Session 1.1: Business Analytics Architecture, Google Antigravity IDE Initialization, and Core Arithmetic

TODAY'S PURPOSE

Why This Session Matters

- Set up your Python workspace
- Replace spreadsheet cell formulas with named variables
- Calculate real business metrics in code

Today's flow: 60 min lecture, then 120 min hands-on lab

EXPECTED OUTCOMES

By the end of today, you will be able to:

1. Run a script in Google Antigravity IDE
2. Compute operating expense and margin with variables
3. Fix a NameError or SyntaxError with AI Chat

Concepts for This Session

CONCEPT 1

Python Interpreter

The computational engine that reads Python source code and executes instructions sequentially line-by-line in real time without requiring compilation.

CONCEPT 2

Variable Assignment

Storing data in computer memory using an assignment operator (=) and an intuitive descriptive label (such as net_sales = 574785.0), replacing fragile cell coordinates like C13.

CONCEPT 3

Primitive Data Types

The foundational categories of information processed by Python: whole numbers (int), decimal numbers (float), and text strings (str).

CONCEPT 4

Traceback & Exception

The automatic diagnostic navigation report generated by Python when an instruction violates a syntax or runtime rule, pinpointing the exact file and line number.

How the Concept Works in Practice

FOUNDATION 1

Sequential Execution & Program State

Python executes code strictly line-by-line from top to bottom. Unlike spreadsheets where formulas across sheets can trigger circular reference loops, Python code maintains a clear, linear flow where each statement computes or alters memory state sequentially.

FOUNDATION 2

Cell Coordinate Formulas vs. Descriptive Variables

Transitioning from Excel spatial grid thinking (such as =C13-E13) to self-documenting computational recipes (operating_expense = net_sales - operating_income). Code variables retain clear business meaning and eliminate silent formula drift.

FOUNDATION 3

Interpreter Diagnostics & Traceback Mechanics

Python acts as an interactive assistant. When a syntax or naming rule is violated, Python pauses and outputs a Traceback pointing to the exact line number and cause, turning errors into immediate learning signals rather than system failures.

Build, Test, and Diagnose in the IDE

[1] FILE EXPLORER

Organize Assets

Use the session dataset and keep scripts in a clear workspace.

[2] CODE EDITOR

Write Small Segments

Run each completed segment before expanding the program.

[3] AI AGENT CHAT

Diagnose Errors

Capture the traceback, request an explanation, and review the fix.

[4] INTEGRATED TERMINAL

Verify Execution

Confirm the final program completes with return code 0.

Script Demonstration 1: Business Analytics Architecture, Google Antigravity IDE Initialization, and Core Arithmetic

```
# Amazon Consolidated Financial Metrics for FY2023 (in USD Millions)
    net_sales = 574785.0
    operating_income = 36852.0

    # Calculate Operating Expenses
    operating_expense = net_sales - operating_income

    # Calculate Operating Margin Percentage
    operating_margin_pct = (operating_income / net_sales) * 100.0

    print("Operating Expense (USD M):", operating_expense)
    print("Operating Margin (%):", round(operating_margin_pct, 2))
```

SYNTAX

Readable Structure

Observe the syntax and indentation that control execution.

RUNTIME

Verified Execution

Run the example in the integrated terminal and inspect the output.

BUSINESS VALUE

Applied Automation

Use the pattern to automate a repeatable business decision.

Script Demonstration 2: Business Analytics Architecture, Google Antigravity IDE Initialization, and Core Arithmetic

```
aws_sales = 90757.0
    aws_income = 24631.0
    aws_margin_pct = (aws_income / aws_sales) * 100.0
    print("AWS Operating Margin (%):", round(aws_margin_pct, 2))
```

SYNTAX

Readable Structure

Observe the syntax and indentation that control execution.

RUNTIME

Verified Execution

Run the example in the integrated terminal and inspect the output.

BUSINESS VALUE

Applied Automation

Use the pattern to automate a repeatable business decision.

Script Demonstration 3: Business Analytics Architecture, Google Antigravity IDE Initialization, and Core Arithmetic

```
print("Session practice: verify execution")
```

SYNTAX

Readable Structure

Observe the syntax and indentation that control execution.

RUNTIME

Verified Execution

Run the example in the integrated terminal and inspect the output.

BUSINESS VALUE

Applied Automation

Use the pattern to automate a repeatable business decision.

Exercise 1 and Exercise 2

EXERCISE 1: GUIDED

Foundational Practice

Follow the lab protocol, run the script, and verify its output with the session dataset.

EXERCISE 2: DEBUGGING

Applied Challenge

Extend the program, diagnose an intentional error, apply the verified correction, and confirm return code 0.