```python
import global_random_search
import lib
import numpy as np
import sgd
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
import pandas as pd
import time

f = {
    "function": lib.f_real,
    "gradient": lib.f_grad,
    "dname": "$f(x)$",
    "name": "f",
    "alpha": 0.0065,
}

g = {
    "function": lib.g_real,
    "gradient": lib.g_grad,
    "dname": "$g(x)$",
    "name": "g",
    "alpha": 0.003,
}


def gradient_descent_constant(step_size=0.0065, start=[0, 0], funcs=f, max_time=1):
    start = np.array(start)
    g = sgd.StochasticGradientDescent()
    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")
    start_time = time.perf_counter()
    current_time = 0
    while current_time < max_time:
        current_time = time.perf_counter() - start_time
        g.step()
        yield {
                "f(x)": g._function(g._x_value),
                "x": g._x_value,
                "time": time.perf_counter() - start_time,
        }

max_time=1
if __name__ == "__main__":
    for funcs in f, g:
        res = list(gradient_descent_constant(max_time=max_time, funcs=funcs, step_size=funcs["alpha"]))
        res = pd.DataFrame(res)

        plt.figure()

        for i in range(3):
            ps = [{"min": 0, "max": 10}, {"min": 0, "max": 18}]
            grs = global_random_search.a(
                costf=funcs["function"], parameters=ps, max_time=max_time)
            costs = grs['stats']['it_best_costs']
            plt.plot(grs['stats']['time'], costs, label="global random search", color="orange")
            print(funcs["name"], "total iterations global random search: ", len(grs['stats']['time']))


        plt.plot(res["time"], res["f(x)"], label="gradient descent", color="black")
        plt.title(f"Global Random Search vs Gradient Descent on {funcs['dname']}")
        custom_lines = [
                Line2D([0], [0], color='black', lw=2),
                Line2D([0], [0], color='orange', lw=2),
                ]
        custom_labels = ['gradient descent', 'rnd search a' ]
        plt.legend(custom_lines, custom_labels)
        plt.yscale('log')
        plt.xlabel("time (seconds)")
        plt.ylabel(funcs['dname'])
        plt.tight_layout()
        plt.savefig(f"fig/aii-time-{funcs['name']}.pdf")
        print(funcs["name"], "total iterations gradient descent: ", len(res))
```