Introduction to Programming Concepts

Module Overview

This module will cover the following:

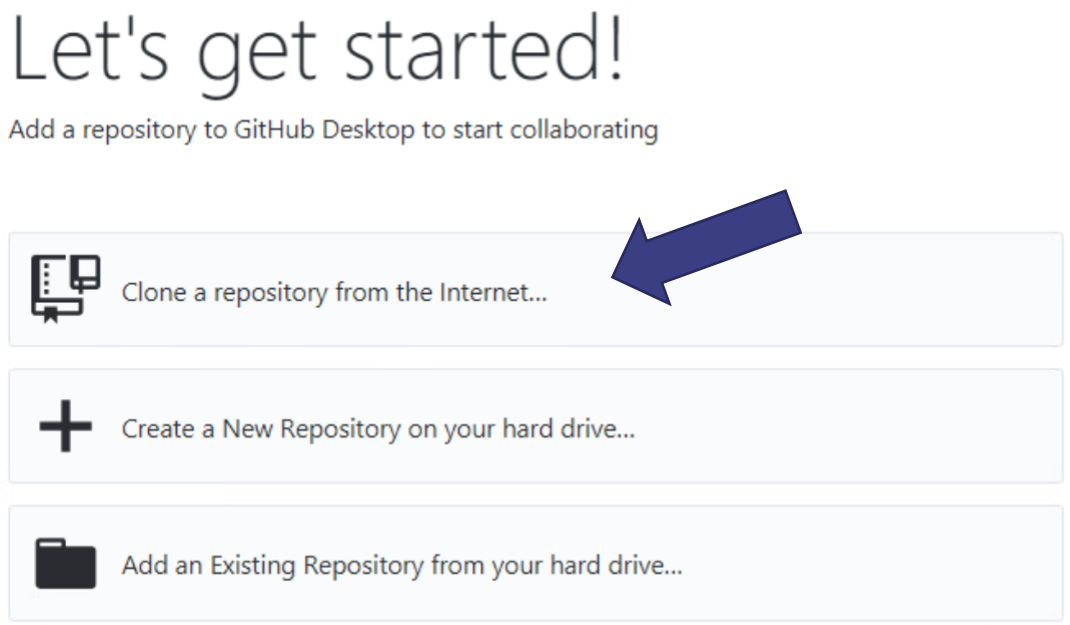
- Starting with Pseudocode
- Variables and Parameters
- Logical Flow
- Storing and Reading Data
- Quality Commenting

Module Setup

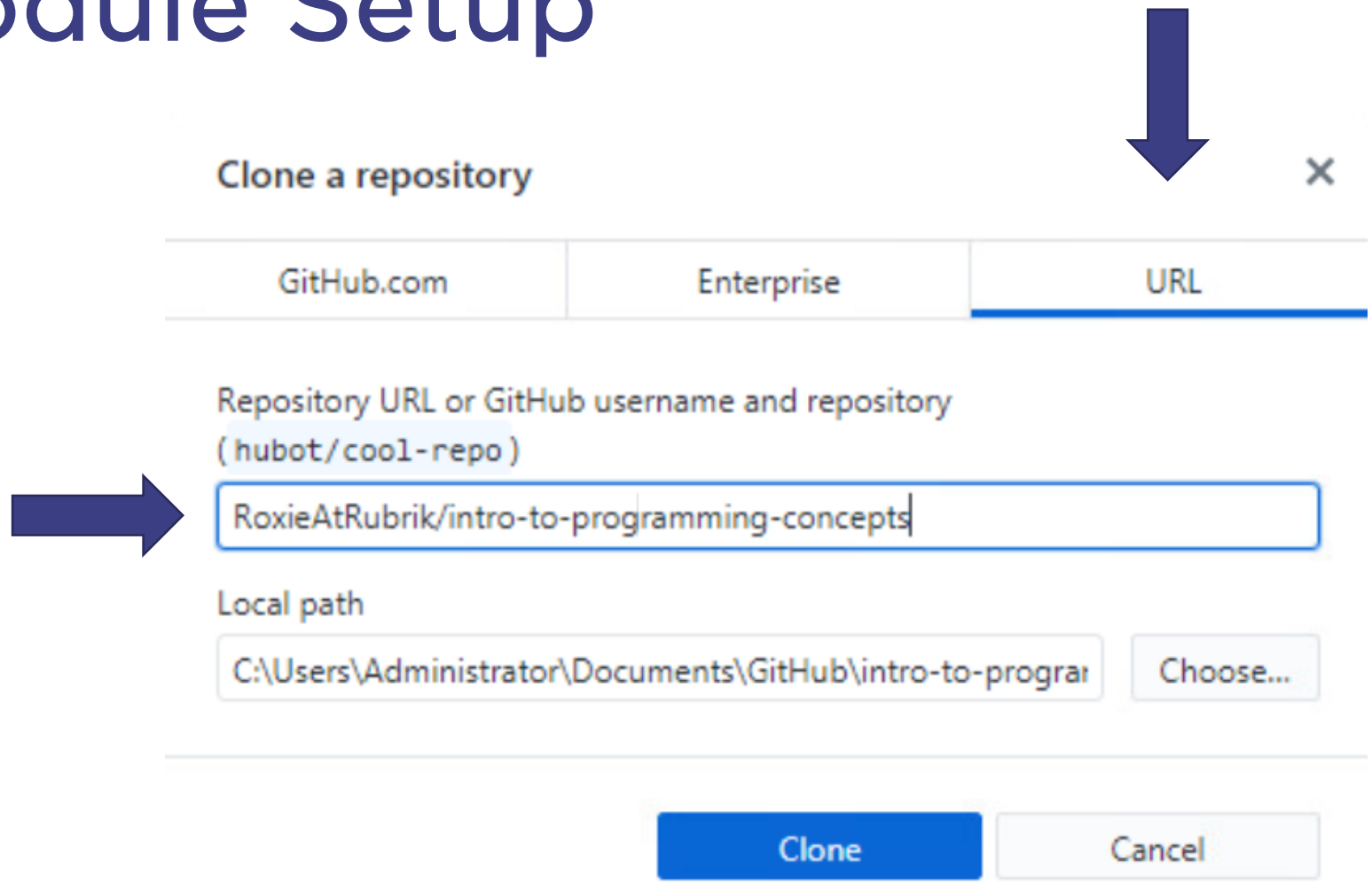
Open the GitHub Desktop application.

Skip any questions asking you to sign in or create an account.

Use the “Clone a repository from the Internet” option.



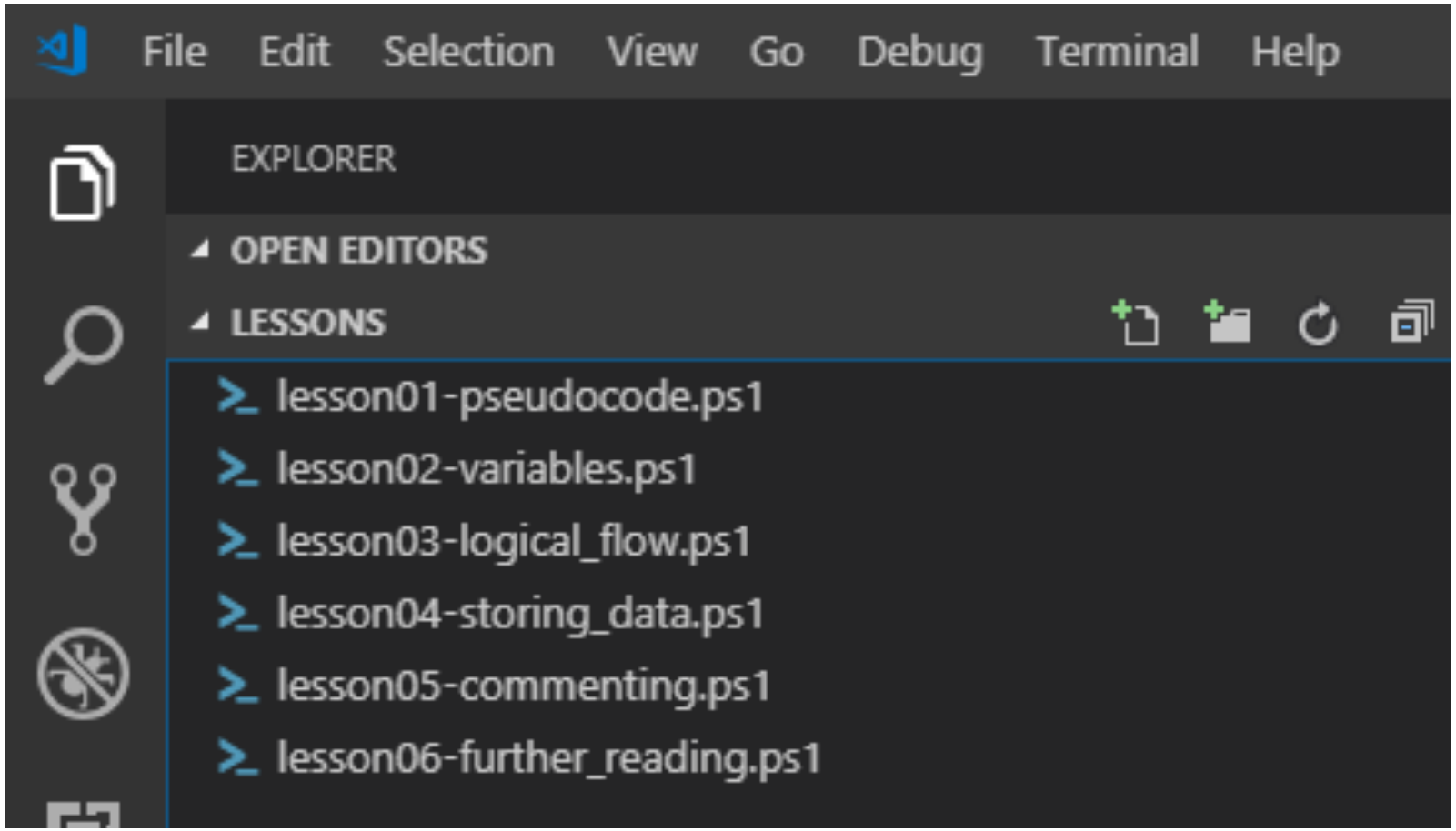
Module Setup



Module Setup

Launch the Visual Studio Code (VSCode) application

1. Click on the **File** Menu
2. Select **Open Folder**
3. Navigate to “intro-to-programming-concepts”
4. Open the “Lessons” folder



Example of the Lessons Folder

Starting with Pseudocode

What is Pseudocode?

A notation resembling a simplified programming language, used in program design.

Why would we use Pseudocode?

- Easy for you and your peers to read and **review** before committing to writing any code.
- Construct the **logical flow** to determine if there are gaps.
- Progress from an **architecture diagram** or flowchart into something that more closely resembles code.

Start with your Goal

- What is the primary goal you wish to achieve with your code?

"Get a list of running processes and sort the results by the process ID"

"Find all virtual machines that have more than two vCPUs"

Things to Do

As you gain more skill and confidence with writing code, your ability to use whitespace, naming conventions, and control structures will improve.

If you are starting off for the first time, don’t worry. Just try your best to write something that sounds logical.

- Keep it simple and concise
- Format the whitespace as if you were writing real code
- Aim to use the correct naming conventions
- Add control structures

Lab Exercise

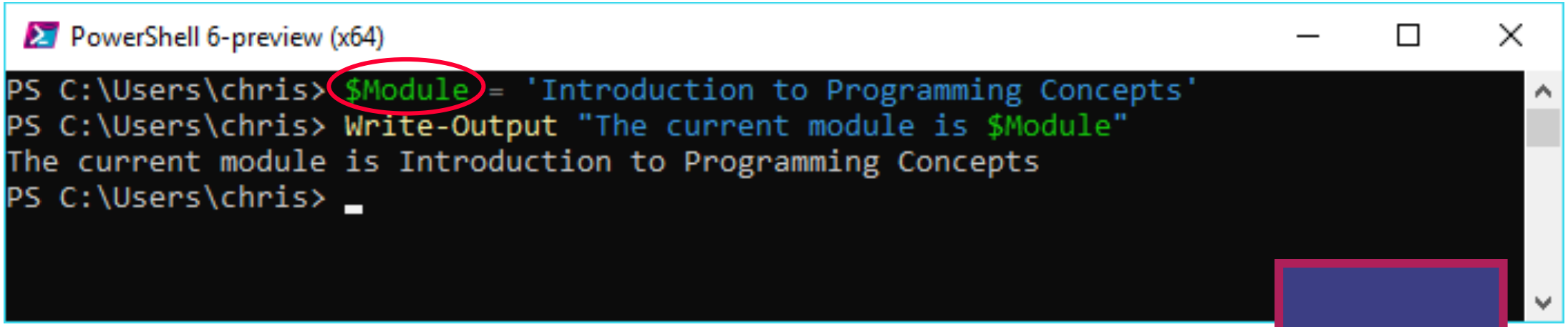
- Return to VSCode
- Use the Explorer to find “lesson01-pseudocode.ps1”
- Complete the goals listed

Variables and Parameters

What are Variables?

A storage location paired with an associated symbolic name, which contains some known or unknown quantity of data referred to as a value.

Example - Variable

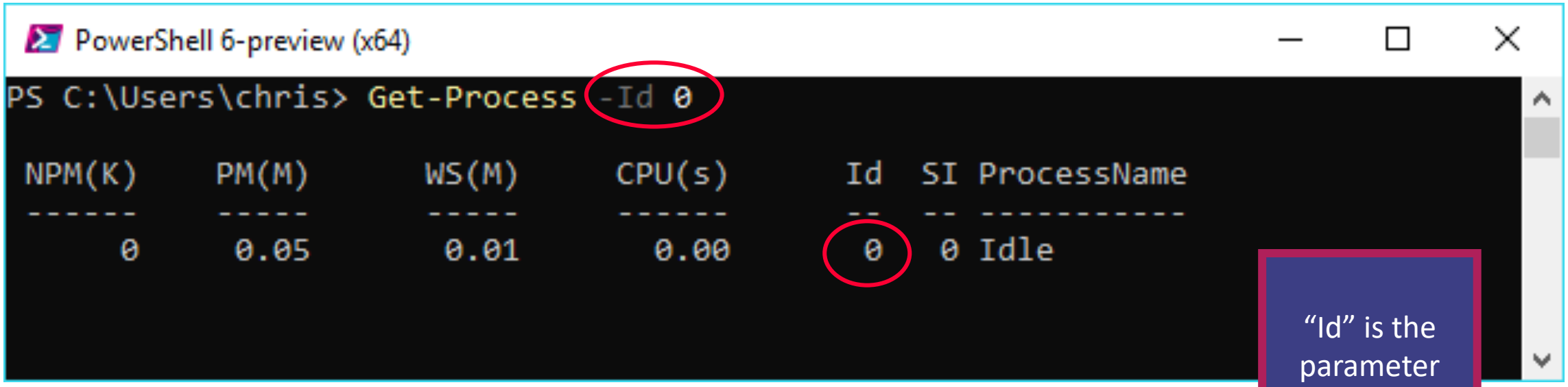


"$Module" is a variable that contains a *string*

What are Parameters?

A parameter is a special kind of variable in computer programming language that is used to pass information between functions or procedures. The actual information passed is called an argument.

Example - Parameter



Why would we use Variables?

- It’s much simpler to declare a variable to **store information** and then refer to the variable.
- Using static values that are input directly into your code is a **bad idea** and takes away the freedom that you typically wish to deliver to your user.
- **Things change!**

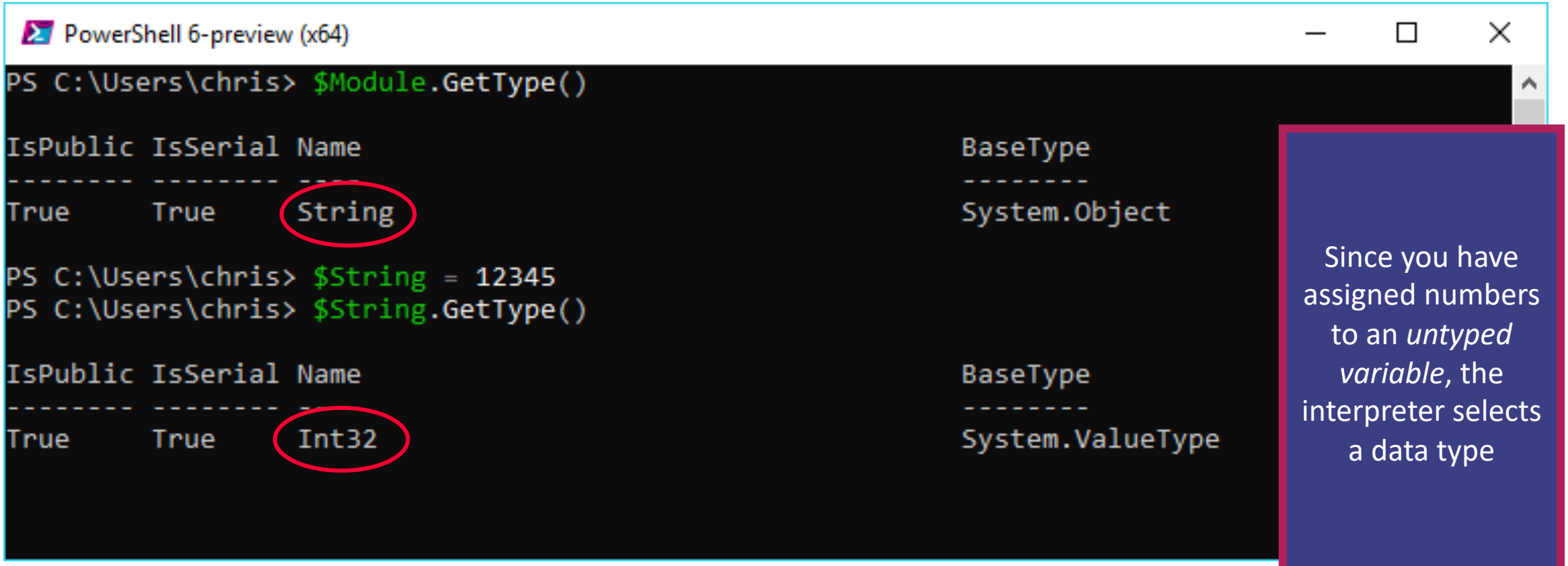
Variable Data Types

Always tell your code which data type to use for a variable.

Otherwise, it will make the variable's data type match the input that it sees.

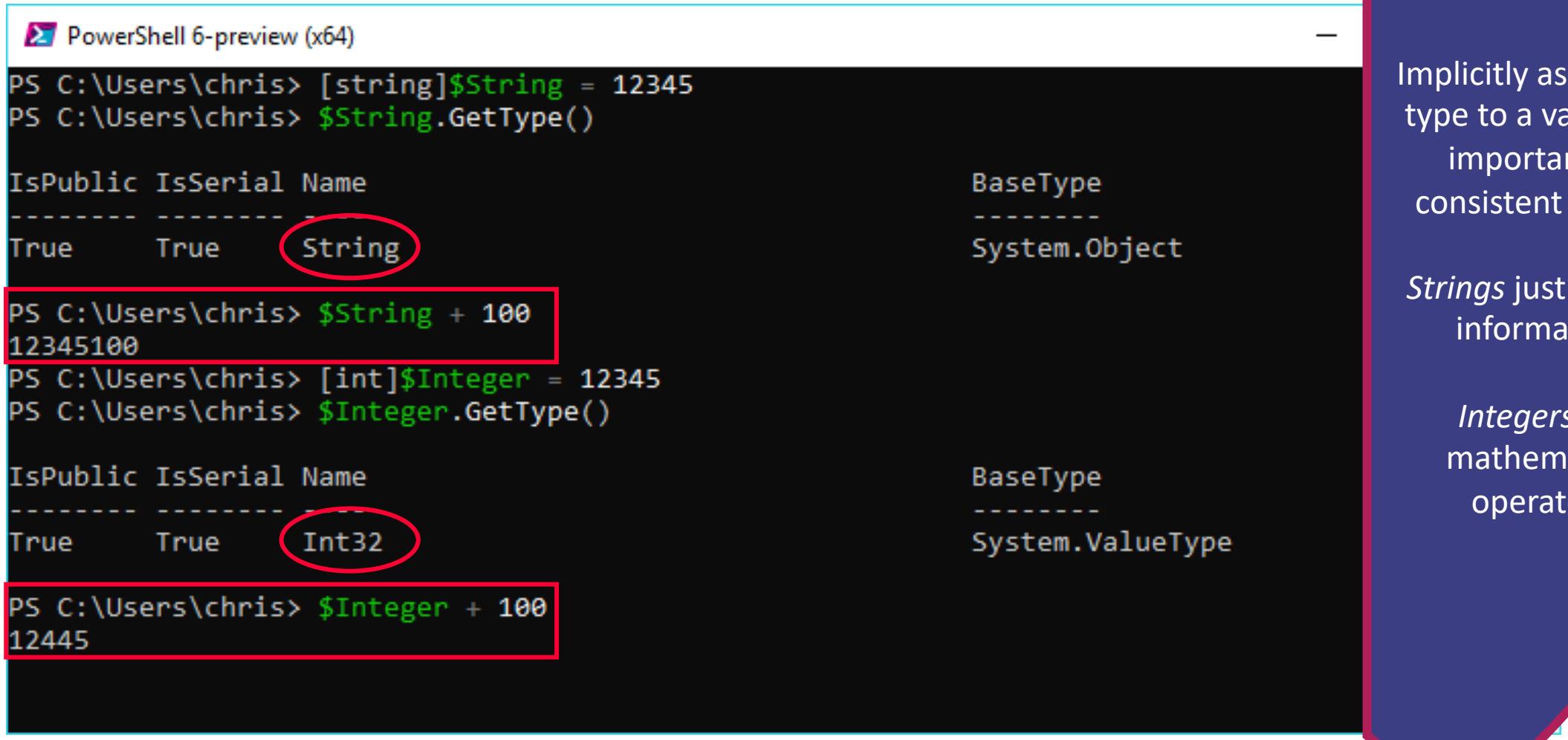
- String
- Int (Integer)
- Bool (Boolean)
- Float (Floating-point)
- Array
- Hashtable
- SecureString

Example – Untyped Variables



Since you have assigned numbers to an *untyped variable*, the interpreter selects a data type

Example – Typed Variables



Implicitly assigning a type to a variable is important for consistent results.

Strings just append information.

Integers use mathematical operators.

Variable Naming Standards

Different projects and languages use different capitalization for variables.

These will be caught when linting your code, or when working on a project that has defined a standard.

- Snake Case
  - my_variable_name
- Camel Case (Pascal Case)
  - MyVariableName
- Lower Camel Case
  - myVariableName

Lab Exercise

- Return to VSCode
- Use the Explorer to find “lesson02-variables.ps1”
- Complete the goals listed

Logical Flow

What is Logical Flow?

The order in which individual statements, instructions or function calls of an imperative program are executed or evaluated.

Why is Logical Flow important?

- You must understand the order in which your code will be interpreted by the server.
- Many of the bugs you will encounter are due to not understanding how logical flow operates.

Choice

Logical decision making points.

- If
- Then
- Else
- ElseIf

```powershell
[Int]$Integer = 12345

If ($Integer.GetType().Name -eq 'Int32') {
    Write-Output "We found an integer!"
}
Else {
    Write-Output "This ain't a number, help!"
}
```

What do you think the result will be?

```powershell
[String]$Integer = 12345

If ($Integer.GetType().Name -eq 'Int32') {
    Write-Output "We found an integer!"
}
Else {
    Write-Output "This ain't a number, help!"
}
```

What do you think the result will be?

Case / Switch

Examine a variable against a list of cases (options) and execute the associated command when the value meets the stated requirements.

- Case
- Switch

Loops

Methods for iterating through a count, condition, or collection.

- Count Controlled
  - Do (something X times)
  - For
  - Next
- Condition Controlled
  - Do (something until X happens)
  - While
  - Until
- Collection Controlled
  - Foreach

Lab Exercise

- Return to VSCode
- Use the Explorer to find “lesson03-logical_flow.ps1”
- Complete the goals listed

Structured Data

What is Structured Data?

Structured data refers to any data that resides in a fixed field within a record or file. This includes data contained in relational databases and spreadsheets.

There are a collection of data serialization languages to define structured data.

Why is Structured Data important?

- It is easy for humans to read and write.
- It is easy for machines to parse and generate.
- The Four Major Activities:
  - You need to **parse the content** from a serialization language.
  - **Read** the required values.
  - **Manipulate** those values.
  - You may need to **store manipulated values** back into a serialization language file.

Serialization Languages

Common languages are everywhere. Different projects and languages are better with one or more serialization languages.

These are the 4 most popular. However, there are a ton more.

https://en.wikipedia.org/wiki/Comparison_of_data_serialization_formats

- CSV
  - Comma-separated Values
- JSON
  - JavaScript Object Notation
- YAML
  - YAML Ain't Markup Language
- XML
  - eXtensible Markup Language

JSON Structure

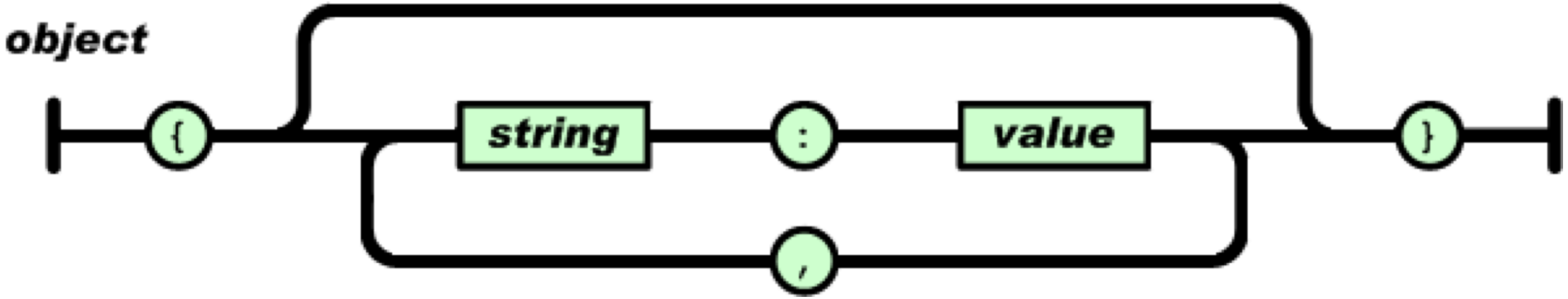
A collection of **name/value** pairs.

- In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.

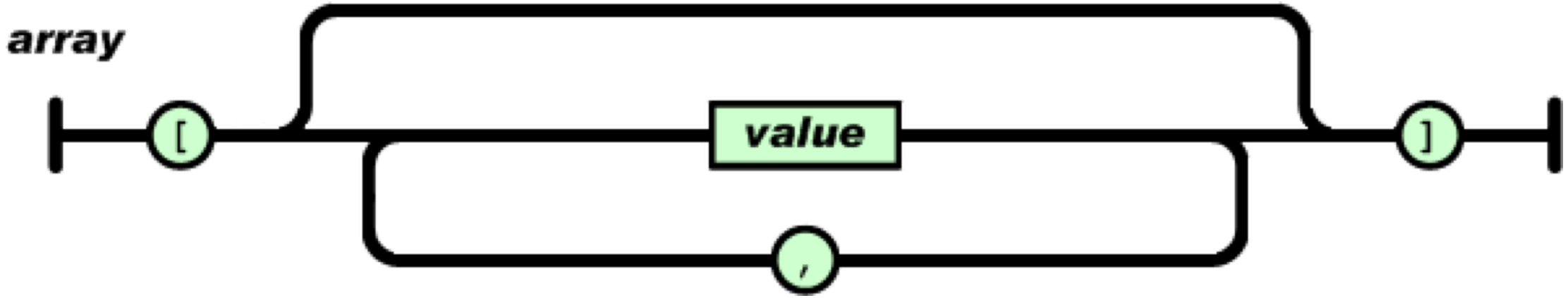
An **ordered list** of values.

- In most languages, this is realized as an array, vector, list, or sequence.

Objects



Arrays



Example Payload

- Key1, Key2, Key3 are first level objects.
- Key4 contains an array of values.
- Key5 is a first level object.
- Key6 contains a collection of objects.
- Key7, Key8, and Key9 are second level objects.

```
{
   "key1":  "value1",
   "key2":  "value2",
   "key3":  "value3",
   "key4":  [
            "value4a",
            "value4b",
            "value4c"
            ],
   "key5":  "value5",
   "key6":  {
            "key7":  "value7",
            "key8":  "value8",
            "key9":  "value9"
            }
}
```

Real World Example

This is from the Slack API documentation for that opens a multi-person direct message or just a 1:1 direct message.

This is the JSON serialized data response that provides information about your conversation request.


```
{
    "ok": true,
    "no_op": true,
    "already_open": true,
    "channel": {
        "id": "D069C7QFK",
        "created": 1460147748,
        "is_im": true,
        "is_org_shared": false,
        "user": "U069C7QF3",
        "last_read": "0000000000.000000",
        "latest": null,
        "unread_count": 0,
        "unread_count_display": 0,
        "is_open": true,
        "priority": 0
    }
}
```


Lab Exercise

- Return to VSCode
- Use the Explorer to find “lesson04-storing_data.ps1”
- Complete the goals listed

Quality Commenting

What is Quality Commenting?

Providing insight and understanding of why you made certain decisions within your code. Multi-line comments are typically used to add descriptive help at the start of a script, but also work to embed comment text within a command.

Why is Quality Commenting important?

- Explain your design approach to your peers, the community, and even your future self.
  - Tip: Be kind to your future self. ☺
- Document any business logic you're adhering against.
- Reference decisions for historical reference and context.
  - Link to a GitHub Issue and/or Pull Request.
  - Link to a Change Ticket or other discussion.

Healthy Practices

- Avoid being overly verbose in your comments.
- It's more about describing why or how something works.
- Here is a bad comment example →

Bad Commenting ☹

```
# I now need to see if the version number is greater than
# 14. To do this, the first thing I do is split the version
# at the dash and store that in a variable called $c
# that will give me an array of two elements. The first
# element contains the word dell and the second the number
# of the version
$c = $b.Version -split "-"
```

Lab Exercise

- Return to VSCode
- Use the Explorer to find “lesson05-commenting.ps1”
- Complete the goals listed

Summary

Module Overview

- Starting with Pseudocode
- Variables and Parameters
- Logical Flow
- Storing and Reading Data
- Quality Commenting

build

Building the Future of Data Management