GS Geetu Sodhi

Python Programming

This course provides a comprehensive introduction to programming using Python, focusing on its fundamental concepts and practical applications. By the end of the course, learners will understand Python's syntax, effectively use variables and data types, and write clear and efficient code. Participants will also gain skills in debugging, handling errors, and performing type conversions to manage data seamlessly. Designed for professionals seeking to enhance their programming skills or beginners starting their coding journey, the course emphasizes hands-on practice and real-world examples. Accessible and easy to follow, it ensures learners build a strong foundation in Python programming.

- **Understanding Python Tokens**
- **Data Types and Variables**
- **Assignments, Operators, Indentation and Expressions**
- **Working with Escape Sequences**
- **Comments**

Input and Output in Python

Quiz

Lesson 1 of 7

Understanding Python Tokens

GS Geetu Sodhi

Python Tokens – The Building Blocks of Python Code

What are Tokens?

In Python, **tokens** are the smallest units of a program that have meaning to the Python interpreter. Think of tokens like words in a sentence — they are the basic building blocks the computer reads to understand what you're telling it to do.

🧩 Types of Python Tokens:

Token Type	Description	Examples
1. Keywords	Reserved words that have special meaning in Python	`if`, `else`, `while`, `def`, `for`, `import`
2. Identifiers	Names used for variables, functions, classes, etc.	`name`, `total_score`, `MyClass`
3. Literals	Values or data in code	`"hello"`, `123`, `True`, `3.14`
4. Operators	Symbols that perform operations	`+`, `-`, `*`, `==`, `and`, `or`
5. Delimiters	Symbols that separate code parts	`()`, `{}`, `[]`, `:`, `,`, `.`
6. Comments	Notes for the programmer (ignored by Python)	`# This is a comment`

Python Tokens

Example

Python Code	Tokens
if score > 50: print("You passed!")	• 'if' → Keyword • 'score' → Identifier • '>' → Operator • '50' → Literal • ':' → Delimiter • 'print' → Identifier (a function) • '"You passed!"' → Literal (string)

Quiz Time

Which of the following is an example of a Python keyword?

◯ score

- ○ 50
- ○ >
- ○ if

SUBMIT

"Master the art of Python by understanding its smallest building blocks—tokens. Every keyword, operator, and literal is a step closer to writing flawless code."

Lesson 2 of 7

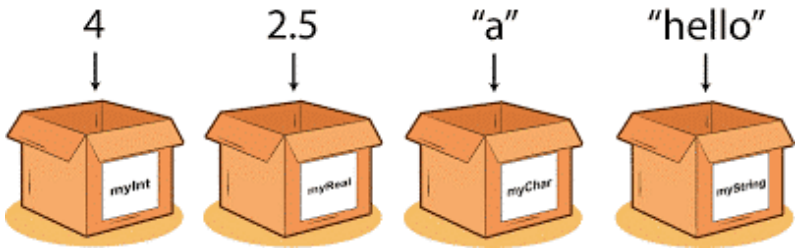
Data Types and Variables

GS Geetu Sodhi

Python Basics: Data Types & Variables

What are Variables?

- A **variable** is like a container that stores a value.
- You **give it a name** and **assign a value** using the = sign



Variables

Example

- name = "Alice"

- age = 13
- is_student = True

Rules for Naming Variables in Python

1. **No spaces allowed**

   Variable names can't contain spaces. Use an underscore *to separate words.*

   ✅ *Example: first*name

   ❌ Wrong: first name

2. **Don't use Python keywords**

   Keywords are special words that Python uses for its own commands, like if, while, or True. You can't use these as variable names.

   ✅ Okay: total_score

   ❌ Not okay: if, class, return

3. **Must not start with a number**

   A variable name can't begin with a number.

   ✅ Good: age1

   ❌ Bad: 1age

4. **Only _ is allowed as a special character**

   You can only use underscores in variable names.

   ❌ No: @, !, -, #

5. **Use meaningful names**

   Choose names that clearly describe what the variable stores.

   ✅ Good: student_grade

   ❌ Not helpful: x, thing, data123

6. **Variable names are case-sensitive**

   Python treats uppercase and lowercase letters as different.

   ✅ score ≠ Score ≠ SCORE

Review Time

Variable names can start with numbers.

False. Variable names must begin with a letter or an underscore, not a number.

Python keywords can be used as variable names.

False. Reserved keywords like 'if' or 'while' cannot be used as variable names.

Variable names are case-sensitive in Python.

True. 'Variable' and 'variable' are treated as different identifiers.

Spaces are allowed in variable names.

False. Variable names cannot contain spaces; use underscores (_) instead.

Underscores are the only special characters allowed in variable names.

True. Other special characters like @, $, or % are not permitted.

Descriptive variable names improve code readability.

True. Using meaningful names makes the code easier to understand and maintain.

CONTINUE

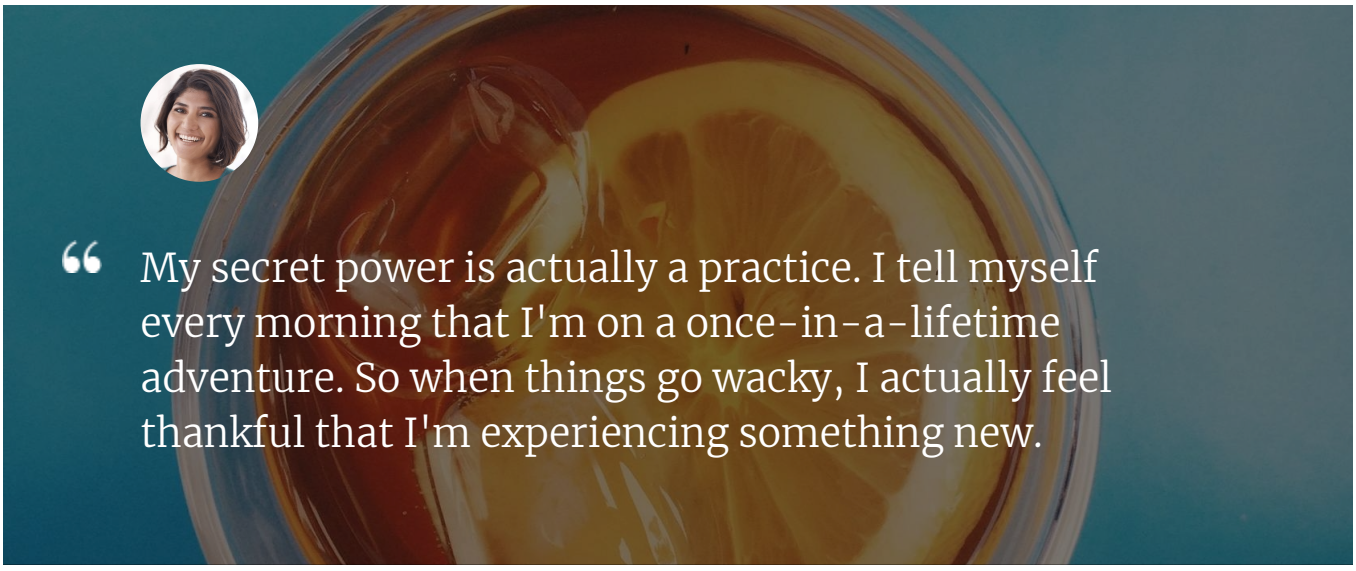
Data Types

Understanding data types is the foundation of programming. They define the kind of values a variable can hold, from numbers and text to complex structures. Mastering them unlocks the true potential of Python's versatility.

Understanding Data Types

Data types are fundamental to programming as they define the kind of data a variable can hold. In Python, data types are categorized into several built-in types, such as text type (str), numeric types (int, float, complex), boolean type (bool), and the None type, which signifies the absence of a value. Each data type serves a specific purpose, enabling developers to perform operations suited to the type of data being handled.

For instance, numeric types like integers and floats are used for mathematical calculations, while strings are ideal for handling textual data. Understanding the characteristics and applications of each data type is crucial for writing efficient and error-free code. Python also allows type conversion, enabling developers to explicitly or implicitly change a variable's data type to suit specific requirements.



Different types of Data Types

1. **Text Type: String (str)**: Strings are sequences of characters enclosed in quotes. They can be stored in either double quotes, e.g., name = "Python", or single quotes, e.g., name = 'Python'.

2. **Numeric Types: int, float, complex**: Numeric types include three subcategories:

   - **Integer (int):** Whole numbers, e.g., age = 25.
   - **Float:** Decimal numbers, e.g., price = 19.99.
   - **Complex:** Numbers with a real and imaginary part, e.g., z = 3 + 4j.

3. **Boolean Type: True or False**: Booleans represent truth values. For example, is_active = True or is_admin = False.

4. **None Type: Absence of Value**: The None type signifies no value or a null value. For example, result = None indicates no result is assigned.

Type Conversion (Type casting)

What is Type Conversion?

Type conversion means **changing the data type** of a value — for example, turning a number into a string or a float into an integer.

Two Types of Type Conversion

1. Implicit Conversion (done by Python automatically)

Python **automatically converts** one type to another when needed — usually from a smaller type to a larger type (like int to float).

```python
num = 5         # int
result = num + 2.5   # float

print(result)       # 7.5 (float)
print(type(result)) # <class 'float'>
```

python code

2. Explicit Conversion (you do it manually)

Also called **type casting** — you decide what type to convert a value to using built-in functions:

Function	Converts to...
int()	Integer
float()	Floating point number
str()	String
bool()	Boolean

.

python Copy Edit

```python
# String to Integer
num_str = "100"
num_int = int(num_str)
print(num_int + 50)  # 150

# Float to Integer
price = 9.99
price_whole = int(price)
print(price_whole)   # 9

# Integer to String
age = 13
text = "You are " + str(age) + " years old."
print(text)

# Any value to Boolean
print(bool(0))       # False
print(bool("Hello")) # True
```

Python is Dynamically Typed

- You don’t need to declare data types — Python detects it automatically.

Changing Values

- Variables can be **updated** anytime:

x = 5 x = x + 1 # Now x is 6

Checking the Type

- Use type() to check the data type of a variable:

score=7.5

print(type(score)) # <class 'float'>

Quiz Time

1000

5

float

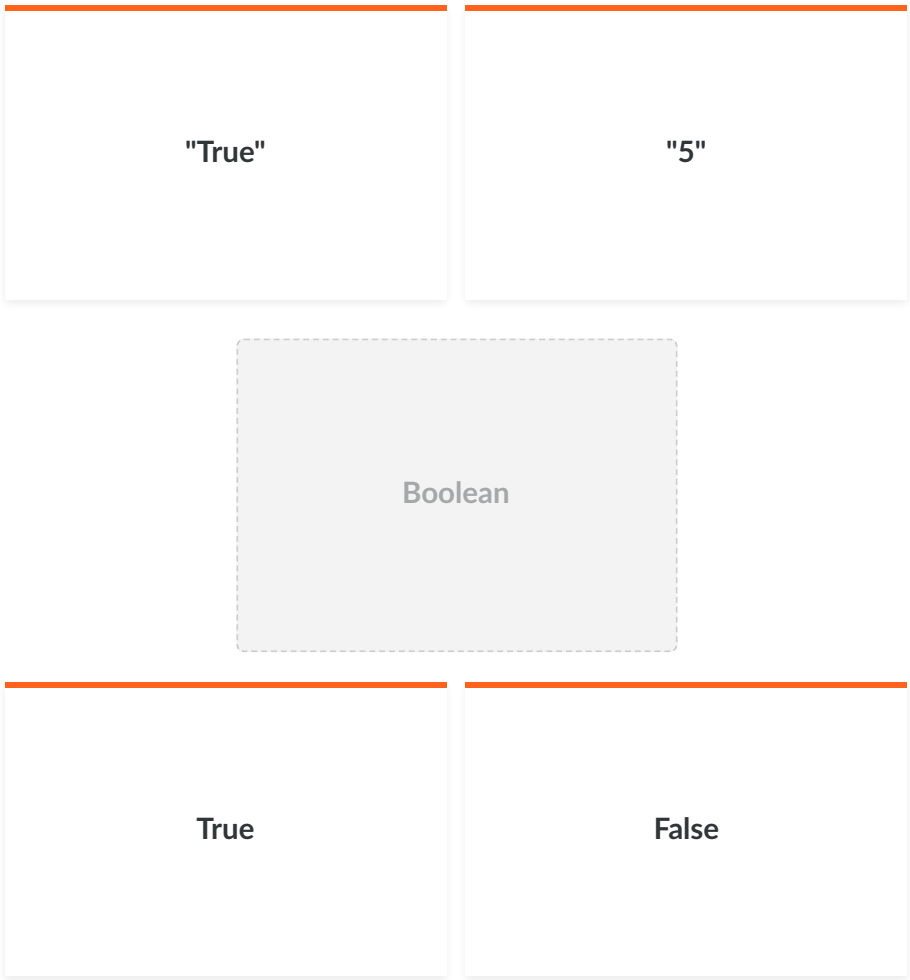
566.7

7.5

String

"Hello"

'H'



Which of the following is a correct way to declare and assign a variable in Python?

- x = 10
- int x = 10
- let x = 10

- ○ var x = 10

SUBMIT

Which of the following statements about Python data types are correct? Select all that apply.

- ☐ Booleans can only represent numeric values.
- ☐ Integers and floats are both numeric types in Python.
- ☐ Python requires explicit declaration of data types for variables.
- ☐ The None type signifies the absence of a value.
- ☐ Strings are sequences of characters enclosed in quotes.

SUBMIT

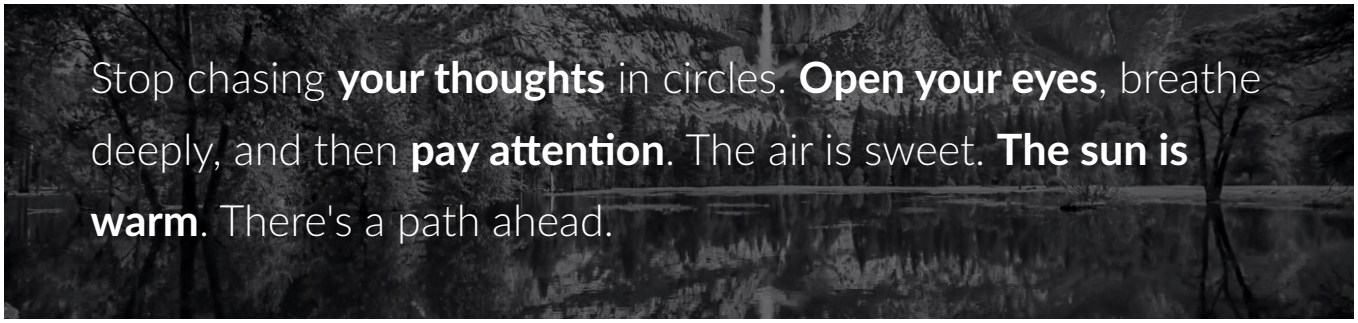
Match each variable name rule in Python with an example that demonstrates it. This will test your understanding of how to correctly name variables in Python.

SUBMIT

Which of the following is an example of explicit type conversion in Python?

- ◯ Declaring a variable without specifying its type
- ◯ Python automatically converting an integer to a float in a calculation
- ◯ Using the type() function to check the type of a variable
- ◯ Using int() to convert a float to an integer

SUBMIT



Lesson 3 of 7

Assignments, Operators, Indentation and Expressions

GS Geetu Sodhi

Some more learning

Programming in Python comes with its own set of amusing quirks. For example, attempting to assign a value backwards, like 87 = scores, will instantly result in a syntax error. It's Python's way of reminding you to follow the rules! Similarly, forgetting proper indentation can lead to unexpected surprises, proving that Python has a sense of humor about structure. Embrace these moments—they make coding both challenging and fun.

Variable Assignment in Python

A variable is a **named reference** to a value stored in memory.

Basic Assignment

```
x = 10
```

```
name = "Geet"
```

- = is the assignment operator.
- The **right side is evaluated first**, then assigned to the left.

Multiple Assignment

Multiple Assignment: Assigning multiple variables at once:

a, b, c = 1, 2, 3

Chained Assignment: Assigning the same value:

x = y = z = 0

Swapping values without a temp variable:

a, b = b, a

Expressions and Operators

➤ Arithmetic Operators

Operator	Description	Example	Result

+	Addition	2 + 3	5
-	Subtraction	5 - 2	3
*	Multiplication	2 * 4	8
/	Division (float)	5 / 2	2.5
//	Floor Division	5 // 2	2
%	Modulus	5 % 2	1
**	Exponent	2 ** 3	8

➤ Comparison / Relational Operators

Operator	Meaning	Example	Result
==	Equal to	3 == 3	True
!=	Not equal to	3 != 4	True
>	Greater than	5 > 3	True
<	Less than	3 < 5	True
>=	Greater or equal	5 >= 5	True

<=	Less or equal	4 <= 5	True

➤ Logical Operators

Operator	Name	Example	Result
and	Logical AND	True and True	True
		True and False	False
or	Logical OR	False or True	True
		False or False	False
not	Logical NOT	not True	False
		not False	True

➤ Assignment Operators (compound)

Operator	Example	Meaning
+=	x += 1	x = x + 1
-=	x -= 2	x = x - 2

*=	x *= 3	x = x * 3
/=	x /= 4	x = x / 4

Indentation in Python

Indentation is **crucial** in Python. It is used to define the **structure** and **blocks of code**, unlike some other languages that use {} or begin...end.

What is Indentation?

Indentation means **spaces (or tabs)** at the beginning of a line to indicate **code hierarchy** — which code belongs to a particular block (like inside a loop, function, or conditional).

✅ Correct Example:

python Copy Edit

```python
if 5 > 2:
    print("Five is greater than two")
```

> The `print()` is **indented**, so Python knows it's part of the `if` block.

❌ Incorrect Example (will raise an error):

python Copy Edit

```python
if 5 > 2:
print("Five is greater than two")
```

> ❗ Error: **IndentationError: expected an indented block**

Example with Multiple Levels

```python
def greet(name):
    if name:
        print("Hello", name)
    else:
        print("No name provided")

greet("Geet")
```

> indentation

Tip: Use an Editor with Auto-Indent

Tools like **VS Code**, **PyCharm**, or **Jupyter Notebooks** automatically handle indentation and show visual guides.

Expressions

What is an Expression in Python?

An **expression** is a piece of code that **produces a value**.

Think of it like:

"Anything that Python can calculate or evaluate to get a result."

Examples of Expressions:

- 2 + 3 → results in 5
- "Hi" + " there" → results in "Hi there"

- len("hello") → results in 4
- 10 > 3 → results in True

In all these examples, Python **evaluates** the expression and gives you a **value**.

Key Point:

An **expression is a *value-producing* line of code.**

That's it — short and sweet! Would you like to try a few yourself?

Quiz Time

Match the variable assignment concepts with their corresponding examples in Python. This will test your understanding of how different assignment techniques are applied.

SUBMIT

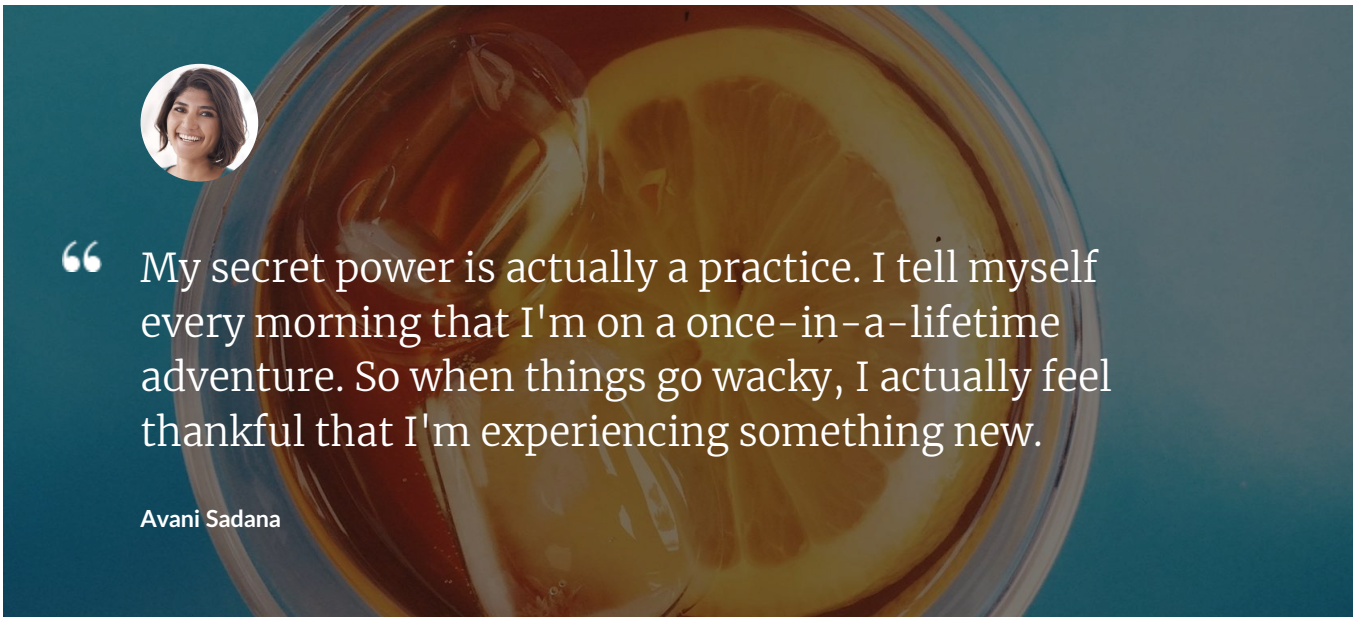
Why is indentation important in Python?

- It helps Python ignore unnecessary lines of code.
- It makes the code run faster.
- It is only required for aesthetic purposes.
- It defines the structure and hierarchy of the code.

SUBMIT

Match each Python operator with its correct description or example. This will test your understanding of how different operators function in Python.

SUBMIT



> My secret power is actually a practice. I tell myself every morning that I'm on a once-in-a-lifetime adventure. So when things go wacky, I actually feel thankful that I'm experiencing something new.

Avani Sadana

Lesson 4 of 7

Working with Escape Sequences

GS Geetu Sodhi

Escape Sequence

An **escape sequence** in programming (especially in Python) is a way to represent **special characters** in a string that cannot be typed directly or would otherwise be interpreted differently by the language.

Definition:

An **escape sequence** starts with a **backslash (\)**, followed by one or more characters that together represent a **special character or instruction**.

Why Use Escape Sequences?

They are used to:

- Insert special characters (like newline, tab, backslash)
- Include characters that normally have a special meaning (like quotes)
- Represent characters that are not printable (like Unicode symbols

🔷 Examples in Python:

Escape Sequence	Description	Example	Output
`\n`	Newline	`"Hello\nWorld"`	```
Hello			
World```			
`\t`	Tab	`"A\tB"`	`A B`
`\'`	Single quote	`'It\'s fine'`	`It's fine`
`\"`	Double quote	`"She said \"yes\""`	`She said "yes"`
`\\`	Backslash	`"Folder\\File"`	`Folder\File`

Raw Strings (to avoid escape sequences)

Sometimes, escape sequences get in the way (like when writing file paths or regular expressions). You can use **raw strings**:

python Copy Edit

```python
raw_string = r"C:\Users\Geet\Documents"
print(raw_string)
# Output: C:\Users\Geet\Documents
```

> The r tells Python **not** to process backslashes as escape characters.

Quiz Time

Which of the following is an example of a valid escape sequence in Python?

- n\
- \x
- \n
- /t

SUBMIT

Lesson 5 of 7

Comments

GS Geetu Sodhi

Comments in Programming

Comments are notes or explanations **written in code** for humans to read — they help explain **what the code does** but are **ignored by Python** when the program runs.

Why Use Comments?

- To explain **what a piece of code does**
- To **temporarily disable** code (for testing)
- To leave **notes** for other developers (or future you!)

Comments in Python

Python supports two types of comments:

1. Single-line Comments

```python
# This is a comment
x = 5  # This sets x to 5
```

2. Multi-line (Block) Comments

Everything after # on that line is ignored by Python. Python doesn't have a true multi-line comment, but you can write multiple single-line comments:

This is a multi-line comment # explaining the next block of code # in detail

```python
# This is a multi-line comment
# explaining the next block of code
# in detail
```

3. Docstrings (Not exactly comments)

Triple quotes """..."""" or '''...''' used at the start of functions/classes/modules to describe their purpose.Docstrings are used for documentation, and can be accessed using help().

```python
def greet(name):
    """This function greets the user by name."""
    print("Hello", name)
```

Quiz Time

Which of the following is a correct way to write a single-line comment in Python?

- ◯ # This is a comment
- ◯ /* This is a comment */
- ◯
- ◯ // This is a comment

SUBMIT

Lesson 6 of 7

Input and Output in Python

GS Geetu Sodhi

print()

The print() function in Python is used to display output on the screen.

```python
print("Hello, world!")
# Output: Hello, world!
```

printing multiple items

.

```python
print("Hello", 123, True)
# Output: Hello 123 True
```

Attributes of print() - end, sep (separator)

1) sep – Defines how values are separated. Default is a **space** (' ').

python Copy Edit

```python
print("A", "B", "C", sep="-")
# Output: A-B-C

print("2025", "06", "04", sep="/")
# Output: 2025/06/04
```

2) end –Defines what is printed **after** the output. Default is **newline** ('\n').

python Copy Edit

```python
print("This is", end=" ")
print("on the same line.")
# Output: This is on the same line.
```

printing variables

When printing a **variable** in Python, **do not** put it inside quotes.

Using comma(,) to print multiple items including string and variables

```python
name = "Geet"
age = 13
print("Name:", name, "Age:", age)
```

Using String concatenation

```python
print("Name: " + name + ", Age: " + str(age))
```

Printing With f-string (formatted string literals)

```python
name = "Geet"
age = 13
print(f"Name: {name}, Age: {age}")
```

Example of Using Escape Sequence in print()

1. New Line (`\n`)

```python
print("Hello\nWorld")
```

Output:

```
Hello
World
```

2. Tab (`\t`)

```python
print("Name\tAge")
print("Geet\t13")
```

Output:

```
Name    Age
Geet    13
```

Quiz Time

What is the default behavior of the 'sep' attribute in the print() function?

- ◯ It separates values with a space.
- ◯ It separates values with a comma.

- ◯ It does not separate values by default.
- ◯ It separates values with a newline.

SUBMIT

Match each concept related to Python's print() function with its corresponding example. This will test your understanding of how different attributes and methods of the print() function are applied.

≡ 1 sep attribute	print('A', 'B', 'C', sep='-') outputs: A-B-C
≡ 2 end attribute	print('Hello', end='!') outputs: Hello!
≡ 3 f-string	name = 'John'; print(f'Hello, {name}') outputs: Hello, John
≡ 4 Escape sequence	

print('Line1\nLine2') outputs: Line1 (newline) Line2

SUBMIT

Select all that apply: Which of the following statements about the Python print() function are correct?

- [ ] The print() function is used to display output on the screen.
- [ ] The print() function cannot handle multiple items in a single call.
- [ ] Variables must always be enclosed in quotes when using the print() function.
- [ ] The 'end' attribute defines what is printed after the output, with a default value of a newline ('\n').
- [ ] The 'sep' attribute defines how values are separated, with a default value of a space (' ').

SUBMIT

input()

input() is a **built-in function** used to **get input from the user** through the keyboard.
It always returns the input as a **string**.

🔷 Basic Syntax

python Copy Edit

```python
variable = input("Your message here: ")
```

The message inside the quotes is called a **prompt** — it tells the user what to type.

Important: input() is always a string

Even if the user types a number, it will be treated as a string.

```python
a = input("Enter first number: ")
b = input("Enter second number: ")
print("Result:", a + b)
```

If you enter:

```sql
Enter first number: 5
Enter second number: 10
```

Output will be:

```makefile
Result: 510
```

➡️ Because both `a` and `b` are strings, `a + b` joins them like text: `"5" + "10"` = `"510"`

Correct Version: Convert Input to Integers for Math

```python
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
print("Result:", a + b)
```

Now with same input:

```sql
Enter first number: 5
Enter second number: 10
```

Output:

```makefile
Result: 15
```

Convert Input to Integer or Float

- Use int() if you want whole numbers (integers).
- Use float() if you want decimal numbers.

```python
num_int = int(input("Enter an integer: "))
num_float = float(input("Enter a decimal number: "))
print(f"Integer: {num_int}, Decimal: {num_float}")
```

What if the User Enters Invalid Data?

If the user types something that **is not a number**, Python will raise a ValueError.

Quiz Time

What is the default behavior of the input() function in Python?

- ○ It automatically trims whitespace from the input.
- ○ It raises an error if the user enters a non-numeric value.
- ○ It always returns the user input as a string.
- ○ It converts user input to an integer by default.

SUBMIT

Match the questions about the input() function in Python with their correct answers. This will test your understanding of how input() behaves and its key attributes.

≡ 1	What does the input() function return by default?	It returns the input as a string.
≡ 2	How should you handle numeric input from the user?	Convert it using int() for integers or float() for decimals.
≡ 3	What happens if the user enters invalid data when converting input to a number?	Python raises a ValueError.
≡ 4	What is the purpose of the input() function?	To get input from the user through the keyboard.

SUBMIT

You're the master of your life, the captain of your ship. Steer it with intention. Will you skirt the coast from one safe harbor to the next? Or will you sail into the vast open blue? Every day you get to decide anew what course to chart.

Quiz

GS Geetu Sodhi

Question

01/14

What is the default behavior of the input() function in Python?

- ◯ It always returns the user input as a string.
- ◯ It converts user input to an integer by default.
- ◯ It raises an error if the user enters a non-numeric value.
- ◯ It automatically trims whitespace from the input.

Question

02/14

What is the primary purpose of docstrings in Python?

- ○ To explain what a specific line of code does for other developers.
- ○ To temporarily disable a block of code for testing purposes.
- ○ To execute a block of code multiple times.
- ○ To provide documentation for functions, classes, or modules that can be accessed using the help() function.

Question

03/14

Match each Python token from the given code snippet with its correct classification. This will test your understanding of how different tokens are identified and classified in Python.

≡ 1	:	Keyword	⌄
≡ 2	print	Identifier	⌄
≡ 3	score	Operator	⌄
≡ 4	>	Literal	⌄
≡ 5	50	Delimiter	⌄
≡ 6	if	Identifier (a function)	⌄

Question

04/14

Why is indentation important in Python?

- ◯ It defines the structure and hierarchy of the code.
- ◯ It is only required for aesthetic purposes.
- ◯ It makes the code run faster.
- ◯ It helps Python ignore unnecessary lines of code.

Question

05/14

What does the assignment operator '=' do in Python?

- It assigns the evaluated right-hand value to the left-hand variable.
- It creates a new variable without assigning a value.
- It swaps the values of two variables.
- It compares two values to check if they are equal.

Question

06/14

Which of the following is a correct way to write a single-line comment in Python?

- This is a comment
- /* This is a comment */
- // This is a comment
- # This is a comment

Question

07/14

What does Python's dynamic typing feature allow you to do?

- [ ] Automatically convert variables to strings when printed.
- [ ] Declare the type of a variable before using it.
- [ ] Change the type of a variable during program execution.
- [ ] Restrict a variable to a single data type throughout its lifecycle.

Question

08/14

An _____ starts with a backslash (\), followed by one or more characters that together represent a special character or instruction.

Type your answer here

Question

09/14

In Python, _____ are the smallest units of a program that have meaning to the Python interpreter, such as keywords, identifiers, operators, literals, and delimiters.

Type your answer here

Question

10/14

Match each Python data type with its correct definition. This will test your understanding of the characteristics and purposes of Python's built-in data types.

≡ 1 Complex	A sequence of characters enclosed in single or double quotes.
≡ 2 Float	A whole number without a decimal point.
≡ 3 String (str)	A number that includes a decimal point.
≡ 4 None type	A number with both a real and an imaginary part.
≡ 5 Boolean (bool)	A data type that represents True or False values.
≡ 6 Integer (int)	A type that signifies no value or a null value.

Question

11/14

How does Python define blocks of code?

- By using curly braces.
- By using semicolons.
- By using keywords like 'begin' and 'end'.
- By using indentation.

Question

12/14

How can you indicate a raw string in Python to avoid processing escape sequences?

- ○ Add a backslash at the end of the string.
- ○ Prefix the string with an 'r'.
- ○ Use double backslashes in the string.
- ○ Enclose the string in triple quotes.

Question

13/14

Which of the following is a valid variable name in Python?

- [ ] 1student
- [ ] student grade
- [ ] if
- [ ] student_grade

Question

14/14

What is the default behavior of the 'sep' attribute in the print() function?

- It separates values with a newline.
- It separates values with a space.
- It separates values with a comma.
- It does not separate values by default.