

Topics covered:

- Data representation in Python
- Concepts of Identifiers, Variables and Literals
- Rules for naming identifiers in Python programming
- Keywords in Python

DATA REPRESENTATION IN PYTHON

Question: What is Data?

Answer: Data is the **collection of raw facts and figures** that can be stored and processed by a computer. **These facts have no inherent meaning until they are processed** or interpreted. Example:

- Text (e.g., "hello")
- Numbers (e.g., 42)
- Booleans (e.g., True, False)

Once processed, data can turn into **Information** — something meaningful and useful.

Question: What is Information?

Answer: Information is **data that has been processed, organized, or structured to have meaning** or relevance.

- While data is raw and unprocessed
- Information is data that helps in **decision-making** or **understanding**

Example:

- **Data:** "John", 50000, Software Engineer
- **Information:** "John is a Software Engineer with a salary of \$50,000."

Types of Data:

1. By **Structure** (Structured data, Semi-structured data, Unstructured data)
2. By **Nature** (Qualitative data and Quantitative data)
3. By **Measurement Scale** (Statistical Data Types: Nominal, Ordinal, Interval, Ratio)



1. By Structure

Type	Description	Examples
Structured Data	Data organized in a fixed format (rows & columns). Easy to search and analyze.	Relational databases, Excel sheets
Unstructured Data	Data with no predefined format or structure. Harder to process.	Text, images, videos, emails
Semi-Structured Data	Has some organizational properties but not fully structured.	JSON, XML, HTML, NoSQL databases

Types of Data (by Structure)



2. By Nature

Type	Description	Examples
Qualitative Data	Descriptive, non-numeric data. Represents qualities.	Color, gender, feedback
Quantitative Data	Numerical data. Can be measured and analyzed mathematically.	Height, salary, temperature

Types of Data (by Nature)

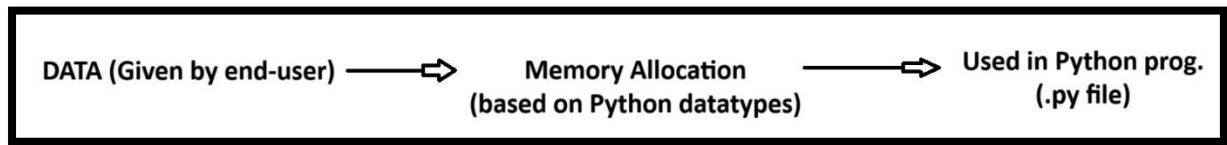


3. By Measurement Scale (Statistical Data Types)

Type	Description	Examples
Nominal	Categories with no order	Gender, country, color
Ordinal	Categories with a meaningful order, but no fixed difference	Rankings, satisfaction levels
Interval	Ordered, measurable differences, but no true zero	Temperature in °C, dates
Ratio	Like interval, but has a true zero	Age, weight, salary, distance

Types of Data (by Measurement Scale)

CONCEPT OF IDENTIFIERS, VARIABLES AND LITERALS



Data Flow in a Python prog.

Data Flow in Python – step by step:

1. Data is provided by the end user

- Through input devices, files, APIs, etc

2. Memory space is automatically allocated

- Python uses data types (like int, str, float, etc.) to determine how much memory to allocate
- Example: age = 25 → Python allocates memory to store an integer

3. Each memory space is given a unique name (identifier)

- These names (like name, age) are **identifiers**

4. Why give names to memory spaces?

- **To easily identify and access the values stored in memory**
- Makes the program readable, maintainable, and manageable
- Without names, we'd have to refer to raw memory addresses (which is impractical)

Identifiers: Identifiers are the **names given to memory locations** (variables, functions, classes, etc.) in a Python program to identify and access the values stored in those locations.

Variable: A **variable** is **a specific kind of identifier that stores data and can change during program execution.**

NOTE: An **identifier** is a general term for *any name* you give in a Python program (like variable names, function names, class names, etc.)

Literals: Literals are **constant values** written directly in the code and assigned to variables.

```
a = "Hello World"
# "a" is a variable (also called as an Identifier)
# "Hello World" is a Literal
```

Types of Literals in Python:

1. Integer Literal
2. Float Literal
3. Boolean Literal
4. Complex Literal
5. String Literal
6. Bytes Literal
7. Bytearray Literal
8. Range Literal
9. NoneType Literal
10. Collection Literals – List, Tuple, Set, Frozenset, Dictionary

RULES FOR NAMING IDENTIFIERS IN PYTHON PROGRAMMING

When **naming variables (or any identifiers like functions, classes, etc.)** in Python, **follow these rules:**

1. **Allowed Characters:** Identifiers can include:

- Alphabets: A–Z, a–z
- Digits: 0–9
- Underscore: _

2. **Cannot Start with a Digit**

- The first character must be a letter or underscore; it cannot be a digit.

3. **Special Symbols Not Allowed (Except _)**

- Identifiers cannot contain special characters like @, #, \$, %, !, etc.

4. **Cannot Use Keywords**

- You cannot use Python keywords (e.g., if, for, class, def, etc.) as variable names.
- Keywords are reserved words with special meaning to the language compilers.


NOTE:

- Python is **case-sensitive**. **Name**, **name**, and **NAME** are all different identifiers
- If you do not follow these rules, Python will raise a **SyntaxError** at runtime.
- Use descriptive names: **score**, **student_name**, **total_price** (best practice)

KEYWORDS IN PYTHON

- Python has a total of **37 keywords**.
- The last two keywords **match** and **case** — were introduced in Python 3.10

List of Python Keywords

False	None	True	and	as
assert	async	await	break	class
continue	def	del	elif	else
except	finally	for	from	global
if	import	in	is	lambda
nonlocal	not	or	pass	raise
return	try	while	with	yield
match	case			

Python Keywords