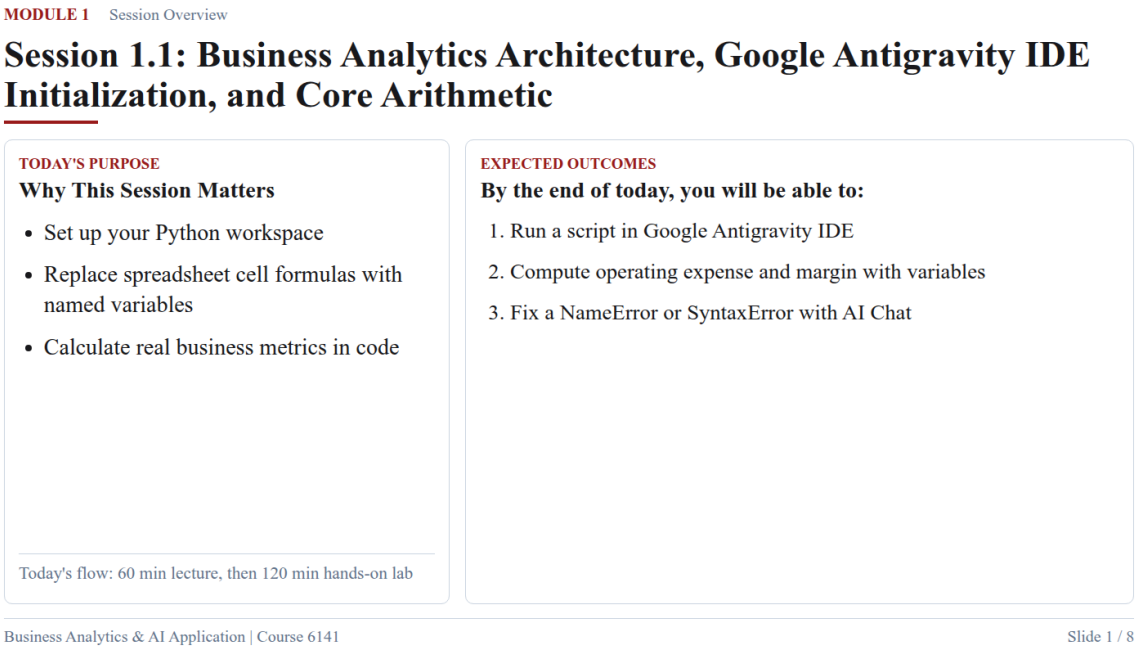
Session 1.1: Business Analytics Architecture, Google Antigravity IDE Initialization, and Core Arithmetic

Course 6141: Business Analytics & AI Application | Reading for Week 1



Schedule: Week 1 | In-person classes (1:2 Delivery Ratio: 60 Min Lecture + 120 Min Lab)

Session Purpose

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

Expected Learning Outcomes (By the end of this session, 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

Keywords (Core Programming Concepts)

- `Python Interpreter`: The computational engine that reads Python source code and executes instructions sequentially line-by-line in real time without requiring compilation.
- `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`.
- `Primitive Data Types`: The foundational categories of information processed by Python: whole numbers (`int`), decimal numbers (`float`), and text strings (`str`).
- `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.
- `Integrated Development Environment (IDE)`: A unified software workstation (Google Antigravity IDE) providing an editor, execution terminal, file explorer, and integrated AI assistant in a single interface.

Core Programming & Computational Foundations

- 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.
- 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.
- 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.

Hands-On Coding Micro-Examples

- *Micro-Example 1 (Variable Arithmetic):* Defining commercial numbers (`net_sales = 574785.0`, `operating_income = 36852.0`) and calculating Operating Expense and Operating Margin Percentage with standard math operators (`-`, `/`, `*`).
- *Micro-Example 2 (Dynamic Parameter Reusability):* Changing a single baseline assumption (such as revising sales tax or discounting prices by 5%) and observing how Python recalculates all downstream figures instantaneously without manual cell dragging.
- *Micro-Example 3 (Traceback Troubleshooting & AI Fix):* Creating an intentional variable typo (`netsales` instead of `net_sales`), observing the `NameError` in the terminal, capturing it with `Windows Key + Shift + S`, and pasting into the AI chat to verify how the co-pilot identifies the typo and supplies the correction.

Lab Practice Dataset

- File Link: `data/session_01_01_dataset.csv`
- Simple 4-row tabular financial dataset used to verify calculations:
  - `Segment_Name`: North America, International, AWS, Consolidated Total.

  - `Net_Sales_USD_M`: Revenue in millions of U.S. Dollars.
  - `Operating_Income_USD_M`: Operating income in millions of U.S. Dollars.

Official Python Documentation & References

- Python 3.12 Official Tutorial: Using Python as an Interactive Calculator.
- PEP 8: Style Guide for Python Code (Readability and Variable Naming Standards).
- Google Antigravity IDE Quickstart: Workspace Panels and AI Pair Programming.

Integrated Development Environment Architecture (Google Antigravity IDE)

- Students use Google Antigravity IDE, a modern AI-powered development environment designed for business analytics, financial modeling, and collaborative data science.

`!assets/antigravity_ide_interface.png`

- **The 4 Workspace Panels:**
  - `[1] File Explorer (Left Panel):` Workspace file hierarchy. Used to organize corporate datasets (`data/`), slide decks (`slides/`), Python scripts (`.py`), and curriculum documentation (`.md`).
  - `[2] Code Editor & Notebook Canvas (Center Top):` Primary workspace for writing Python code, defining financial variables, and executing arithmetic business formulas.
  - `[3] AI Agent Chat Canvas (Right Panel):` An interactive AI pair-programming assistant powered by Gemini. Students paste code snippets or error screenshots directly into this window to receive plain-English root cause explanations and verified code corrections.
  - `[4] Integrated Terminal & Console (Center Bottom):` Command line interface running the Python execution kernel. Displays script output results and detailed system error tracebacks.
- **AI Error Diagnosis Protocol (Capture and Resolve in Seconds):**
  - *Zero Fear Principle:* In Python, an error is not a system failure or a test penalty. It is simply a diagnostic navigation clue (Traceback) showing the exact line number and rule violation.
  - *Step A (Identify):* When code execution stops, look at the last line in Panel `[4]` (Terminal), which names the error type (e.g., `NameError`, `SyntaxError`, `TypeError`).
  - *Step B (Capture):* Take a fast screen capture of the error message using `Windows Key + Shift + S` (Windows) or `Command + Shift + 4` (macOS).
  - *Step C (Paste & Prompt):* Click inside Panel `[3]` (AI Agent Chat), press `Ctrl + V` to attach the captured image, and ask: *"Why did this error happen, and what is the exact corrected code?"*
  - *Step D (Apply):* Review the AI explanation, understand the typo or missing parenthesis, and copy the verified fix back into Panel `[2]`.

Step-by-Step Lab Protocol (Exercise 1: Foundational Practice)

- *Step 1: Open Environment & Dataset:* Launch Google Antigravity IDE. Place `data/session_01_01_dataset.csv` in your working directory and open the raw file via Panel `[1]` (File Explorer) to observe comma-separated tabular records.

- *Step 2: Inspect Primary Sources & Financial Terminology:* Confirm against Note 15 of Amazon's Form 10-K that Consolidated Net Sales for FY2023 equaled `$574,785` million and Operating Income equaled `$36,852` million.
- *Step 3: Define Variables and Execute First Python Calculations:* In your first Python notebook cell, type the following arithmetic statements and run the cell:

```
# 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))
```

- *Step 4: Bridge Excel Spreadsheet Mechanics to Python Code:*
  - In Microsoft Excel, if `Net_Sales` is in cell `C13` (`574785.00`) and `Operating_Income` is in cell `E13` (`36852.00`), the expense formula in cell `D13` is `=C13-E13`, and the margin formula in cell `F13` is `=(E13/C13)*100`.
  - Notice that Python variable assignment (`operating_income / net_sales`) performs the identical mathematical operation as cell references, but uses memorable descriptive English words instead of abstract grid coordinates like `E13`.
- *Step 5: Compare Segment Profitability:* Execute a comparative calculation for Amazon Web Services (AWS) in FY2023:

```
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))
```

  Confirm that the AWS operating margin (`27.14%`) is over four times higher than the consolidated corporate operating margin (`6.41%`), demonstrating how cloud infrastructure drives corporate operating cash flow.

Step-by-Step Applied Coding Practice (Exercise 2+: Hands-On Programming & Debugging)

- *Coding Challenge:* In Google Antigravity IDE, create a new script named `session_1_1_practice.py` to calculate financial metrics across multiple business units and practice error debugging.
- *Task 1 (Segment Margin Calculation):* Define variables for Amazon Web Services (AWS) using FY2023 figures (`aws_sales = 90757.0`, `aws_income = 24631.0`). Compute `aws_margin_pct` and print the formatted result with `round(..., 2)`.

- *Task 2 (Intentional Syntax Error & AI Diagnosis):* Deliberately omit the closing parenthesis in your `print()` statement (e.g., `print("AWS Margin:", round(aws_margin_pct, 2))`. Run the script in Panel `[4]` (Terminal), observe the `SyntaxError`, capture the terminal message using `Windows Key + Shift + S`, paste it into Panel `[3]` (AI Agent Chat) with `Ctrl + V`, and verify the AI's explanation and fix.
- *Task 3 (Multi-Unit Comparison Output):* Combine consolidated and cloud margin calculations into a single clean script that prints both metrics formatted with labels. Verify that the script executes with return code 0.