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

f = {
    "function": lib.f_real,
    "gradient": lib.f_grad,
}

g = {
    "function": lib.g_real,
    "gradient": lib.g_grad,
}


def gradient_descent_constant(step_size=0.0065, start=[0, 0], funcs=f, max_iter=10000, exp="exp/aii-gd-constant.csv"):
    start = np.array(start)
    g = sgd.StochasticGradientDescent()
    g.max_iter(max_iter)
    g.step_size(step_size)
    g.start(start)
    def function_generator():
        while True:
            yield funcs["function"], funcs["gradient"]
    g.function_generator(function_generator())
    g.debug(True)
    g.alg("constant")
    for i in range(max_iter):
        g.step()
        yield {
                "f(x)": g._function(g._x_value),
                "x": g._x_value,
        }

if __name__ == "__main__":

    plt.figure()

    for alpha in [0.004, 0.0035, 0.003, 0.0025]:
        res = list(gradient_descent_constant(max_iter=1000, step_size=alpha, funcs=g))
        res = pd.DataFrame(res)
        plt.plot(list(range(len(res["f(x)"]))),
                 res["f(x)"], label=f"$\\alpha={alpha}$")
    plt.title("Tuning step size for gradient descent on $g(x)$")
    plt.legend()
    plt.yscale('log')
    plt.tight_layout()
    plt.savefig('fig/aii-tune-g.pdf')
```