Assignment 2

```
import pandas as pd

import numpy as np

from scipy.stats import ttest_ind

from statsmodels.stats.multitest import multipletests

# Simulated RNA-Seq Data Generation

np.random.seed(42)

genes = ["Gene1", "Gene2", "Gene3", "Gene4"]

sample1_counts = np.random.poisson(lam=100, size=10)  # 10 replicates for sample 1

sample2_counts = np.random.poisson(lam=80, size=10)  # 10 replicates for sample 2


# Create a DataFrame for counts

data = pd.DataFrame({

    "Gene": genes,

    "Sample1_Counts": [np.random.poisson(lam=100, size=10) for _ in genes],

    "Sample2_Counts": [np.random.poisson(lam=80, size=10) for _ in genes]

})

# Calculate mean counts for each gene

data["Sample1_Mean"] = [np.mean(counts) for counts in data["Sample1_Counts"]]

data["Sample2_Mean"] = [np.mean(counts) for counts in data["Sample2_Counts"]]


# Differential Expression Analysis

p_values = []

for _, row in data.iterrows():

    _, p_value = ttest_ind(row["Sample1_Counts"], row["Sample2_Counts"])

    p_values.append(p_value)


data["PValue"] = p_values
```

```
# Correct p-values for multiple testing (e.g., using FDR)
data["AdjPValue"] = multipletests(data["PValue"], method='fdr_bh')[1]


# Identify Differentially Expressed Genes
differential_genes = data[data["AdjPValue"] < 0.05]
# Simulated Gene Ontology terms for demonstration
annotations = {
    "Gene1": "GO:0001234,GO:5678901",
    "Gene2": "GO:2345678,GO:8901234",
    "Gene3": "GO:1234567",
    "Gene4": "GO:5678901,GO:2345678"
}


# Functional Annotation and Biological Interpretation
differential_genes["GO_Annotations"] = differential_genes["Gene"].map(annotations)
report_lines = []
report_lines.append("RNA-Seq Differential Expression Analysis Report\n")
report_lines.append("="*50)
report_lines.append("Summary of Analysis:\n")
report_lines.append("The analysis was conducted to identify differentially expressed genes between
two conditions using RNA-Seq data.\n")
report_lines.append("="*50)


report_lines.append("Differentially Expressed Genes:\n")
if not differential_genes.empty:
    report_lines.append(f"{'Gene':<10} {'Mean Sample1':>15} {'Mean Sample2':>15} {'Adj.
P-Value':>20}")
    report_lines.append("="*60)
    for _, row in differential_genes.iterrows():
        report_lines.append(f"{row['Gene']:<10} {row['Sample1_Mean']:>15.2f}
{row['Sample2_Mean']:>15.2f} {row['AdjPValue']:>20.4f}")
else:
    report_lines.append("No genes were found to be differentially expressed.\n")

report_lines.append("="*50)
report_lines.append("Functional Annotations:\n")
if not differential_genes.empty:
    report_lines.append(f"{'Gene':<10} {'GO Annotations':<30}")
    report_lines.append("="*40)
    for _, row in differential_genes.iterrows():
        go_terms = row["GO_Annotations"] if pd.notna(row["GO_Annotations"]) else "No annotations available"
        report_lines.append(f"{row['Gene']:<10} {go_terms:<30}")
else:
    report_lines.append("No functional annotations available.\n")

# Save the report to a file
with open("RNA_SEQ_ANALYSIS.txt", "w") as report_file:
    report_file.write("\n".join(report_lines))

print("Analysis report generated as 'RNA_SEQ_ANALYSIS.txt'.")
```