In [ ]:
```python
# Import Libraries
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
from statsmodels.stats.proportion import proportions_ztest
```

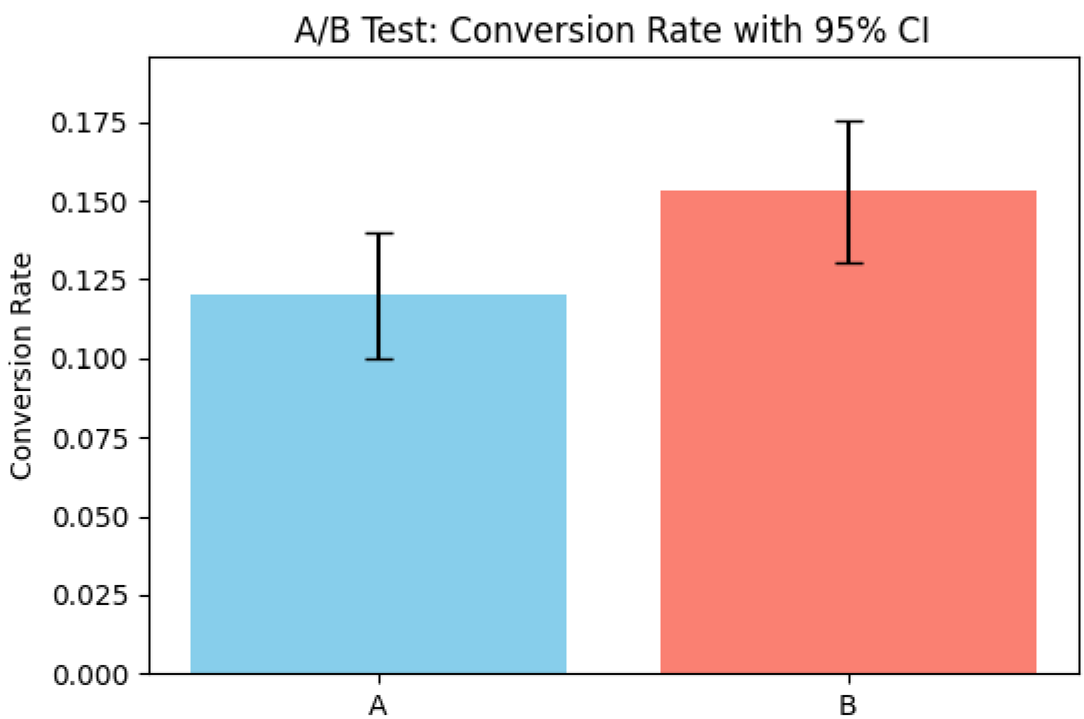
In [ ]:
```python
# simulate data
np.random.seed(42)

# Assume 10,000 visitors each
n_A, p_A = 10000, 0.12  # Variant A: 12% conversion
n_B, p_B = 10000, 0.10  # Variant B: 10% conversion
```

In [ ]:
```python
# simulate number of purchases
success_A = np.random.binomial(n_A, p_A)
success_B = np.random.binomial(n_B, p_B)
```

In [3]:
```python
import numpy as np
import pandas as pd
from scipy import stats

#  Sample A/B Test Data (Replace with your real data)
n_A = 1000       # Visitors in Variant A
success_A = 120  # Conversions in Variant A

n_B = 980        # Visitors in Variant B
success_B = 150  # Conversions in Variant B

#  Function to calculate conversion rate & 95% Confidence Interval
def proportion_ci(successes, n, alpha=0.05):
    p_hat = successes / n
    z = stats.norm.ppf(1 - alpha/2)  # z-value for 95% CI
    se = np.sqrt(p_hat * (1 - p_hat) / n)  # standard error
    return p_hat, (p_hat - z * se), (p_hat + z * se)

#  Compute for Variant A
p_A_hat, ci_low_A, ci_high_A = proportion_ci(success_A, n_A)

#  Compute for Variant B
p_B_hat, ci_low_B, ci_high_B = proportion_ci(success_B, n_B)

#  Combine results into a table
results = pd.DataFrame({
    'Variant':          ['A', 'B'],
    'Visitors':         [n_A, n_B],
    'Conversions':      [success_A, success_B],
    'CR (p̂)':          [p_A_hat, p_B_hat],
    'CI Lower (95%)':   [ci_low_A, ci_low_B],
    'CI Upper (95%)':   [ci_high_A, ci_high_B]
})

print(results)
```

```
  Variant  Visitors  Conversions    CR (p̂)  CI Lower (95%)  CI Upper (95%)
0       A      1000          120  0.120000        0.099859        0.140141
1       B       980          150  0.153061        0.130519        0.175603
```

In [4]:
```
# Plot Conversion Rates with Error Bars
plt.figure(figsize=(6,4))
plt.bar(
    results['Variant'],
    results['CR (p̂)'],
    yerr=[
        results['CR (p̂)'] - results['CI Lower (95%)'],
        results['CI Upper (95%)'] - results['CR (p̂)']
    ],
    capsize=5,
    color=['skyblue', 'salmon']
)
plt.ylabel('Conversion Rate')
plt.title('A/B Test: Conversion Rate with 95% CI')
plt.ylim(0, max(results['CI Upper (95%)']) + 0.02)
plt.show()
```



In [7]:
```
import numpy as np
from scipy.stats import norm

#  Define your A/B test data
n_A = 1000       # visitors in Variant A
success_A = 120  # conversions in Variant A

n_B = 980        # visitors in Variant B
success_B = 150  # conversions in Variant B
```

```python
#  Conversion rates
p_A = success_A / n_A
p_B = success_B / n_B

#  Pooled proportion
p_pool = (success_A + success_B) / (n_A + n_B)

#  Standard error under H0
se = np.sqrt(p_pool * (1 - p_pool) * (1/n_A + 1/n_B))

#  Z-statistic for one-sided test (H1: pB > pA)
z_stat = (p_B - p_A) / se

#  One-sided p-value
p_value = 1 - norm.cdf(z_stat)

print(f"Z-statistic: {z_stat:.3f}")
print(f"p-value:     {p_value:.3f}")

if p_value < 0.05:
    print("→ Reject H0: Variant B has a significantly higher conversion rate."
else:
    print("→ Fail to reject H0: No significant lift from B over A.")
```

```
Z-statistic: 2.143
p-value:     0.016
→ Reject H0: Variant B has a significantly higher conversion rate.
```

In [ ]:

```python
import numpy as np, matplotlib.pyplot as plt, time
from statsmodels.stats.proportion import proportions_ztest
from IPython.display import clear_output

# Use the same true rates from above
true_p_A, true_p_B = 0.10, 0.12
batch_size = 100  # visitors per batch per variant
n_batches = 60    # simulate 60 time steps (e.g. minutes)

# Initiate Counters
n_visits_A = n_visits_B = 0
n_success_A = n_success_B = 0

# List to store metrices for plotting
batches = []
p_value = []
lifts   = []
```

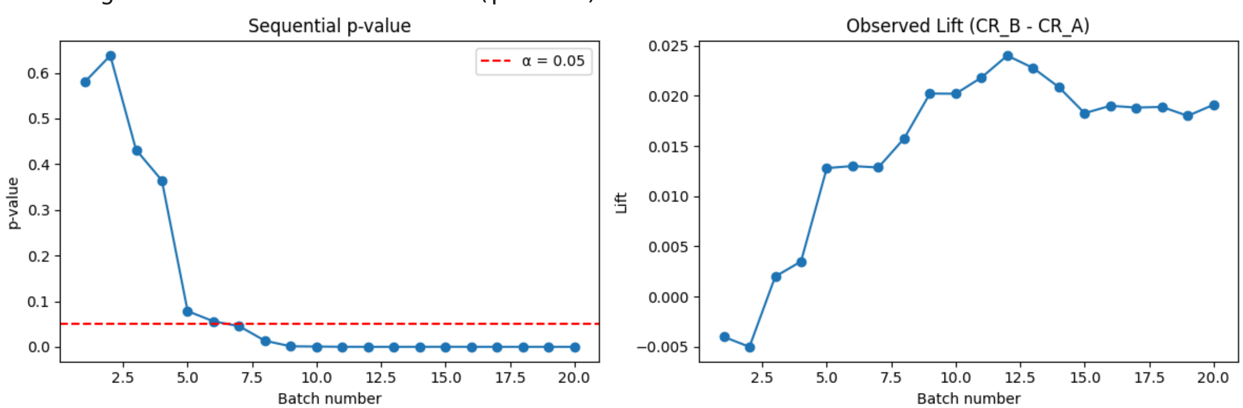
In [10]:

```python
import numpy as np
import matplotlib.pyplot as plt
import time
from IPython.display import clear_output
from scipy.stats import norm  # manual z-test implementation

# ◈ Parameters
n_batches   = 20        # number of batches
```

```
batch_size  = 500       # visitors per batch
true_p_A    = 0.10      # true conversion rate for A (10%)
true_p_B    = 0.12      # true conversion rate for B (12%)

# � Initialize counters
n_visits_A  = 0
n_visits_B  = 0
n_success_A = 0
n_success_B = 0

# � For plotting
batches   = []
p_values  = []
lifts     = []

# � Manual two-proportion z-test function (no statsmodels)
def two_proportion_ztest(success_A, visits_A, success_B, visits_B):
    p_A = success_A / visits_A
    p_B = success_B / visits_B
    p_pool = (success_A + success_B) / (visits_A + visits_B)
    se = np.sqrt(p_pool * (1 - p_pool) * (1/visits_A + 1/visits_B))
    z = (p_B - p_A) / se
    p_val = 1 - norm.cdf(z)  # one-sided test: B > A
    return z, p_val

# � Sequential simulation
for batch in range(1, n_batches + 1):
    # simulate one batch of visitors
    new_A = np.random.binomial(batch_size, true_p_A)
    new_B = np.random.binomial(batch_size, true_p_B)

    # update totals
    n_visits_A += batch_size
    n_visits_B += batch_size
    n_success_A += new_A
    n_success_B += new_B

    # current conversion rates
    cr_A = n_success_A / n_visits_A
    cr_B = n_success_B / n_visits_B
    lift = cr_B - cr_A

    # two-proportion z-test
    z_stat, p_val = two_proportion_ztest(n_success_A, n_visits_A, n_success_B,

    # record for plotting
    batches.append(batch)
    p_values.append(p_val)
    lifts.append(lift)

    # clear output for live update
    clear_output(wait=True)
    print(f"Batch {batch}/{n_batches}")
```

```python
    print(f" Variant A: {n_visits_A} visits, {n_success_A} buys → CR = {cr_A:.
    print(f" Variant B: {n_visits_B} visits, {n_success_B} buys → CR = {cr_B:.
    print(f" Observed lift: {lift:.3%}")
    print(f" z-stat = {z_stat:.2f}, p-value = {p_val:.4f}")
    if p_val < 0.05:
        print("  → Significant lift detected (p<0.05).")
    else:
        print("  → No significant lift yet.")

    # plot p-value & lift trends
    fig, axes = plt.subplots(1, 2, figsize=(12, 4))

    axes[0].plot(batches, p_values, '-o')
    axes[0].axhline(0.05, color='red', linestyle='--', label='α = 0.05')
    axes[0].set_title('Sequential p-value')
    axes[0].set_xlabel('Batch number')
    axes[0].set_ylabel('p-value')
    axes[0].legend()

    axes[1].plot(batches, lifts, '-o')
    axes[1].set_title('Observed Lift (CR_B - CR_A)')
    axes[1].set_xlabel('Batch number')
    axes[1].set_ylabel('Lift')

    plt.tight_layout()
    plt.show()

    time.sleep(0.8)  # pause to simulate real-time
```

```
Batch 20/20
 Variant A: 10000 visits, 1005 buys → CR = 10.050%
 Variant B: 10000 visits, 1196 buys → CR = 11.960%
 Observed lift: 1.910%
 z-stat = 4.32, p-value = 0.0000
  → Significant lift detected (p<0.05).
```



In [ ]: