**Data Basics**

Data is everywhere. It is now considered a fuel to drive businesses efficiently. A huge volume of data is produced daily from different sources like social media platforms, online transactions, and sensor-based devices. This data can be of different varieties. Different organizations store and process data as per their business requirements. Data provides valuable information for better decision-making, improving customers' experience, and organizational operations.

It is very important to understand the fundamental concepts related to data. This lesson will help you learn about the difference between data and information, variables, data types, basic structures used in data analytics, as well as different categories of data.

### kills covered in Lesson:

* **Skill 1.1:** Define the concept of data
* **Skill 1.2:** Describe basic data variable types
* **Skill 1.3:** Describe basic structures used in data analytics
* **Skill 1.4:** Describe data categories

# Data Concepts Introduction

The terms data and information are often used interchangeably, but there is a difference between them. This section will help you understand what data really is and how it is different from information.

This skill covers how to:

* Define data and information
* Differentiate between data and information
* Define statistics and its relation with data

# Data and Information

**Data**is a collection of facts or figures that are recorded for analysis. For example, it can be product names, heights, student marks, registration codes, tax codes, interview recordings, etc.

**Information**is organized data that is analyzed and presented to make it meaningful and suitable for making decisions.

The example in Table 1-1 illustrates the difference between data and information:

**Table 1-1**

|  |  |
| --- | --- |
| **Raw Data** | John, 85, 100 |
| **Information** | John got 85 out of 100 marks. |

In Table 1-1, the given raw data [John, 85, 100] is just one text and two numbers but has no meaning. After providing some context to this raw data, information can be derived from it.

The following table lists differences between data and information:

**Table 1-2**

|  |  |
| --- | --- |
| **Data** | **Information** |
| A collection of facts that is raw and has little or no meaning. | Processed data that has some meaning. |
| Data is independent of information. | Information is dependent on data. |
| Data cannot be used for making decisions unless analyzed. | Information can be used to make personal and business decisions. |
| It may be difficult to understand data. | It is easier to understand information. |

# Data and Statistics

Data is sometimes confused, and therefore wrongly used interchangeably, with statistics. However, there is a clear difference between them.

**Statistics** is the summary of the data collected and can be a numerical value. Though not necessary, statistics are often presented in the form of charts, graphs, or tables.

Table 1-3 shows the marks of ten students on a Math assignment. It is an example of raw data.

**Table 1-3**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Marks for Math assignment** | | | | | | | | | | |
| **roll** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **marks** | 60 | 40 | 80 | 65 | 55 | 70 | 50 | 85 | 45 | 50 |

The statistics that can be derived from the raw data in Table 1-3 are represented in Table 1-4.

**Table 1-4**

|  |  |
| --- | --- |
| **Statistics** | |
| **Average mark** | 60 |
| **Maximum mark** | 85 |
| **Minimum mark** | 40 |

### What is data analysis?

Data analysis is a process of converting raw data into useful information for decision-making by users.

# Variables and Data Types Introduction

In the field of data analytics, a variable is a value, such as a number, that could represent a height measurement, someone's age, or income, as well as a quantity or characteristic that can be counted or measured like class size, class grades, or gender. It is called a variable because its value can change. Variables can have different values during different stages of a program. For example, age is a variable that can vary for different people and can also vary over time.

This skill covers how to:

* Define variables
* Identify different data types
* Define type checking

# Variables

In computer programming, **variables** are the containers for storing data values, which might change during the execution of the program.

There are a few rules you have to follow when naming variables. These rules differ slightly depending on the programming language.

The following rules are common across many programming languages:

* A variable name can only consist of letters (A-Z, a-z), digits (0-9), and an underscore (\_).
* A variable name can start with a letter or an underscore character, but cannot start with a number.
* A variable name is case-sensitive. So the variable names number and Number might be different.
* A variable name should not be a keyword. Keywords are the reserved words in any programming language.

Apart from the rules, there are some best practices that should be followed when naming variables.

* The variable name should be descriptive. This means it should tell what the variable represents. For example, the variables accountNumber, firstName, and testResult clearly represent the value that is being held.
* The variable name should neither be so long that it is easily misspelled during coding nor so short that it isn’t descriptive enough.

These best practices help improve code readability and make code easier to understand.

# Data Types

In computer programming, a **data type** specifies the type of data that a variable can store. Some data types are seen in all programming languages, while other data types are language-specific.

The following are some of the common data types:

**Table 1-5**

|  |  |  |
| --- | --- | --- |
| **Data Types** | **Description** | **Examples** |
| Integer | It represents positive or negative whole numbers without fractions or decimals. | 5, 10, -8 |
| Float | It represents real numbers with floating point representation. | 5.5, -3.5 |
| Character | It is used to store a single character. | ‘A’, ‘X’, ‘1’ |
| String | It is a collection of one or more characters. It is enclosed in double quotes. In some languages, like Python, it can also be enclosed in single or triple quotes. | “John”, “car”, “Hello World” |
| Boolean | It is used to store boolean values i.e., True or False. | True, False |

A variable can hold different types of values. For example, the name of a student must be stored as a string, the points\_earned should be stored as an integer to allow calculation of total or average points, and the variable pass should be stored as a boolean.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Table 1-5**

|  |  |
| --- | --- |
| name = "John" | A string value is assigned to the variable name. |
| points\_earned = 9 | An integer value is assigned to the variable points\_earned. |
| pass = True | A boolean value is assigned to the variable pass. |

# Type Checking

In every programming language, there is a process of checking that values have been assigned their correct data types. This is known as **type checking** or **data type checking**. It is required to minimize errors during the execution of programs.

Type checking occurs at either runtime or compile time.

* **Compile Time** is the time period when the source code in a specific programming language (such as Java, C#, and C++) is converted to machine code or binary code.
* **Runtime** is the time period when the program is actually executing, either using the converted machine code or through a program that interprets script, such as a web browser.

There are two categories of type checking implemented in most programming languages; static and dynamic.

* In **statically-typed languages**, type-checking occurs at compile time. Java, C#, C++, and Scala are some examples of statically-typed programming languages.
* In **dynamically-typed languages**, type-checking occurs at runtime or execution time. Python, JavaScript, Perl, and PHP are some examples of dynamically-typed programming languages.

It is required that you declare the data types of variables before you use them in a statically typed language, but it is not required in a dynamically typed language.

Consider the following two code examples:

**Table 1-6**

|  |  |
| --- | --- |
| **Java example** | **Python example** |
| int n;  n = 5; | n = 5 |

Both examples are doing the same thing i.e., creating a variable called n and assigning it the value 5. But there is a difference. In the first line of the Java example, the variable n is declared as an integer because Java is a statically-typed language, so it expects each variable to be declared as a specific type before assigning a value to it. The data type of a variable does not need to be declared in Python because Python is a dynamically-typed language. The first value assigned to the variable determines the data type of the variable.

Watch this video by Chris Jackson to learn more about variables and data types in Python.

**Data Structures Introduction**

A data structure is a specific means of collecting and organizing data so that it can be effectively accessed and processed.

There are many data structures that can be used, as determined by the requirement and use case. This section will help you learn about some of the most common structures like tables, arrays, and lists.

This skill covers how to:

* Define tables
* Define arrays
* Define lists

**Tables**

A **table** is a data structure in which data is arranged in rows and columns.

* The **row** is also called a record, a tuple, or a vector.
* The **column** is also called a field, a parameter, a property, or an attribute.
* The intersection of a row and a column is called a **cell**.

In database systems, data is often stored as tables in which columns are data fields and rows represent data records. In Microsoft Excel sheets, data is stored in the form of a table having rows and columns.

Table 1-7 is an example of a student table:

* It has a header, three columns, and four rows.
* The first row is called a **header** and it is used to display the column names.
* The three columns are roll\_number, name, and age.
* The four rows hold the actual data records.

**Table 1-7**

|  |  |  |
| --- | --- | --- |
| **roll\_number** | **name** | **age** |
| 1 | John | 20 |
| 2 | San | 25 |
| 3 | Mary | 21 |
| 4 | Lara | 22 |

Watch this video by Eric Johnson to understand tables.

**Arrays**

An **array** is a linear data structure that is used to store a collection of items of the same data type. An array has the following characteristics:

* It is indexed-based. This means that each item has an index value, with the first item of the array having the index 0.
* It is fixed in size. This means the size of the array cannot be increased or decreased during runtime.
* It provides random access. This means you can access any item in the array using its index.

Table 1-8 is an example of an array arr of size 5. The items in this array are 10, 20, 25, 15, and 12, with indices 0, 1, 2, 3, and 4, respectively:

**Table 1-8**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index → | 0 | 1 | 2 | 3 | 4 |
| arr | **10** | **20** | **25** | **15** | **12** |

The array items can be randomly accessed using their indices. For example, arr[0] accesses the first item of the array i.e., 10; arr[1] accesses the second item of the array, i.e., 20, and so on.

The same array can be written in Java and JavaScript as shown in Table 1-9 and Table 1-10 respectively:

**Table 1-9**

|  |  |
| --- | --- |
| **Array in Java** | **Output** |
| public class TestArray{  public static void main(String[] args) {  // Create and initialize array  int[5] arr = {10, 20, 25, 15, 12};    /\* Access and display for output the item at index 3 and index 4 \*/  System.out.println(arr[3]);  System.out.println(arr[4]);  }  } | 15  12 |

In the Java example written in Table 1-9 above, an integer array arr with 5 elements is created and initialized. Then, the items at indices 3 and 4 are accessed and printed; they are 15 and 12 respectively.

**Table 1-10**

|  |  |
| --- | --- |
| **Array in JavaScript** | **Output** |
| // Create and initialize array  let arr = [10, 20, 25, 15, 12];    /\* Access and display for output the item at index 3 and index 4 \*/  console.log(arr[3]);  console.log(arr[4]); | 15  12 |

In the JavaScript example written in Table 1-10 above, an integer array arr with 5 elements is created and initialized. Then, the items at indices 3 and 4 are accessed and printed; they are 15 and 12 respectively.

**Lists**

A **list** is a linear data structure that is used to store a collection of items.

* Lists are supported in many programming languages such as Python, Java, C#, and C++.
* A list is created using square brackets in Python, as seen in Table 1-11.

**Table 1-11**

|  |  |
| --- | --- |
| Python | numList = [10, 20, 25, 15, 12]; |

* The items in the list are ordered, which means any new items added to a list will be placed at the end of the list. Note that many implementations of a list include methods that allow you to insert an item at a different location.
* The items in lists are **mutable**, which means the items in a list can be changed, added, and removed after creating the list.
* Lists allow duplicate items, which means the items in a list can have the same value.
* Similar to an array, list items are indexed where the first item has an index of 0, the second item has an index of 1, and so on.
* Unlike an array, the items in a list can be of the same or different data types, as shown in Table 1-12.

**Table 1-12**

|  |  |
| --- | --- |
| Python | numList = ["John", 20, True, 15, "London"]; |

Watch this video by Brian Overland to learn how to build and manage lists in Python.

**Data Categories Introduction**

There are different categories of data based on their nature, complexity, and use case. It is very important to understand the difference between the categories in order to use data efficiently. For each category of data, you will find tools and technologies designed to help you derive valuable information.

This skill covers how to:

* Differentiate between structured and unstructured data
* Identify and use different types of data
  + Big Data
  + Qualitative Data
  + Quantitative Data
  + Imputed Data
  + Metadata

**Structured and Unstructured Data**

Data can be of different varieties and complexity. Some data, like student records and transaction details, can be structured. Data can also be unstructured, examples include audio, emails, and social media posts. **Structured data** can be easily stored, processed, and analyzed by traditional database systems, whereas **unstructured data** is challenging to store and manage.

**Structured Data**

Structured data is organized in a predefined format that makes it easy to search and analyze. Some typical examples of structured data are spreadsheets and relational database tables. The data type for each field is often also predefined. Relational database tables are usually stored in a relational database management system (RDBMS) table with a fixed schema. **Schema** refers to the logical structure of data. A schema does not contain actual data, but only information about the data. For example, the schema of a table in a database contains information like the table name, field names, data types of the fields, etc. Some examples of structured data are customer records, student records, product information, and sales transactions.

Table 1-13 shows an example of structured data that is stored in a table having rows and columns:

**Table 1-13**

|  |  |  |
| --- | --- | --- |
| **roll\_number** | **name** | **age** |
| 1 | John | 20 |
| 2 | San | 25 |
| 3 | Mary | 21 |
| 4 | Lara | 22 |

This is a student table with the fields roll\_number, name, and age. Each column has a predefined data type. The fields roll\_number and age are numeric, and name is a string.

**Unstructured Data**

Unstructured data is unorganized. It does not have any predefined format. It is very hard to analyze unstructured data programmatically. The data types for each piece of data are also not predefined. It cannot be stored in a relational database management system (RDBMS). It is usually stored in non-relational databases such as MongoDB, Cassandra, Couchbase, etc. Some examples of unstructured data are social media posts, images, audio, video, and emails.

The key differences between structured and unstructured data are summarized in Table 1-14 below:

**Table 1-14**

|  |  |  |
| --- | --- | --- |
|  | **Structured Data** | **Unstructured Data** |
| **Data definition** | It is organized and usually has a predefined format. | It is unorganized and has no predefined format. |
| **Data Type** | The data type for each attribute is predefined. | The data type for each piece of data is not predefined. |
| **Storage** | It is stored in relational databases like Oracle, Microsoft SQL Server, or MySQL. | It is stored in non-relational databases such as MongoDB, Cassandra, or Couchbase. |
| **Searching or Analysis** | It is easy to search and analyze. | It is hard to search and analyze. |
| **Examples** | Customer records, student records, product information, and sales transactions | Social media posts, images, audio, video, and emails |

Watch this video by Jonathan Dinu to learn more about structured and unstructured data.

**Types of Data**

A huge volume and different varieties of data are produced every day from different sources like bank transactions, social media, and IoT devices. This data can be categorized into many different types depending on its variety, size, nature, complexity, and use cases.

In this section, the following different types of data are explained:

* Big data
* Qualitative data
* Quantitative data
* Imputed data
* Metadata

**Big Data**

**Big data** refers to a collection of data having a huge volume and many varieties, and it grows exponentially with time. It is so large in size and complexity that it becomes difficult for traditional data management systems like RDBMSs to store and process efficiently. But these huge volumes of data can be analyzed to get insights that help companies address many business problems and make better business decisions.

Big data has three basic characteristics - volume, variety, and velocity, usually referred to as 3Vs because they all begin with the letter V. But there have been additional Vs added over time to define the characteristics of big data and more are being discovered.

The following are the 6Vs commonly used to describe big data:

1. **Volume** refers to the huge volume of data that is difficult to store and manage using traditional data systems. Terabytes of data are generated every day from social media applications and sensor-enabled devices.
2. **Variety** refers to the different varieties of data. The data can be structured, unstructured, or semi-structured data. It is easy to store and process structured data in traditional data management systems like RDBMSs. But it is very difficult to store and process the unstructured or semi-structured data in them.
3. **Velocity** refers to the speed at which big data is generated. Companies want to manage and analyze new data quickly to get the required information.
4. **Veracity** refers to the degree to which big data can be trusted. It simply means the data should be good in quality. The data should be clean and accurate and should not have missing values.
5. **Variability** refers to data that is extremely variable and keeps on changing always.
6. **Value** refers to the business value of big data. There is no advantage of storing huge volumes and different varieties of data if some business value cannot be derived from it.

With the rise of big data, many tools and frameworks have been created to store and process big data.

The following are examples of tools and frameworks for storing and processing big data:

* **Apache Hadoop**: Apache Hadoop is an open-source framework used to store and analyze big data. It has two main components: HDFS (Hadoop Distributed File System) for storage and MapReduce for processing.
* **NoSQL databases**: NoSQL stands for “Not only SQL”. NoSQL databases are non-tabular databases that have flexible schemas and provide a different mechanism for the storage and retrieval of data than RDBMS tables.
* **Apache Spark**: Apache Spark is an open-source unified analytics engine for processing large-scale data.

Watch this video by Dursun Delen to learn about big data and its sources.

**Qualitative Data**

**Qualitative data** refers to data that is non-numeric and cannot be quantified. In other words, it is descriptive data that cannot be expressed numerically. It is also called categorical data. Some examples of qualitative data are gender, qualification, religion, and letter grade on an assignment. It can be either nominal or ordinal.

**Nominal qualitative data** is a type of qualitative data where each data point represents a distinct category or label that cannot be quantified or ordered.  
  
Some examples of nominal data are:

* Gender (Male, Female),
* Colors (Red, Green, Blue, etc.),
* Languages (English, French, German, Hindi, etc.)

There is no order relation between the values of nominal data. For example, It is not possible to state which color is better or has a higher rank than the others. We cannot say that the color “Green” is better than the color “Blue”.

**Ordinal qualitative data** is a type of qualitative data that involves assigning variables to observations, indicating the relative position or rank.  
  
Some examples of ordinal data are:

* Size of clothes (i.e., S, M, L, XL, XXL)
* Grade on an assignment (i.e., A, B, C, D)
* Behavior of students (i.e., bad, good, very good, excellent)

There is an order relation between the values of ordinal data. For example, the clothing size ‘S’ is smaller than ‘M’, which is again smaller than ‘L’.

**Quantitative Data**

**Quantitative data** refers to numeric data that can be quantified. It is also called numerical data. It can be either continuous or discrete.

**Continuous data** is a type of quantitative data that can have an infinite number of values within a specified range.

* Time, age, and height are some examples of continuous data.

**Discrete data** is a type of quantitative data that can have a countable number of values within a specified range.

* The number of shops in a city, the number of players on a team, and the number of students in a school are some examples of discrete data. These are measured only as whole units (i.e., 1, 2, 3) and not in fractions.

In simple terms, continuous data is measured, and discrete data is counted.

**Imputed Data**

In the real world, there may be some datasets that contain missing values. The missing values are generally represented as NaN, NULL, N/A, blanks, etc. Missing values are one of the most common problems in data analysis.

There are various ways to handle missing value:

* One way is to discard the records that have missing values. But doing so may result in the loss of valuable information.
* A better way is to replace missing data with some substituted value. This technique is known as **imputation**.

The substituted value that is used to replace missing data is known as **imputed data**. Imputed data is derived from the existing part of the data.

The following are some of the popular methods of data imputation:

* **Imputation Using Mean or Median Values**: In this technique, the missing values in a column are replaced by the mean or median value of non-missing values in the same column. This method can be used only with numeric data.
  + **Mean** or **average** is the sum of all the numbers divided by the total number of numbers. For example, the mean of 5 numbers [9, 12, 8, 14, 7] is (9+12+8+14+7)/5, i.e., 10.
  + **Median** is the middle number in the sorted list of numbers in ascending or descending order. For example, to find the median of 5 numbers [9, 12, 8, 14, 7], first sort the numbers in ascending order [7, 8, 9, 12, 14] and find the middle number, i.e., 9. So 9 is the median.
* **Imputation Using Most Frequent Values**: In this technique, the missing values in a column are replaced by the most frequent value of non-missing values in the same column. This method can be used for both numeric and non-numeric data.
* **Imputation Using Zero or Constant Values**: In this technique, the missing values of a column are replaced by zero or any other constant value. This method can be used for both numeric and non-numeric data.

**Metadata**

**Metadata** is used to describe and give information about data that is apart from the content of the data, but is often useful for understanding the data.

Here are some examples of metadata:

* The library catalog holds metadata information like author, title, subject, and the publication date of a book.
* The table of contents and the index in a book are also considered metadata for that book.
* The schema of any table in a database system holds the metadata information like the name of the table, column names, their data types, etc.
* Every database contains a data dictionary that holds metadata information about tables, columns, data types, constraints, table relationships, views, and indexes.

Looking at the metadata examples, we can say that metadata is used everywhere. There are many advantages to using metadata, including:

* It helps organize the data better.
* It helps optimize the use of datasets.
* It gives better insights that help improve the products or processes.

**Apply Your Knowledge**

Consider an example of a system that stores data about you. It might be your school, your bank, a social media network you use, or your doctor's office. Make a list of the data stored there and categorize it as qualitative, quantitative, or metadata. Can you think of situations where data from this source might need to be imputed? If so, which method or methods could you use?

**Summary**

* Data is a raw collection of facts with little or no meaning.
* Information is processed data that has some meaning.
* In computer programming, variables are the containers for storing data values.
* Variables can be of different data types like integer, float, string, character, boolean.
* In a statically-typed language, data type checking is done at compile time.
* In a dynamically-typed language, data type checking is done at runtime.
* A data structure is a way of storing and organizing data in a system in such a way that it can be effectively accessed and processed.
* Tables, lists, and arrays are some of the data structures used in data analytics.
* Data can be structured (i.e., employee records, transaction records, etc.), as well as unstructured (i.e., audio, social media posts, etc.).
* Big data refers to a collection of data having huge volumes and many varieties. It grows exponentially with time.
* Qualitative data are non-numeric and quantitative data are numeric.
* Data imputation is the process of replacing the missing data in a dataset with some substituted value that is derived from the existing part of the data.
* Metadata is used to describe and give information about the structure and other characteristics of data.