```python
import cycler
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import yfinance as yf
from scipy.stats import norm

plt.style.use("seaborn-white")

# ====================================================================================
#                                      Figure 1
# ====================================================================================

df = yf.download("^FCHI")
df.columns = df.columns.str.lower()
log_return = np.log(df.close / df.close.shift(1))

mean = log_return.mean()
std = log_return.std(ddof=0)
x = np.linspace(log_return.min(), log_return.max(), 1000)

plt.figure(figsize=(10, 8))
log_return.plot(c="b")
plt.xlabel("Time", fontsize=15)
plt.ylabel("Log returns", fontsize=15)
plt.show()

# ====================================================================================
#                                      Figure 2
# ====================================================================================

plt.figure(figsize=(10, 8))

plt.hist(log_return, density=True, bins=200, color="b")
plt.plot(x, norm.pdf(x, loc=mean, scale=std), c="r", lw=3)
plt.xlabel("Log returns", fontsize=15)
plt.ylabel("Density", fontsize=15)
plt.legend(["Normal distribution", "Log returns distribution"])
plt.show()

# ====================================================================================
#                                      Figure 3
# ====================================================================================

put = Option(s0=36, v0=0.04, T=1, K=40, call=False)
heston = HestonProcess(mu=0.06, kappa=0.0005, theta=0.04, eta=0.1, rho=-0.5)

eta_list = np.linspace(0, 0.5, 50)[::-1]

plt.figure(figsize=(12, 8))
prices = []
for i, eta in eta_list:

    heston = HestonProcess(mu=0.06, kappa=0.0005, theta=0.04, eta=eta, rho=-0.5)
    price = monte_carlo_simulation_LS(option=put, process=heston, n=2_000_000, m=20)
    prices.append(price)

plt.plot(eta_list, prices, c="r", marker="o")
plt.axhline(y=4.487, c="k")
plt.annotate(
    "crr tree price",
    xy=(0.25, 4.5),
    xycoords="data",
    xytext=(0.35, 4.6),
    arrowprops=dict(facecolor="black", shrink=0.05),
    fontsize=15,
)
```

```python
plt.gca().invert_xaxis()
plt.xlabel("$\eta$", fontsize=20)
plt.ylabel("LS Option value", fontsize=15, labelpad=20)
plt.tick_params(axis="y", which="major", labelsize=18, pad=10)
plt.tick_params(axis="x", which="major", labelsize=21, pad=20)
plt.ylim(3.8, 4.7)
plt.show()

# ======================================================================================
#                                        Figure 4
# ======================================================================================

heston = HestonProcess(mu=0.06, kappa=0.0005, theta=0.04, eta=0.1, rho=-0.5)
s, v = heston.simulate(s0=60, v0=0.05, T=1, n=5, m=252, return_vol=True)

color = plt.cm.viridis(np.linspace(0, 1, 5))
mpl.rcParams["axes.prop_cycle"] = cycler.cycler("color", color)

plt.figure(figsize=(17, 7))
plt.subplot(1, 2, 1)
plt.plot(s)
plt.xlabel("$t$", fontsize=20)
plt.ylabel("$S_t$", fontsize=20, labelpad=20).set_rotation(0)
plt.title("Simulation of 5 Heston paths", fontsize=15, pad=10)

plt.subplot(1, 2, 2)
plt.plot(v)
plt.xlabel("$t$", fontsize=20)
plt.ylabel("$v_t$", fontsize=20, labelpad=20).set_rotation(0)
plt.title("Underlying volatility process", fontsize=15, pad=10)
plt.show()

# ======================================================================================
#                                        Figure 5
# ======================================================================================

m_list = range(1, 16 + 1)

put = Option(s0=36, v0=0.04, T=1, K=40, call=False)
heston = HestonProcess(mu=0.06, kappa=0.0005, theta=0.04, eta=0.1, rho=-0.5)
heston_euler = HestonProcess(
    mu=0.06, kappa=0.0005, theta=0.04, eta=0.1, rho=-0.5, milstein=False
)

exact_price = heston_semi_closed(option=put, process=heston)
print(exact_price)

errors = []
errors_euler = []

plt.figure(figsize=(12, 8))
for i, m in enumerate(m_list):
    print(i)
    mc_price = monte_carlo_simulation(option=put, process=heston, n=5_000_000, m=m)
    mc_price_euler = monte_carlo_simulation(
        option=put, process=heston, n=5_000_000, m=m
    )
    error = np.abs(exact_price - mc_price) / exact_price * 100
    error_euler = np.abs(exact_price - mc_price_euler) / exact_price * 100
    errors.append(error)
    errors_euler.append(error_euler)

plt.plot(m_list, errors, c="r", marker="o")
plt.plot(m_list, errors_euler, c="b", marker="o")

plt.xlabel("Number of discretization points", fontsize=20, labelpad=15)
plt.ylabel("Bias (%)", fontsize=20, labelpad=15)
plt.tick_params(axis="y", which="major", labelsize=18, pad=10)
plt.tick_params(axis="x", which="major", labelsize=21, pad=20)
```

```python
plt.legend(["Milstein", "Euler"], fontsize="xx-large")
plt.show()

# ==============================================================================
#                                    Figure 6
# ==============================================================================

put = Option(s0=36, v0=0.04, T=1, K=40, call=False)
gmb = GeometricBrownianMotion(mu=0.06, sigma=0.2)
heston = HestonProcess(mu=0.06, kappa=0.0005, theta=0.04, eta=0.1, rho=-0.5)

europeen_put_price = heston_semi_closed(option=put, process=heston)
price_crr = crr_pricing(r=0.06, sigma=0.2, option=put, n=25_000)

n = 100
price_ls_list = []
price_cd_list = []
price_cd2_list = []

for i in range(1, 100):

    price_gmb, X_gmb = monte_carlo_simulation_LS(
        option=put, process=gmb, n=n, m=100, return_all=True, seed=i
    )
    price_heston, Y_heston = monte_carlo_simulation_LS(
        option=put, process=heston, n=n, m=100, return_all=True, seed=i
    )

    price_mc, X_europeen = monte_carlo_simulation(
        option=put, process=heston, n=n, m=100, return_all=True, seed=i
    )
    price_ls, Y_americain = monte_carlo_simulation_LS(
        option=put, process=heston, n=n, m=100, return_all=True, seed=i
    )

    c_star_1 = np.cov(X_gmb, Y_heston)[0, 1] / np.cov(X_gmb, Y_heston)[1, 1]
    c_star_2 = (
        np.cov(X_europeen, Y_americain)[0, 1] / np.cov(X_europeen, Y_americain)[1, 1]
    )

    price_cd_1 = price_heston - c_star_1 * (price_gmb - price_crr)
    price_cd_2 = price_ls - c_star_2 * (price_mc - europeen_put_price)

    price_ls_list.append(price_heston)
    price_cd_list.append(price_cd_1)
    price_cd2_list.append(price_cd_2)

results = pd.DataFrame(
    {
        f"Standard estimator (Var = {np.round(np.var(price_ls_list), 3)})": price_ls_list,
        f"Europeen put control variate (Var = {np.round(np.var(price_cd2_list), 3)})": pric
e_cd2_list,
        f"GMB control variate (Var = {np.round(np.var(price_cd_list), 3)})": price_cd_list,
    }
)


plt.figure(figsize=(20, 8))
sns.boxplot(data=results)
plt.tick_params(axis="y", which="major", labelsize=18, pad=10)
plt.tick_params(axis="x", which="major", labelsize=21, pad=15)
plt.show()

# ==============================================================================
#                                    Figure 7
# ==============================================================================

put = Option(s0=36, v0=0.04, T=1, K=40, call=False)
heston = HestonProcess(mu=0.06, kappa=0.0005, theta=0.04, eta=0.1, rho=-0.5)
```

```
n = 3000
price_list = []
price_antithetic_list = []

for i in range(1, 100):

    price = monte_carlo_simulation_LS(option=put, process=heston, n=n, m=20, seed=i)
    price_antithetic = monte_carlo_simulation_LS(
        option=put, process=heston, n=n, m=20, antithetic=True, seed=i
    )

    price_antithetic = (price + price_antithetic) / 2

    price_list.append(price)
    price_antithetic_list.append(price_antithetic)

results = pd.DataFrame(
    {
        f"Standard estimator (Var = {np.round(np.var(price_list), 3)})": price_list,
        f"Antithetic estimator (Var = {np.round(np.var(price_antithetic_list), 3)})": price
_antithetic_list,
    }
)

plt.figure(figsize=(14, 8))
sns.boxplot(data=results)
plt.tick_params(axis="y", which="major", labelsize=18, pad=10)
plt.tick_params(axis="x", which="major", labelsize=21, pad=15)
plt.show()

# ==========================================================================================
#                                      Figure 8
# ==========================================================================================

put = Option(s0=36, v0=0.04, T=1, K=40, call=False)
heston = HestonProcess(mu=0.06, kappa=0.0005, theta=0.04, eta=0.1, rho=-0.5)

_, X_americain = monte_carlo_simulation_LS(
    option=put, process=heston, n=5000, m=20, return_all=True, seed=1
)

p = pd.Series(X_americain)

plt.figure(figsize=(12, 7))
p.expanding().mean().iloc[:3000].plot(c="r")
plt.ylim(2.5, 6)
plt.xlabel("Number of simulations", fontsize=20)
plt.ylabel("Price of the option", fontsize=20, labelpad=15)
plt.tick_params(axis="y", which="major", labelsize=18, pad=15)
plt.tick_params(axis="x", which="major", labelsize=21, pad=20)
plt.show()
```