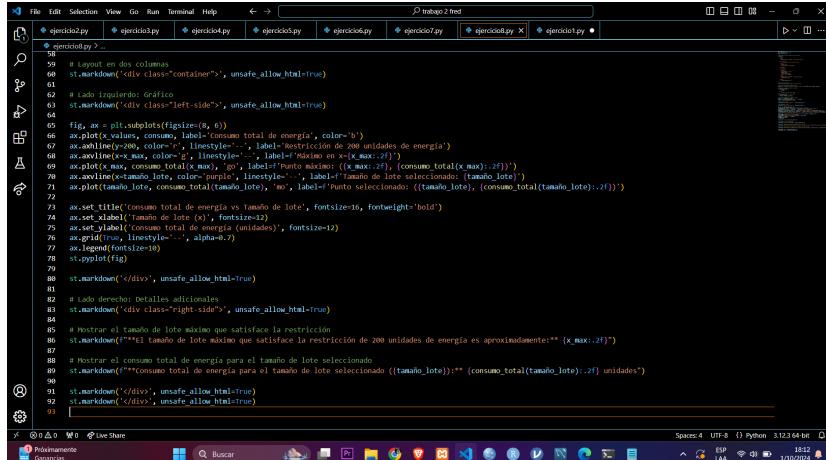
Caso 2: $x > 10$

$$x \cdot x \cdot (1 + 0.1(x - 10)) \leq 200$$

INTERPRETACION DEL PROBLEMA :

El tamaño óptimo de lote x depende de si $n \leq 10$, o $n > 10$. En el primer caso, se maximiza el tamaño del lote manteniendo el rendimiento completo; el rendimiento disminuye y la energía debe ajustarse para no exceder el límite.

CAPTURA DEL CODIGO

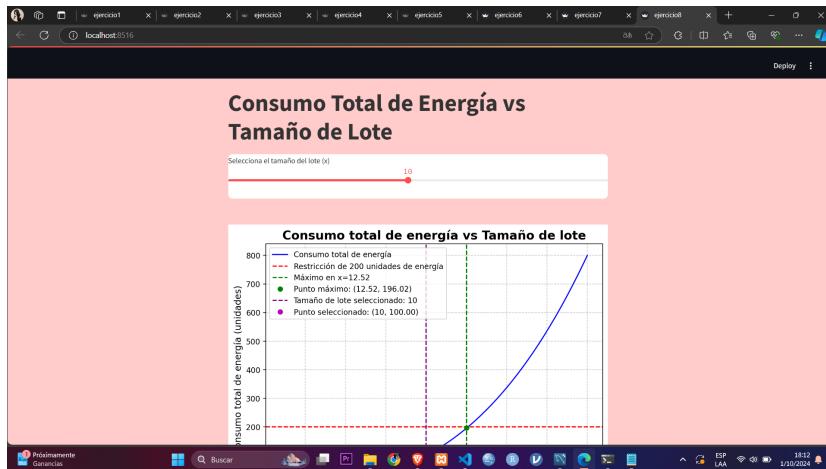


```

File Edit Selection View Go Run Terminal Help ← → ⌘ trabajo 2 fred
ejercicio2.py ejerciciol.py ejerciciod.py ejercicod.py ejercicod.py ejercicod.py ejercicod.py ejercicod.py ejercicod.py
ejercicod.py > ...
59
60 # Lado izquierdo: Gráfico
61 st.markdown('<div class="row">', unsafe_allow_html=True)
62 st.markdown('  <div class="col-12">', unsafe_allow_html=True)
63
64 fig, ax = plt.subplots(figsize=(6, 6))
65 ax.set_title('Consumo total de energía', color='blue')
66 ax.set_xlabel('Tamaño de lote', color='blue', label='Restricción de 200 unidades de energía')
67 ax.set_ylabel('Consumo total de energía', color='blue', label='Máximo en x=12.52')
68 ax.axhline(y=200, color='red', linestyle='--', label='Restricción de 200 unidades de energía')
69 ax.axvline(x=x_max, color='green', linestyle='--', label='Máximo en x={x_max:.2f}')
70 ax.plot(x_max, consumo_total(x_max), 'ro', label='Punto máximo: ({x_max:.2f}, {consumo_total(x_max):.2f})')
71 ax.plot(tamano_lote, consumo_total(tamano_lote), 'bo', label='Consumo total(tamano_lote)')
72 ax.plot(tamano_lote, consumo_total(tamano_lote), 'wo', label='Punto seleccionado: ({tamano_lote}, {consumo_total(tamano_lote):.2f})')
73
74 ax.set_title('Consumo total de energía vs. Tamaño de lote', fontsize=16, fontweight='bold')
75 ax.set_xlabel('Tamaño de lote (x)', fontsize=12)
76 ax.set_ylabel('Consumo total de energía (unidades)', fontsize=12)
77 ax.grid(True, linestyle='--', alpha=0.7)
78 ax.legend(fontsize=10)
79 st.pyplot(fig)
80 st.markdown('</div>', unsafe_allow_html=True)
81
82 # Lado derecho: Detalles adicionales
83 st.markdown('<div class="col-12">', unsafe_allow_html=True)
84
85 # Mostrar el tamaño de lote máximo que satisface la restricción
86 st.markdown('**# El tamaño de lote máximo que satisface la restricción de 200 unidades de energía es aproximadamente:** {x_max:.2f}')
87
88 # Mostrar el consumo total de energía para el tamaño de lote seleccionado
89 st.markdown('**Consumo total de energía para el tamaño de lote seleccionado:** {consumo_total(tamano_lote):.2f} unidades')
90
91 st.markdown('</div>', unsafe_allow_html=True)
92 st.markdown('</div>', unsafe_allow_html=True)
93

```

CAPTURA DEL RESULTADO



PROBLEMA 9 Y ENUNCIADO :

Una empresa almacena datos en la nube. El costo de almacenamiento por TB es de $50 + 5x$ dólares, donde x es la cantidad de TB de almacenamiento utilizado. La empresa tiene un presupuesto de 500 dólares. Maximiza la cantidad de datos almacenados sin exceder el presupuesto.

FORMULAS :

Maximizar: x

Sujeto a: $50 + 5x \leq 500$

Restricciones

1. Presupuesto total:

$$50 + 5x \leq 500$$

Función

Maximizar la cantidad de almacenamiento x sujeto a la restricción del presupuesto.

Solución del ejercicio :

El costo de almacenamiento por x TB se expresa como:

$$C(x) = 50 + 5x$$

La restricción se puede reescribir como:

$$50 + 5x \leq 500$$

Resolviendo esta ecuación:

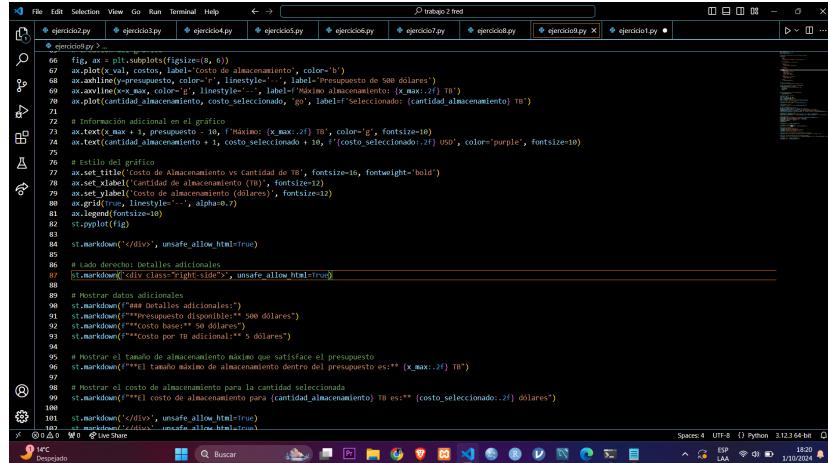
$$5x \leq 500 - 50$$

$$5x \leq 450 \Rightarrow x \leq 90$$

INTERPRETACION DEL PROBLEMA :

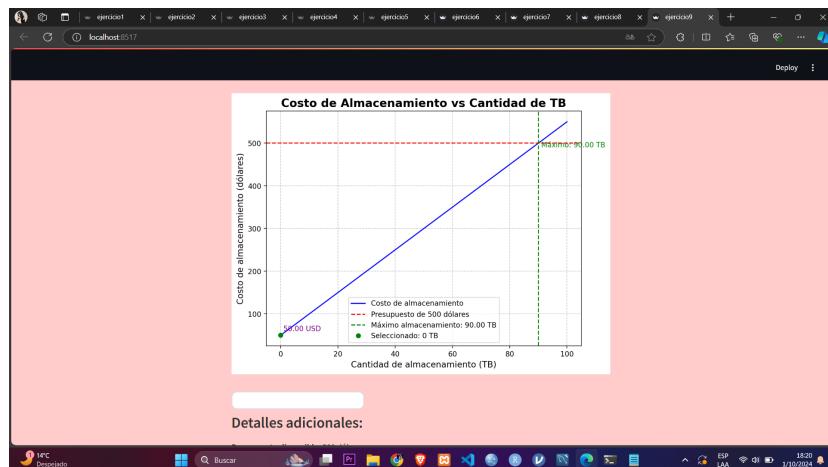
La empresa puede almacenar un máximo de 90 TB sin exceder su presupuesto de 500 dólares.

CAPTURA DEL CODIGO



```
File Edit Selection View Go Run Terminal Help ↻ → ⌂ trabajo 2 fred
ejercicio1.py ejercicio1.ipynb ejercicio4.py ejercicio5.py ejercicio6.py ejercicio7.py ejercicio8.py ejercicio9.py ejercicio10.py
ejercicio1.py > ...
66 fig, ax = plt.subplots(figsize=(8, 6))
67 ax.plot(x_val, costos, label='Costo de almacenamiento', color='blue')
68 ax.set_xlabel('Cantidad de almacenamiento (TB)', label='Presupuesto de 500 dólares')
69 ax.set_ylabel('Costo de almacenamiento (dólares)', label='Presupuesto de 500 dólares')
70 ax.axvline(x=x_max, color='red', linestyle='--', label='Presupuesto de 500 dólares: (x_max,:2) TB')
71 ax.plot(cantidad_almacenamiento, costo_seleccionado, 'go', label='Seleccionado: (cantidad_almacenamiento) TB')
72 # Información adicional en el gráfico
73 ax.text(x_max + 1, presupuesto - 10, f'Máximo: (x_max,:2) TB', color='green', fontsize=10)
74 ax.text(cantidad_almacenamiento + 1, costo_seleccionado + 10, f'(costo_seleccionado,:2) USD', color='purple', fontsize=10)
75 # Estilo del gráfico
76 ax.set_title('Costo de Almacenamiento vs Cantidad de TB', fontsize=16, fontweight='bold')
77 ax.set_xlabel('Cantidad de almacenamiento (TB)', fontsize=12)
78 ax.set_ylabel('Costo de almacenamiento (dólares)', fontsize=12)
79 ax.grid(True, linestyle='--', alpha=0.2)
80 ax.legend(fontsize=10)
81 st.pyplot(fig)
82
83 # Mostrar datos adicionales
84 st.markdown('</div>', unsafe_allow_html=True)
85 # Lado derecho: Detalles adicionales
86 st.markdown('<div class="right-side">', unsafe_allow_html=True)
87
88 # Mostrar datos adicionales
89 st.markdown("## Detalles adicionales:")
90 st.markdown("Presupuesto disponible: 500 dólares")
91 st.markdown("Costo base: 50 dólares")
92 st.markdown("Costo por TB adicional: 5 dólares")
93
94 # Mostrar el punto de almacenamiento máximo que satisface el presupuesto
95 st.markdown(f"El tamaño máximo de almacenamiento dentro del presupuesto es: {x_max:.2f} TB")
96
97 # Mostrar el costo de almacenamiento para la cantidad seleccionada
98 st.markdown(f"El costo de almacenamiento para (cantidad_almacenamiento) TB es: {(costo_seleccionado,:2)} dólares")
99
100 st.markdown('</div>', unsafe_allow_html=True)
101 st.markdown('</div>', unsafe_allow_html=True)
```

CAPTURA DEL RESULTADO



PROBLEMA 10 Y ENUNCIADO:

Un sistema de mensajería tiene una latencia $L(x) = 100 - 2x$, donde x es el número de mensajes por segundo. La latencia no puede ser inferior a 20 ms debido a restricciones del protocolo. Maximiza el número de mensajes enviados sin que la latencia caiga por debajo de este límite **FORMULA :**

$$\text{Maximizar: } x$$

$$\text{Sujeto a: } 100 - 2x \geq 20$$

Restricciones

1. Latencia mínima:

$$L(x) \geq 20$$

Función

Maximizar la cantidad de mensajes x sujeto a la restricción de latencia.

Solucion

La latencia se expresa como:

$$L(x) = 100 - 2x$$

La restricción de latencia mínima se puede reescribir como:

$$100 - 2x \geq 20$$

Resolviendo esta desigualdad:

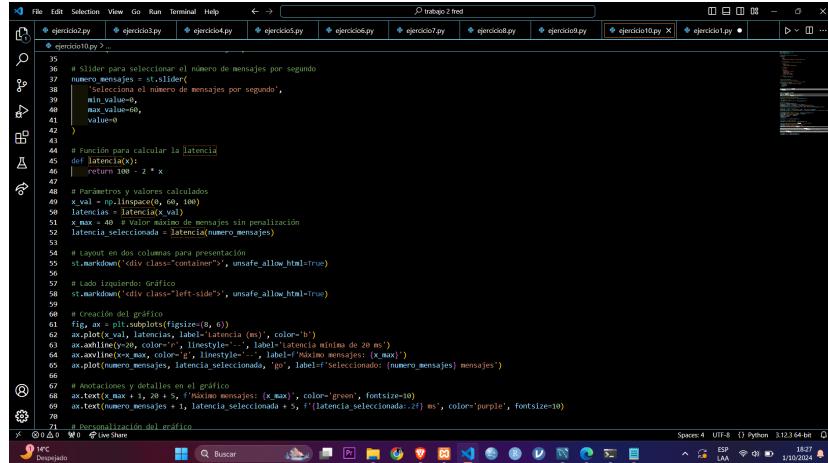
$$-2x \geq 20 - 100$$

$$-2x \geq -80 \Rightarrow x \leq 40$$

INTERPRETACION DEL PROBLEMA :

El número máximo de mensajes que se pueden enviar por segundo, sin que la latencia caiga por debajo de 20 ms, es 40.

CAPTURA DEL CODIGO



```
File Edit Selection View Go Run Terminal Help ← → trabajo 2 fred
ejercicio1.py ejercicio10.py ejercicio2.py ejercicio3.py ejercicio4.py ejercicio5.py ejercicio6.py ejercicio7.py ejercicio8.py ejercicio9.py ejercicio10.py ejercicio11.py
35 # Slider para seleccionar el número de mensajes por segundo
36 numero_mensajes = st.slider(
37     'Selecciona el número de mensajes por segundo',
38     min_value=0,
39     max_value=40,
40     value=0
41 )
42
43 # Función para calcular la latencia
44 def latencia(x):
45     return 100 - x * x
46
47 # Parámetros y valores calculados
48 x_val = np.linspace(0, 40, 100)
49 latencia = latencia(x_val)
50
51 x_max = 40 # Valor máximo de mensajes sin penalización
52 latencia_seleccionada = latencia(numero_mensajes)
53
54 # Layout en dos columnas para presentación
55 st.markdown(<div class="container">, unsafe_allow_html=True)
56
57 # Lado Izquierdo: Gráfico
58 st.markdown(<div class="left-side">, unsafe_allow_html=True)
59
60 # Creación del gráfico
61 fig, ax = plt.subplots(figsize=(8, 6))
62 ax.plot(x_val, latencia, label='Latencia (ms)')
63 ax.axvline(x_max, color='g', linestyle='--', label='Latencia mínima de 20 ms')
64 ax.axvline(x_max, color='g', linestyle='-', label='Máximo mensajes: ('+str(x_max)+')')
65 ax.plot(numero_mensajes, latencia_seleccionada, 'go', label='Selezionado: ('+str(numero_mensajes)+') mensajes')
66
67 # Anotaciones y detalles en el gráfico
68 ax.text(x_max + 1, 20 + 5, f'Máximo mensajes: ({x_max})', color='green', fontsize=10)
69 ax.text(numero_mensajes + 1, latencia_seleccionada + 5, f'({latencia_seleccionada:.2f} ms)', color='purple', fontsize=10)
70
71 # Personalización del gráfico
```

CAPTURA DEL RESULTADO

