Q1. Python function to calculate F-value and p-value (variance ratio test)

```
import numpy as np
from scipy.stats import f

def f_test(data1, data2):
    var1 = np.var(data1, ddof=1)
    var2 = np.var(data2, ddof=1)

    # Ensure F ≥ 1
    if var1 >= var2:
        F = var1 / var2
        dfn = len(data1) - 1
        dfd = len(data2) - 1
    else:
        F = var2 / var1
        dfn = len(data2) - 1
        dfd = len(data1) - 1

    p_value = 2 * (1 - f.cdf(F, dfn, dfd))
    return F, p_value
```

Q2. Python function to return critical F-value (two-tailed test)

```
from scipy.stats import f

def critical_f_value(alpha, dfn, dfd):
    return f.ppf(1 - alpha/2, dfn, dfd)

critical_f_value(0.05, 10, 12)
```

Q3. Random samples + F-test (Python program)

```
import numpy as np
from scipy.stats import f

np.random.seed(0)

sample1 = np.random.normal(0, 4, 30)  # variance = 16
sample2 = np.random.normal(0, 3, 25)  # variance = 9

var1 = np.var(sample1, ddof=1)
var2 = np.var(sample2, ddof=1)

F = var1 / var2
dfn = len(sample1) - 1
dfd = len(sample2) - 1

p_value = 2 * (1 - f.cdf(F, dfn, dfd))

print("F-value:", F)
print("Degrees of freedom:", (dfn, dfd))
print("p-value:", p_value)
```

Q4. Variances = 10 and 15, $n_1 = n_2 = 12$

F=1510=1.5F = \frac{15}{10} = 1.5F=1015=1.5

Degrees of freedom:

- dfn = 11
- dfd = 11

Critical F (α = 0.05, two-tailed) ≈ **3.59**

✅ Conclusion:

Since **1.5 < 3.59**, **fail to reject H_0**
➡ Variances are **not significantly different**

Q5. Manufacturer's claim (α = 0.01)

F=0.0060.005=1.2F = \frac{0.006}{0.005} = 1.2F=0.0050.006=1.2

Degrees of freedom:

- dfn = 24
- dfd = ∞ (population variance assumed)

Critical F (α = 0.01) ≈ **2.54**

✅ Conclusion:

Since **1.2 < 2.54**, **fail to reject H_0**
➡ Manufacturer's claim is **justified**

Q6. Python function for mean & variance of F-distribution

Formula:

- Mean = d2d2−2\frac{d_2}{d_2 - 2}d2−2d2, for d2>2d_2 > 2d2>2
- Variance = 2d22(d1+d2−2)d1(d2−2)2(d2−4)\frac{2d_2^2(d_1 + d_2 - 2)}{d_1(d_2 - 2)^2(d_2 - 4)}d1(d2−2)2(d2−4)2d22(d1+d2−2), for d2>4d_2 > 4d2>4

```
def f_distribution_stats(dfn, dfd):
    mean = dfd / (dfd - 2) if dfd > 2 else None
    variance = (2 * dfd**2 * (dfn + dfd - 2)) / (dfn * (dfd - 2)**2 *
(dfd - 4)) if dfd > 4 else None
    return mean, variance
```

Q7. Sample variances = 25 and 20, α = 0.10

F=2520=1.25F = \frac{25}{20} = 1.25F=2025=1.25

Degrees of freedom:

- dfn = 9
- dfd = 14

Critical F (α = 0.10) ≈ **2.35**

✅ Conclusion:

Since **1.25 < 2.35**, **fail to reject H_0**
➡ Variances are **not significantly different**

Q8. Restaurant waiting times (α = 0.05)

Data

- Restaurant A: 24, 25, 28, 23, 22, 20, 27
- Restaurant B: 31, 33, 35, 30, 32, 36

Python solution

```
import numpy as np
from scipy.stats import f

A = np.array([24,25,28,23,22,20,27])
B = np.array([31,33,35,30,32,36])

varA = np.var(A, ddof=1)
varB = np.var(B, ddof=1)

F = max(varA, varB) / min(varA, varB)
dfn = len(A)-1
dfd = len(B)-1

p_value = 2 * (1 - f.cdf(F, dfn, dfd))
F, p_value
```

✅ Conclusion:

If **p-value > 0.05**, variances are **not significantly different**

Q9. Test scores variance test (α = 0.01)

Data

- Group A: 80, 85, 90, 92, 87, 83
- Group B: 75, 78, 82, 79, 81, 84

```
A = np.array([80,85,90,92,87,83])
B = np.array([75,78,82,79,81,84])

varA = np.var(A, ddof=1)
varB = np.var(B, ddof=1)

F = max(varA, varB) / min(varA, varB)
dfn = len(A)-1
dfd = len(B)-1

p_value = 2 * (1 - f.cdf(F, dfn, dfd))
F, p_value
```

✅ Conclusion:

At **1% significance**, if **p-value > 0.01**,
➡ **Fail to reject H_0** (variances are equal)
