```
import numpy as np
from scipy import integrate
from scipy.stats import norm, lognorm

# Parameters
stock_price = 100.0
strike_price = 95.0
risk_free_rate = 0.05
time_to_maturity = 0.5
volatility = 0.2
num_mc_simulations = int(1e7)
```

Black-Scholes formula

The Black-Scholes formula for call and put option are:

$$\begin{aligned} C(S, K, r, T, \sigma) &= SN(d_1) - Ke^{-rT}N(d_2) \\ P(S, K, r, T, \sigma) &= Ke^{-rT}N(-d_2) - SN(-d_1) \\ &= Ke^{-rT}(1 - N(d_2)) - S(1 - N(d_1)) \end{aligned}$$

where

$$\begin{aligned} d_1 &= \frac{1}{\sigma\sqrt{T}}\left[\log\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)T\right] \\ d_2 &= d_1 - \sigma\sqrt{T} \end{aligned}$$

```
def black_scholes(S, K, r, T, sigma, option_type):
    d1 = (np.log(S / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    Q1 = norm.cdf(d1)
    Q2 = norm.cdf(d2)

    if option_type == 'call':
        option_price = S * Q1 - K * np.exp(-r * T) * Q2
    elif option_type == 'put':
        option_price = K * np.exp(-r * T) * (1-Q2) - S * (1-Q1)

    return option_price

call_BS = black_scholes(stock_price, strike_price, risk_free_rate,
                                  time_to_maturity, volatility, 'call')
put_BS = black_scholes(stock_price, strike_price, risk_free_rate,
                                 time_to_maturity, volatility, 'put')

print(f'Call option price (BS):{call_BS:.4f}')
print(f'Put option price (BS):{put_BS:.4f}')
```

```
Call option price (BS):9.8727
Put option price (BS):2.5272
```

Monte Carlo simulation

The arbitrage-free values can be obtained through risk-neutral pricing approah:

$$C(S, K, r, T, \sigma) = \mathbb{E}^{\mathbb{Q}}\left[e^{-rT}(S_T - K)^+ \,\middle|\, S\right]$$

$$P(S, K, r, T, \sigma) = \mathbb{E}^{\mathbb{Q}}\left[e^{-rT}(K - S_T)^+ \,\middle|\, S\right]$$

Under the risk-neutral measure $\mathbb{Q}$, the underlying S_t grows at a risk-free rate r with diffusion:

$$dS_t = rS_t dt + \sigma S_t dW_t$$

```
def monte_carlo(S, K, r, T, sigma, option_type, num_simulations):

    # Generate random price paths
    drift = (r - 0.5 * sigma ** 2) * T
    diffusion = sigma * np.sqrt(T) * np.random.standard_normal(num_simulations)
```

```
    prices = S * np.exp(drift + diffusion)

    if option_type == 'call':
        payoffs = np.maximum(prices - K, 0)
    elif option_type == 'put':
        payoffs = np.maximum(K - prices, 0)

    dis_payoffs = np.exp(-r * T) * payoffs
    option_price = np.mean(dis_payoffs)

    # Calculate estimated standard error
    standard_error = np.sqrt(np.var(dis_payoffs) / num_simulations)

    return option_price, standard_error

call_MC, call_standard_error = monte_carlo(stock_price, strike_price, risk_free_rate,
                                  time_to_maturity, volatility, 'call',num_mc_simulations)
put_MC, put_standard_error = monte_carlo(stock_price, strike_price, risk_free_rate,
                                 time_to_maturity, volatility, 'put',num_mc_simulations)

print(f'Call option price (MC):{call_MC:.4f}')
print(f'Estimated standard error:{call_standard_error:.8f}')
print(f'Put option price (MC):{put_MC:.4f}')
print(f'Estimated standard error:{put_standard_error:.8f}')
```

```
Call option price (MC):9.8788
Estimated standard error:0.00356299
Put option price (MC):2.5259
Estimated standard error:0.00158613
```

Numerical integration

Recall that $S_T \sim$ Log–Normal($(r - \frac{\sigma^2}{2})T,\ \sigma^2 T$) under the risk-neutral measure $\mathbb{Q}$. Hence, we can evaluate the expectation through numerical integration. Let f denote the probability density function of the lognoraml random variabel S_T.

$$C(S, K, r, T, \sigma) = \mathbb{E}^{\mathbb{Q}}\left[e^{-rT}(S_T - K)^+ \,\middle|\, S\right] = \int_k^\infty e^{-rT}(s_T - K)f(s_T)ds_T$$
$$P(S, K, r, T, \sigma) = \mathbb{E}^{\mathbb{Q}}\left[e^{-rT}(K - S_T)^+ \,\middle|\, S\right] = \int_0^k e^{-rT}(K - s_T)f(s_T)ds_T$$

```
def numerical_integration(S, K, r, T, sigma, option_type):
  mu = np.log(S) + ( r - 0.5*sigma**2 ) * T
  sig = sigma * np.sqrt(T)

  if option_type == 'call':
        f_c = lambda s: np.exp(-r*T) * (s-K) * lognorm.pdf(s, sig, scale=np.exp(mu))
        return integrate.quad(f_c, K, np.inf)[0]
  elif option_type == 'put':
        f_p = lambda s: np.exp(-r*T) * (K-s) * lognorm.pdf(s, sig, scale=np.exp(mu))
        return integrate.quad(f_p, 0, K)[0]

call_integration = numerical_integration(stock_price, strike_price, risk_free_rate,
                                time_to_maturity, volatility, 'call')
put_integration = numerical_integration(stock_price, strike_price, risk_free_rate,
                                time_to_maturity, volatility, 'put')

print(f'Call option price (int):{call_integration:.4f}')
print(f'Put option price (int):{put_integration:.4f}')
```

```
Call option price (int):9.8727
Put option price (int):2.5272
```

Fourier inversion

$$
\begin{aligned}
C(S,K,r,T,\sigma) &= \mathbb{E}^{\mathbb{Q}}\left[e^{-rT}(S_T-K)^+ \,\middle|\, S\right] \\
&= \int_\Omega s_T e^{-rT}\mathbb{1}_{\{s_T>K\}}\, d\mathbb{Q} - Ke^{-rT}\mathbb{Q}(S_T>K) \\
&= S\tilde{\mathbb{Q}}\{S_T>K\} - Ke^{-rT}\mathbb{Q}(S_T>K) \\
&= SQ_1 - Ke^{-rT}Q_2 \\
P(S,K,r,T,\sigma) &= Ke^{-rT}(1-Q_2) - S(1-Q_1) \\
\text{where } \frac{d\tilde{\mathbb{Q}}}{d\mathbb{Q}} &= \frac{S_t}{Se^{rt}},\ Q_1 := \tilde{\mathbb{Q}}\{S_T>K\},\ Q_2 := \mathbb{Q}(S_T>K)
\end{aligned}
$$

Gil Pelaez formula, arised from Fourier inversion theorem, connects probability with it Fourier dual:

$$
\begin{aligned}
k &:= log(\frac{K}{S}) \\
X &:= log(\frac{S_T}{S}) \sim \text{Normal}(\ (r-\frac{\sigma^2}{2})T,\ \sigma^2 T\ ) \text{ under } \mathbb{Q} \\
Q1 &:= \tilde{Q}\{S_T>K\} = \tilde{Q}\{X>k\} \\
&= \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-iuk}\phi_X(u-i)}{iu\,\phi_X(-i)}\right]du \\
Q2 &:= Q\{S_T>K\} \\
&= \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-iuk}\phi_X(u)}{iu}\right]du \\
&\text{where } \phi_X(u) = E^{\mathbb{Q}}[e^{iuX}]
\end{aligned}
$$

```
def fourier_inversion(S, K, r, T, sigma, option_type):
    k = np.log(K/S)
    cf = lambda u: np.exp( 1j * u * (r - 0.5 * sigma**2) * T
                          - 0.5 * u**2 * T * sigma**2 )
    integrand_Q1 = lambda u: np.real((np.exp(-u*k*1j) * cf(u-1j) / (u*1j))  / cf(-1j))
    integrand_Q2 = lambda u: np.real(np.exp(-u*k*1j) * cf(u) /(u*1j))

    Q1 = 1/2 + 1/np.pi * integrate.quad(integrand_Q1, 1e-9, np.inf )[0]
    Q2 = 1/2 + 1/np.pi * integrate.quad(integrand_Q2, 1e-9, np.inf )[0]

    if option_type == 'call':
        return S * Q1 - K * np.exp(-r*T) * Q2
    elif option_type == 'put':
        return K * np.exp(-r*T) * (1-Q2) - S * (1-Q1)

call_F = fourier_inversion(stock_price, strike_price, risk_free_rate,
                                time_to_maturity, volatility, 'call')
put_F = fourier_inversion(stock_price, strike_price, risk_free_rate,
                                time_to_maturity, volatility, 'put')

print(f'Call option price (Fourier):{call_F:.4f}')
print(f'Put option price (Fourier):{put_F:.4f}')
```

```
Call option price (Fourier):9.8727
Put option price (Fourier):2.5272
```

Technical Note - $N(d_1)$ and $N(d_2)$

$N(d_1)$ and $N(d_2)$ are the risk-neutral probabilities of $S_T > K$ under the stock and money market account numeraires, respectively. To see this:

$$
\begin{aligned}
C(S,K,r,T,\sigma) &= \mathbb{E}^{\mathbb{Q}}\left[e^{-rT}(S_T-K)^+ \,\middle|\, S\right] \\
&= \mathbb{E}^{\mathbb{Q}}\left[e^{-rT}S_T\mathbb{1}_{S_T>K} \,\middle|\, S\right] - Ke^{-rT}\mathbb{E}^{\mathbb{Q}}\left[\mathbb{1}_{S_T>K} \,\middle|\, S\right] \\
&= S\mathbb{E}^{\tilde{\mathbb{Q}}}\left[\mathbb{1}_{S_T>K} \,\middle|\, S\right] - Ke^{-rT}\mathbb{E}^{\mathbb{Q}}\left[\mathbb{1}_{S_T>K} \,\middle|\, S\right] \\
&= SN(d_1) - Ke^{-rT}N(d_2)
\end{aligned}
$$

where the measure $\tilde{\mathbb{Q}}$ (in which S_t becomes the numeraire) is defined as:

$$\frac{d\tilde{\mathbb{Q}}}{d\mathbb{Q}} = \frac{S_t}{Se^{rt}}$$

Technical Note - Gil Pelaez formula

$$\begin{aligned}
F_X(x) &= \frac{1}{2} - \frac{1}{2\pi}\int_{\mathbb{R}} e^{-iux}\phi_X(u) \cdot \frac{1}{iu}\, du \\
&= \frac{1}{2} - \frac{1}{2\pi}\int_0^{\infty} \left(-e^{iux}\phi_X(-u) + e^{-iux}\phi_X(u)\right)\frac{1}{iu}\, du \\
&= \frac{1}{2} - \frac{1}{\pi}\int_0^{\infty} \frac{Im[e^{-iux}\phi_X(u)]}{u} du \\
&= \frac{1}{2} - \frac{1}{\pi}\int_0^{\infty} Re\left[\frac{e^{-iux}\phi_X(u)}{iu}\right] du \\
f_X(x) &= \frac{1}{\pi}\int_0^{\infty} Re\left[e^{-iux}\phi_X(u)\right] du
\end{aligned}$$