```
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(1e6)
```

Black-Scholes formula

The **BS formula** for a call and put are:

$$C(S, K, r, T, \sigma) = SN(d_1) - Ke^{-rT}N(d_2)$$

$$P(S, K, r, T, \sigma) = Ke^{-rT}N(-d_2) - SN(-d_1)$$

where

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

$$d_2 = d_1 - \sigma\sqrt{T}$$

```
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)

    if option_type == 'call':
        option_price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    elif option_type == 'put':
        option_price = K * np.exp(-r * T) * norm.cdf(-d2) - S * norm.cdf(-d1)

    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 value of an option 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:

$$S_T = Se^{(r - \frac{\sigma^2}{2})T + \sigma W_T}$$

```
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):9.8527
Estimated standard error:0.01125805
```

```
Put option price (MC):2.5269
Estimated standard error:0.00501392
```

Numerical integration

Notice that $S_T \sim \text{Log–Normal}(\ (r - \frac{\sigma^2}{2})T,\ \sigma^2 T\ )$ under the risk-neutral measure $\mathbb{Q}$.

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

```
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)*max(K-s,0)*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
```

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 numeraires, respectively. To see this:

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

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

$$
\frac{d\tilde{\mathbb{Q}}}{d\mathbb{Q}} = \frac{S_T e^{r0}}{S e^{rT}}
$$