```python
import week6
import sgd
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import sys

def runp(alpha=0.5, beta=0.9, n=5):
    T = pd.read_csv("data/T.csv").values
    fg = week6.generate_optimisation_functions(
        T, minibatch_size=n, seed=None)
    o = sgd.StochasticGradientDescent()
    start = np.array([3, 3])
    o.start(start)
    o.step_size(alpha)
    o.beta(beta)
    o.alg("heavy_ball")
    o.function_generator(fg)
    xs = []
    fs = []
    xs.append(o._x_value)
    fs.append(week6.f(o._x_value, T))
    for i in range(200):
        o.step()
        xs.append(o._x_value)
        fs.append(week6.f(o._x_value, T))
    return {
        "x1": [x[0] for x in xs],
        "x2": [x[1] for x in xs],
        "f": fs,
    }



x_min, x_max, y_min, y_max = [-5, 5, -5, 5]
T = pd.read_csv("data/T.csv").values
# Generate data for wireframe plot
resolution = 100
x_range = np.linspace(x_min, x_max, resolution)
y_range = np.linspace(y_min, y_max, resolution)
X, Y = np.meshgrid(x_range, y_range)

# Plot wireframe
fig = plt.figure(figsize=(12, 6))
resolution = 100
Z_contour = np.zeros_like(X)
for i in range(resolution):
    for j in range(resolution):
        Z_contour[i, j] = week6.f([X[i, j], Y[i, j]], T)

# Plot contour
ax_contour = fig.add_subplot(122)
contour = ax_contour.contourf(X, Y, Z_contour, levels=20, cmap='viridis')
plt.colorbar(contour, ax=ax_contour, label='$f_T(x)$')
ax_contour.set_xlabel('$x_1$')
ax_contour.set_ylabel('$x_2$')
ax_contour.set_xlim([-5, 5])
ax_contour.set_ylim([-5, 5])
plt.suptitle('Gradient Descent with Heavy Ball step on $f_T(x)$')

ax_f = fig.add_subplot(121)

np.random.seed(57)
T = pd.read_csv("data/T.csv").values
count_good_best = 0
count_diverged = 0
count_runs = 0
for alpha, beta in [(0.1, 0.3), (0.1, 0.5), (0.01, 0.3), (0.1, 0.99)]:
        count_runs += 1
        alpha_a = alpha # * (1-beta)
        n = 25
        run = runp(n=n, alpha=alpha, beta=beta)
        if min(run['f']) < 0.2:
            count_good_best += 1
        if run['f'][len(run)-1] > 15:
            count_diverged += 1
        label = f"$\\alpha={alpha_a}$, $\\beta={beta}$"
        ax_contour.plot(run["x1"], run["x2"], label=label)
        ax_f.plot(run['f'], label=label)

print("good best", count_good_best)
print("diverged", count_diverged)
print("runs", count_runs)
ax_f.set_yscale('log')
ax_f.set_xlabel("iteration $t$")
ax_f.set_ylabel("$f_T(x_t)$")
ax_f.legend(loc="upper right")
plt.savefig("fig/ciii.pdf")
plt.show()
```