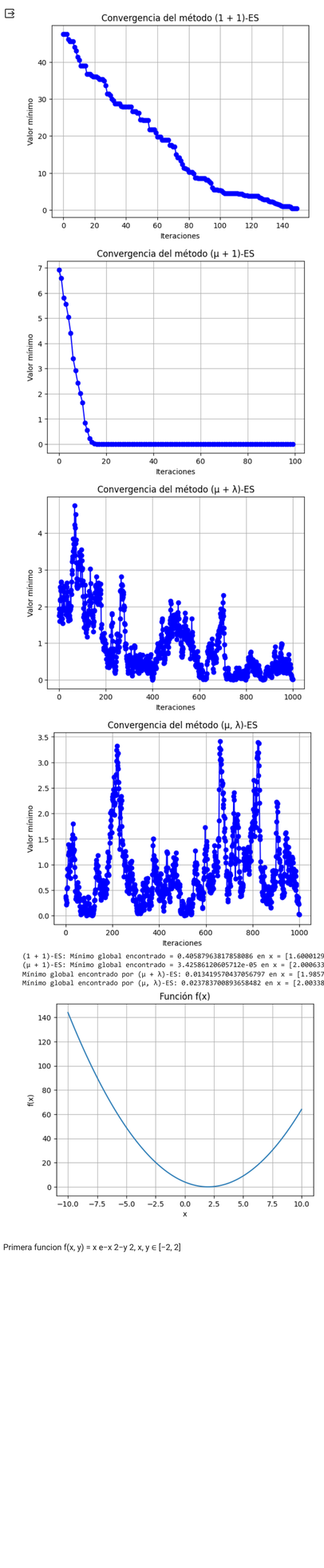
Segunda funcion Estrategias Evolutivas (1 + 1)-ES, (μ + 1)-ES, (μ + λ)-ES y (μ, λ)-ES:

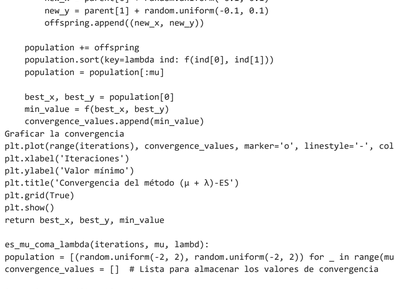
```
import random
import numpy as np
import matplotlib.pyplot as plt

# Función dada
def f(x):
    return np.sum((x - 2)**2)

# Estrategia Evolutiva (1 + 1)-ES
def es_1_1(iteraciones):
    x = np.random.uniform(-10, 10, 2)
    convergence_values = []  # Lista para almacenar los valores de convergencia

    for _ in range(iteraciones):
        x_hijo = x + np.random.normal(0, 0.1, 2)  # Mutación
        if f(x_hijo) < f(x):
            x = x_hijo
        # Registrar el valor mínimo en cada iteración
        convergence_values.append(f(x))

    # Graficar la convergencia
    plt.plot(range(iteraciones), convergence_values, marker='o', linestyle='-', color='b')
    plt.xlabel('Iteraciones')
    plt.ylabel('Valor mínimo')
    plt.title('Convergencia del método (1 + 1)-ES')
    plt.grid(True)
    plt.show()
    return x, f(x)

# Estrategia Evolutiva (μ + 1)-ES
def es_mu_1(iteraciones, mu):
    x = np.random.uniform(-10, 10, 2)
    convergence_values = []  # Lista para almacenar los valores de convergencia

    for _ in range(iteraciones):
        descendientes = [x + np.random.normal(0, 0.1, 2) for _ in range(mu)]
        x_hijo = min(descendientes, key=f)  # Seleccionamos el mejor descendiente
        if f(x_hijo) < f(x):
            x = x_hijo
        # Registrar el valor mínimo en cada iteración
        convergence_values.append(f(x))

    # Graficar la convergencia
    plt.plot(range(iteraciones), convergence_values, marker='o', linestyle='-', color='b')
    plt.xlabel('Iteraciones')
    plt.ylabel('Valor mínimo')
    plt.title('Convergencia del método (μ + 1)-ES')
    plt.grid(True)
    plt.show()
    return x, f(x)

# Estrategia Evolutiva (μ + λ)-ES
def es_mu_lambda(iteraciones, mu, lambd):
    poblacion = np.random.uniform(-10, 10, size=(mu, 2))
    convergence_values = []  # Lista para almacenar los valores de convergencia

    for _ in range(iteraciones):
        descendientes = np.array([p + np.random.normal(0, 0.1, 2) for p in poblacion])
        valores_descendientes = np.array([f(x) for x in descendientes])
        indices_mejores = np.argsort(valores_descendientes)[:mu]
        poblacion = descendientes[indices_mejores]

        mejor_x = poblacion[0]
        mejor_valor = f(mejor_x)
        convergence_values.append(mejor_valor)

     # Graficar la convergencia
    plt.plot(range(iteraciones), convergence_values, marker='o', linestyle='-', color='b')
    plt.xlabel('Iteraciones')
    plt.ylabel('Valor mínimo')
    plt.title('Convergencia del método (μ + λ)-ES')
    plt.grid(True)
    plt.show()
    return mejor_x, mejor_valor

# Estrategia Evolutiva (μ, λ)-ES
def es_mu_coma_lambda(iteraciones, mu, lambd):
    poblacion = np.random.uniform(-10, 10, size=(mu, 2))
    convergence_values = []  # Lista para almacenar los valores de convergencia
    for _ in range(iteraciones):
        descendientes = np.array([p + np.random.normal(0, 0.1, 2) for p in poblacion])
        valores_descendientes = np.array([f(x) for x in descendientes])
        indices_mejores = np.argsort(valores_descendientes)[:mu]
        poblacion = descendientes[indices_mejores]

        mejor_x = poblacion[0]
        mejor_valor = f(mejor_x)
        convergence_values.append(mejor_valor)

     # Graficar la convergencia
    plt.plot(range(iteraciones), convergence_values, marker='o', linestyle='-', color='b')
    plt.xlabel('Iteraciones')
    plt.ylabel('Valor mínimo')
    plt.title('Convergencia del método (μ, λ)-ES')
    plt.grid(True)
    plt.show()
    return mejor_x, mejor_valor

# Ejemplo de uso
min_x_1_1, min_valor_1_1 = es_1_1(150)
min_x_mu_1, min_valor_mu_1 = es_mu_1(100, mu=10)
min_x_mu_lambda, min_valor_mu_lambda = es_mu_lambda(iteraciones=1000, mu=100, lambd=20)
min_x_mu_coma_lambda, min_valor_mu_coma_lambda = es_mu_coma_lambda(iteraciones=1000, mu=100, lambd=30)

print(f"(1 + 1)-ES: Mínimo global encontrado = {min_valor_1_1} en x = {min_x_1_1}")
print(f"(μ + 1)-ES: Mínimo global encontrado = {min_valor_mu_1} en x = {min_x_mu_1}")
print(f"Mínimo global encontrado por (μ + λ)-ES: {min_valor_mu_lambda} en x = {min_x_mu_lambda}")
print(f"Mínimo global encontrado por (μ, λ)-ES: {min_valor_mu_coma_lambda} en x = {min_x_mu_coma_lambda}")

# Función dada
def f(x):
    return np.sum((x - 2)**2)

# Generar valores de x
x = np.linspace(-10, 10, 100)

# Evaluar la función para cada valor de x
y = np.array([f(xi) for xi in x])

# Graficar
plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Función f(x)')
plt.grid(True)
plt.show()
```

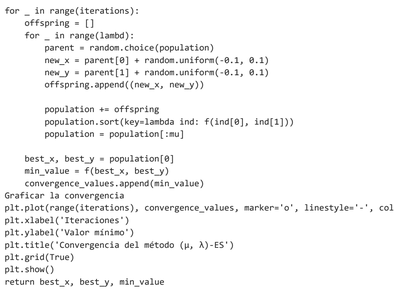


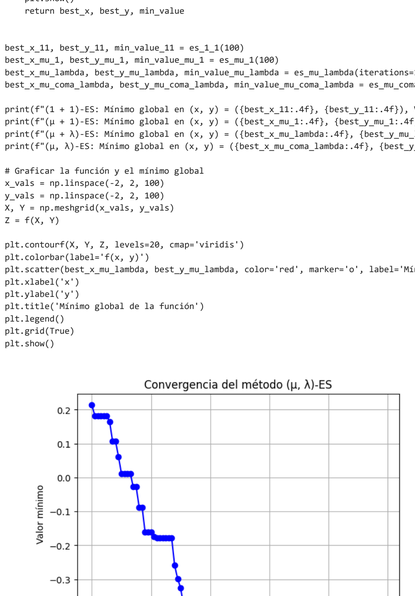

```
(1 + 1)-ES: Mínimo global encontrado = 0.40587963817858086 en x = [1.6000129  1.50412707]
(μ + 1)-ES: Mínimo global encontrado = 3.42586120605712e-05 en x = [2.00063373 1.99418132]
Mínimo global encontrado por (μ + λ)-ES: 0.013419570437056797 en x = [1.98575963 2.11496427]
Mínimo global encontrado por (μ, λ)-ES: 0.023783700893658482 en x = [2.00338855 1.84581758]
```


Primera funcion f(x, y) = x e−x 2−y 2, x, y ∈ [−2, 2]

```python
import random
import math
import numpy as np
import matplotlib.pyplot as plt

def f(x, y):
    return x * np.exp(-x**2 - y**2)

def es_1_1(iterations):
    x = random.uniform(-2, 2)
    y = random.uniform(-2, 2)
    best_value = f(x, y)

    mutation_step = 0.1
    # Lista para almacenar los valores de convergencia
    convergence_values = []

    for _ in range(iterations):
        new_x = x + random.uniform(-mutation_step, mutation_step)
        new_y = y + random.uniform(-mutation_step, mutation_step)
        new_value = f(new_x, new_y)

        if new_value < best_value:
            x, y = new_x, new_y
            best_value = new_value

        # Registrar el valor mínimo en cada iteración
        convergence_values.append(best_value)

    # Graficar la convergencia
    plt.plot(range(iterations), convergence_values, marker='o', linestyle='-', color='b')
    plt.xlabel('Iteraciones')
    plt.ylabel('Valor mínimo')
    plt.title('Convergencia del método (μ, λ)-ES')
    plt.grid(True)
    plt.show()

    return x, y, best_value

def es_mu_1(iterations):
    x = random.uniform(-2, 2)
    y = random.uniform(-2, 2)
    best_value = f(x, y)

    mutation_step = 0.1
    # Lista para almacenar los valores de convergencia
    convergence_values = []

    for _ in range(iterations):
        new_x = x + random.uniform(-mutation_step, mutation_step)
        new_y = y + random.uniform(-mutation_step, mutation_step)
        new_value = f(new_x, new_y)

        if new_value < best_value:
            x, y = new_x, new_y
            best_value = new_value

        # Registrar el valor mínimo en cada iteración
        convergence_values.append(best_value)

    # Graficar la convergencia
    plt.plot(range(iterations), convergence_values, marker='o', linestyle='-', color='b')
    plt.xlabel('Iteraciones')
    plt.ylabel('Valor mínimo')
    plt.title('Convergencia del método (μ, λ)-ES')
    plt.grid(True)
    plt.show()

    return x, y, best_value

def es_mu_lambda(iterations, mu, lambd):
    population = [(random.uniform(-2, 2), random.uniform(-2, 2)) for _ in range(mu)]
    convergence_values = []  # Lista para almacenar los valores de convergencia

    for _ in range(iterations):
        offspring = []
        for _ in range(lambd):
            parent = random.choice(population)
            new_x = parent[0] + random.uniform(-0.1, 0.1)
            new_y = parent[1] + random.uniform(-0.1, 0.1)
            offspring.append((new_x, new_y))

        population += offspring
        population.sort(key=lambda ind: f(ind[0], ind[1]))
        population = population[:mu]

        best_x, best_y = population[0]
        min_value = f(best_x, best_y)
        convergence_values.append(min_value)
    # Graficar la convergencia
    plt.plot(range(iterations), convergence_values, marker='o', linestyle='-', color='b')
    plt.xlabel('Iteraciones')
    plt.ylabel('Valor mínimo')
    plt.title('Convergencia del método (μ + λ)-ES')
    plt.grid(True)
    plt.show()
    return best_x, best_y, min_value

def es_mu_coma_lambda(iterations, mu, lambd):
    population = [(random.uniform(-2, 2), random.uniform(-2, 2)) for _ in range(mu)]
    convergence_values = []  # Lista para almacenar los valores de convergencia

    for _ in range(iterations):
        offspring = []
        for _ in range(lambd):
            parent = random.choice(population)
            new_x = parent[0] + random.uniform(-0.1, 0.1)
            new_y = parent[1] + random.uniform(-0.1, 0.1)
            offspring.append((new_x, new_y))

            population += offspring
            population.sort(key=lambda ind: f(ind[0], ind[1]))
            population = population[:mu]

        best_x, best_y = population[0]
        min_value = f(best_x, best_y)
        convergence_values.append(min_value)
    # Graficar la convergencia
    plt.plot(range(iterations), convergence_values, marker='o', linestyle='-', color='b')
    plt.xlabel('Iteraciones')
    plt.ylabel('Valor mínimo')
    plt.title('Convergencia del método (μ, λ)-ES')
    plt.grid(True)
    plt.show()
    return best_x, best_y, min_value


best_x_11, best_y_11, min_value_11 = es_1_1(100)
best_x_mu_1, best_y_mu_1, min_value_mu_1 = es_mu_1(100)
best_x_mu_lambda, best_y_mu_lambda, min_value_mu_lambda = es_mu_lambda(iterations=10, mu=10, lambd=20)
best_x_mu_coma_lambda, best_y_mu_coma_lambda, min_value_mu_coma_lambda = es_mu_coma_lambda(iterations=10, mu=10, lambd=20)

print(f"(1 + 1)-ES: Mínimo global en (x, y) = ({best_x_11:.4f}, {best_y_11:.4f}), Valor mínimo global: {min_value_11:.4f}")
print(f"(μ + 1)-ES: Mínimo global en (x, y) = ({best_x_mu_1:.4f}, {best_y_mu_1:.4f}), Valor mínimo global: {min_value_mu_1:.4f}")
print(f"(μ + λ)-ES: Mínimo global en (x, y) = ({best_x_mu_lambda:.4f}, {best_y_mu_lambda:.4f}), Valor mínimo global: {min_value_mu_lambda:.4f}")
print(f"(μ, λ)-ES: Mínimo global en (x, y) = ({best_x_mu_coma_lambda:.4f}, {best_y_mu_coma_lambda:.4f}), Valor mínimo global: {min_value_mu_coma_lambda:.4f}")

# Graficar la función y el mínimo global
x_vals = np.linspace(-2, 2, 100)
y_vals = np.linspace(-2, 2, 100)
X, Y = np.meshgrid(x_vals, y_vals)
Z = f(X, Y)

plt.contourf(X, Y, Z, levels=20, cmap='viridis')
plt.colorbar(label='f(x, y)')
plt.scatter(best_x_mu_lambda, best_y_mu_lambda, color='red', marker='o', label='Mínimo global')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Mínimo global de la función')
plt.legend()
plt.grid(True)
plt.show()
```








```
(1 + 1)-ES: Mínimo global en (x, y) = (-0.7064, 0.0261), Valor mínimo global: -0.4286
(μ + 1)-ES: Mínimo global en (x, y) = (-0.7312, -0.0255), Valor mínimo global: -0.4281
(μ + λ)-ES: Mínimo global en (x, y) = (-0.7072, -0.0188), Valor mínimo global: -0.4287
(μ, λ)-ES: Mínimo global en (x, y) = (-0.6989, 0.0136), Valor mínimo global: -0.4287
```