PYTHON–START TO CODE

BRIDGE COURSE

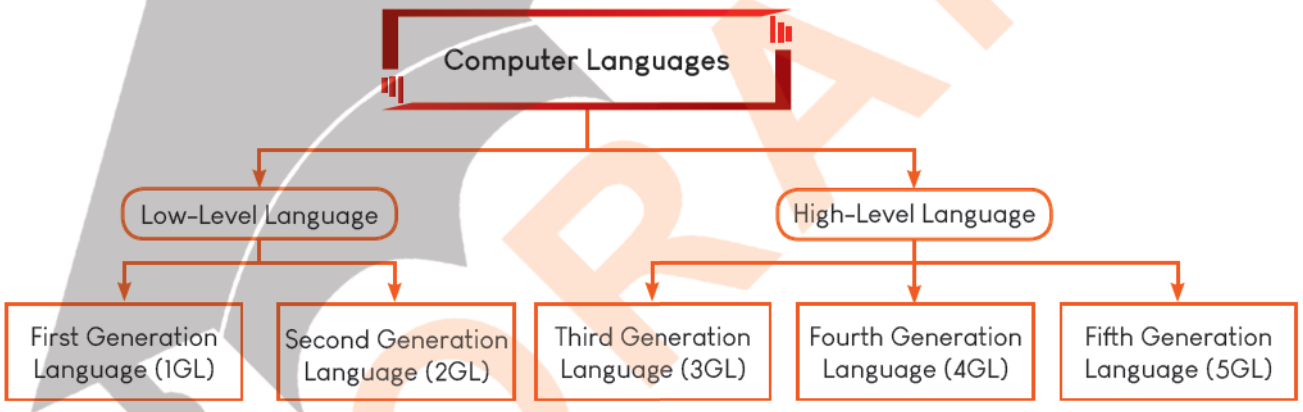
GRADE-7

PRIMARY PREVIEW

- Computer Languages
- Installing Python
- Input and Output
- Variables in Python
- Operators
- Language Translator
- Modes in Python
- Character Set
- Data Types
- Precedence of Operators
- Python
- Opening an Existing Program
- Keywords and Identifiers
- Comments in Python
- Some More Programs

COMPUTER LANGUAGES

Programming languages are used to communicate with computers through words or characters that describe actions. Writing these instructions is called programming and the people who do it are called programmers.



LOW-LEVEL LANGUAGE

A **low-level language** is machine-dependent and works directly with a computer's hardware, giving exact instructions for managing it.

First Generation Language

The **First Generation** of programming languages or machine language, uses binary digits (0 and 1) that computers directly understand. It is machine-dependent and works only on specific hardware.

Second Generation Language

The **Second Generation** of programming languages or assembly language, uses symbols or mnemonics (like MOV, ADD, SUB) instead of binary. It is machine-dependent but easier to read and write.

HIGH-LEVEL LANGUAGE

A **high-level language** lets programmers write machine-independent programs using English-like commands, making it easier to read and write than low-level languages. It is used to create software for different computers and devices.

FACT File

Fortran, developed in **1957**, was the first high-level language specifically designed for scientific and mathematical computations.

Third Generation Language

Third Generation languages use English-like commands, are easier to read and write and are machine-independent with a compiler or interpreter. Examples include C, C++ and Java.

Fourth Generation Language

Fourth Generation languages focus on what to do, not how to do it. They are faster, easier and closer to human language. Examples include SQL, MATLAB and R.

Fifth Generation Language

Fifth Generation languages use visual tools and logic-based programming. They aim to build systems that solve problems using **Artificial Intelligence**. Examples include Prolog, LISP, etc.

LANGUAGE TRANSLATOR

A **language translator** converts high-level code into machine language so the computer can run it. The main types are Assembler, Compiler and Interpreter.

- Assembler: An **assembler** translates assembly language into machine code (0s and 1s) that the computer can understand and process.
- Compiler: A **compiler** translates the entire source code into machine language at once and shows all errors after compilation.
- Interpreter: An **interpreter** translates and runs code line by line, showing errors one at a time, unlike a compiler.

PYTHON

Python is a simple, readable high-level language used in web development, data analysis, AI and scientific computing. Created by Guido van Rossum in the late 1980s, it was named after the comedy show Monty Python's Flying Circus.

FEATURES OF PYTHON

Python is a simple, powerful and versatile language with many applications. Its key features are listed below:

- Python's simple, English-like syntax makes it easy for beginners to learn and understand programming concepts.
- Python executes code line by line, providing immediate results, which is ideal for step-by-step learning.
- Python is freely available to download and use without any license fees.
- Python code can run on different operating systems, such as Windows, macOS and Linux, without requiring modifications.
- Python includes a vast standard library that offers a wide range of pre-written code and functions for various tasks.

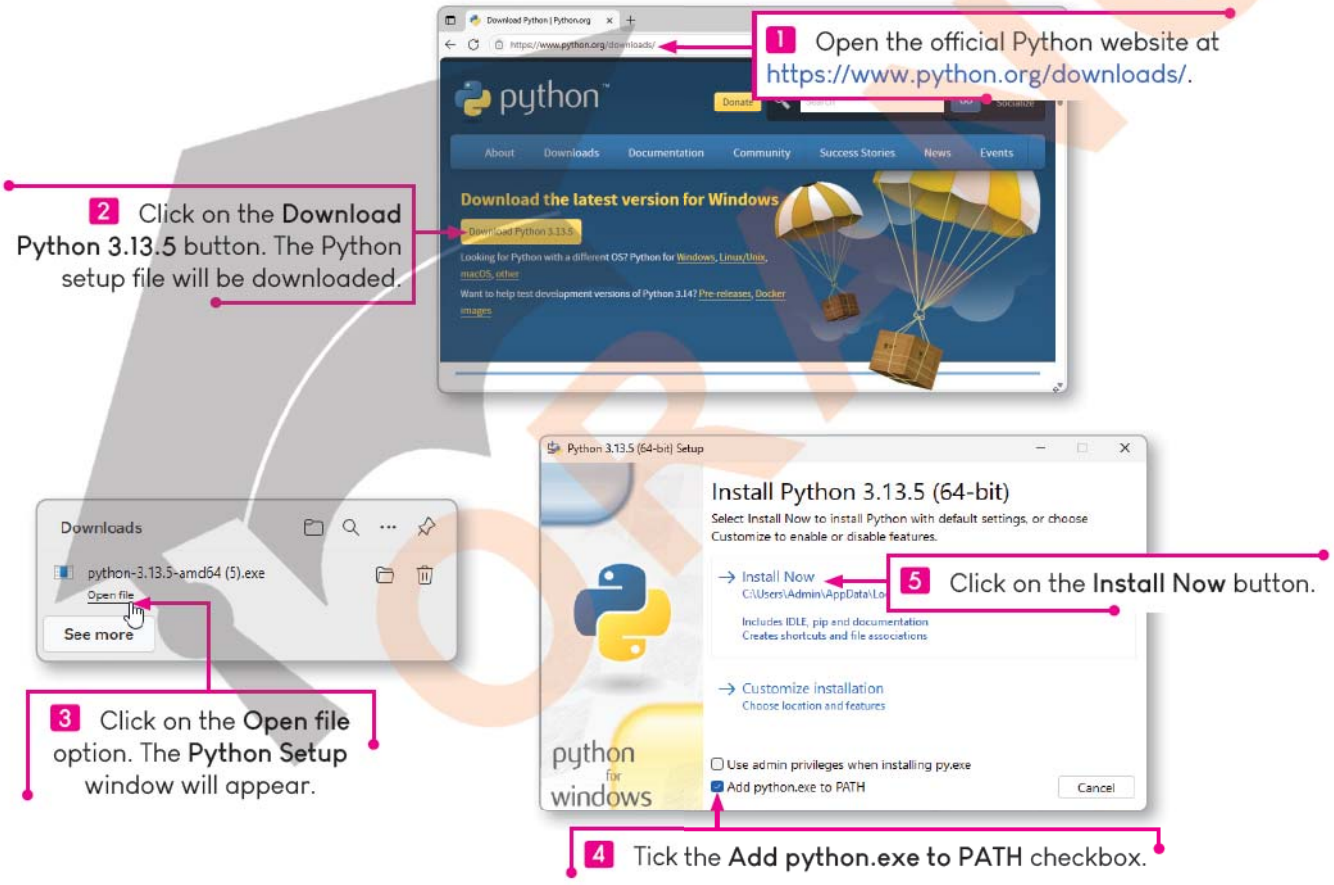
FACT File

Python is one of the most popular programming languages, with over **12 million** developers worldwide.

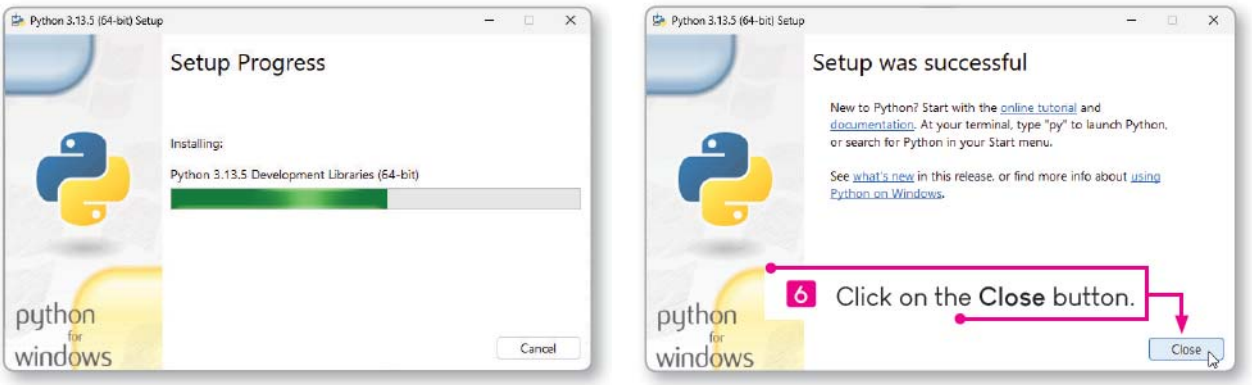
INSTALLING PYTHON

Installing Python is a simple process that can be completed in a few easy steps.

Follow these steps to download and install the Python software:



After a few seconds, once the installation is complete, a message will appear saying **Setup was successful**.



Python is now installed and ready to use on your computer.

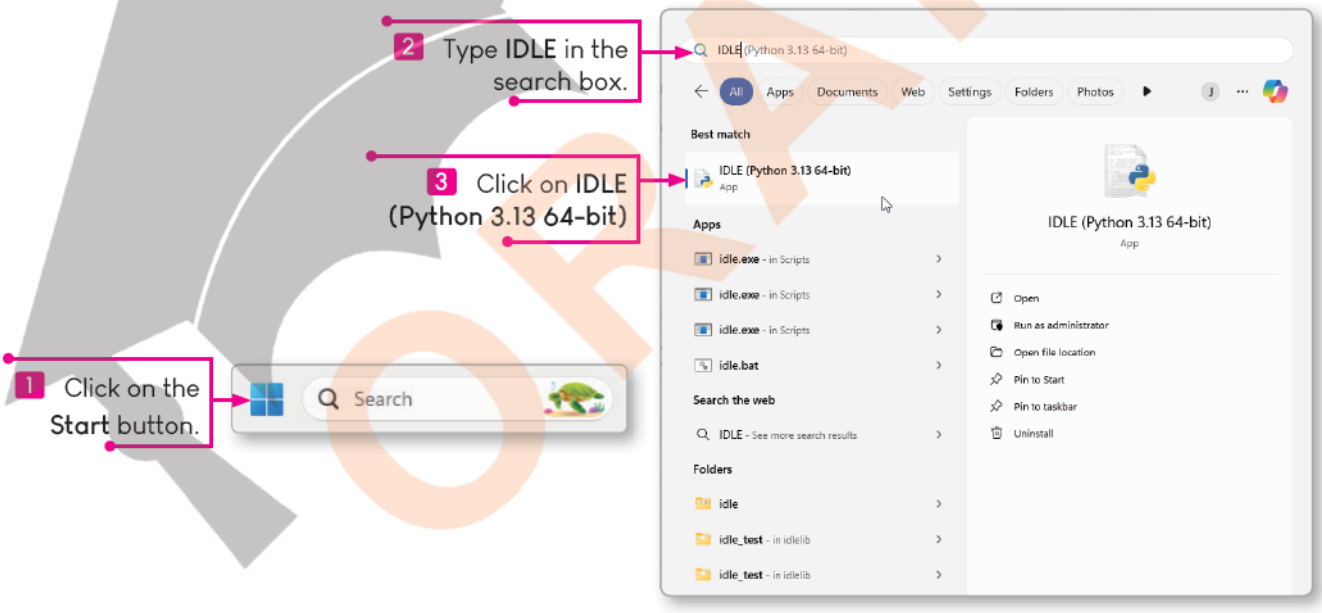
MODES IN PYTHON

Python offers two main ways to interact with the programming environment: **Shell Mode** and **Script Mode**. Each mode has its own use cases and advantages depending on the task at hand.

SHELL MODE

Shell Mode allows you to execute Python code interactively. You can type Python code directly at the prompt (>>>) and it will be executed immediately. This is a great way to experiment with Python syntax or perform quick calculations.

To start Python and work in Shell Mode, follow the given steps:

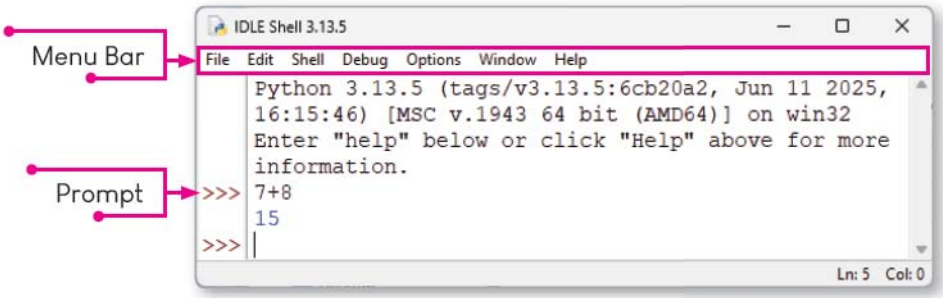


The main IDLE window will appear. IDLE stands for **Integrated Development and Learning Environment**. It is a code editor that allows you to write and run Python programs directly within the editor.

Components of the IDLE Shell Window

The IDLE shell window has two main components: the Menu Bar and the Prompt.

- **Menu Bar:** Located at the top of the window, it offers options for file management, running scripts and customising IDLE, including File, Edit, Shell, Debug, Options, Window and Help.
- **Prompt:** This is the area to type Python code, marked by >>>, where results appear immediately after each command.

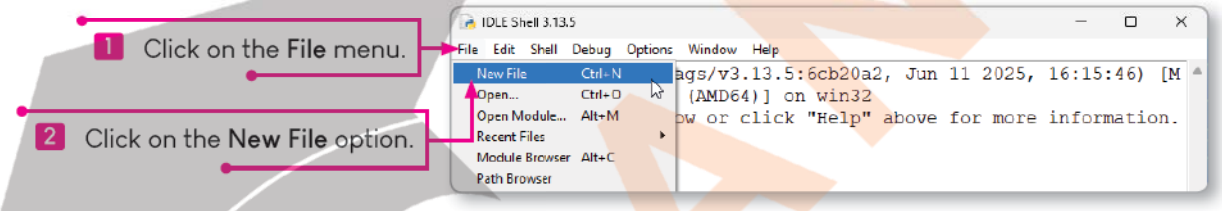


SCRIPT MODE

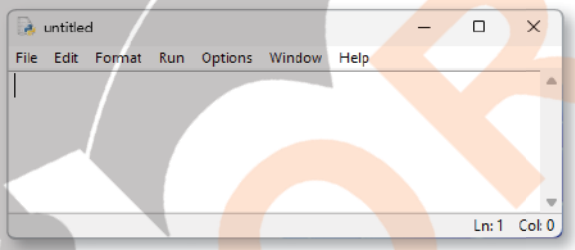
Script Mode is used for writing longer Python programs that can be saved to your computer and run later. IDLE provides a built-in text editor where you can write, edit and save Python scripts.

Creating a New File

In Python IDLE, creating a new file allows you to write and save your Python programs. To create a new file, follow the given steps:



A new window will open where you can start writing your Python code and save it for execution.



SHORT SIGN

To create a new file:
Ctrl + N

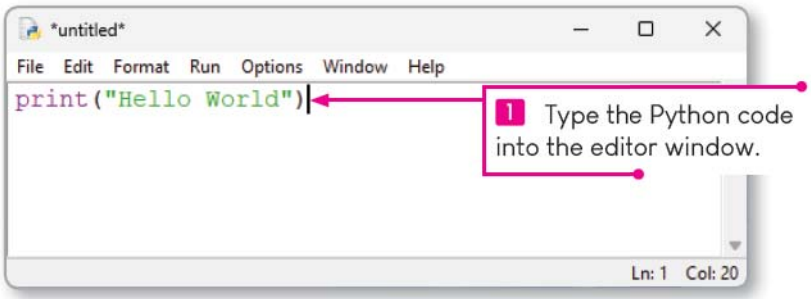
Writing and Saving a Program

Once you've created a new file, you can begin writing your Python program. To write and save a program, follow the given steps:

Ask AI AGENT

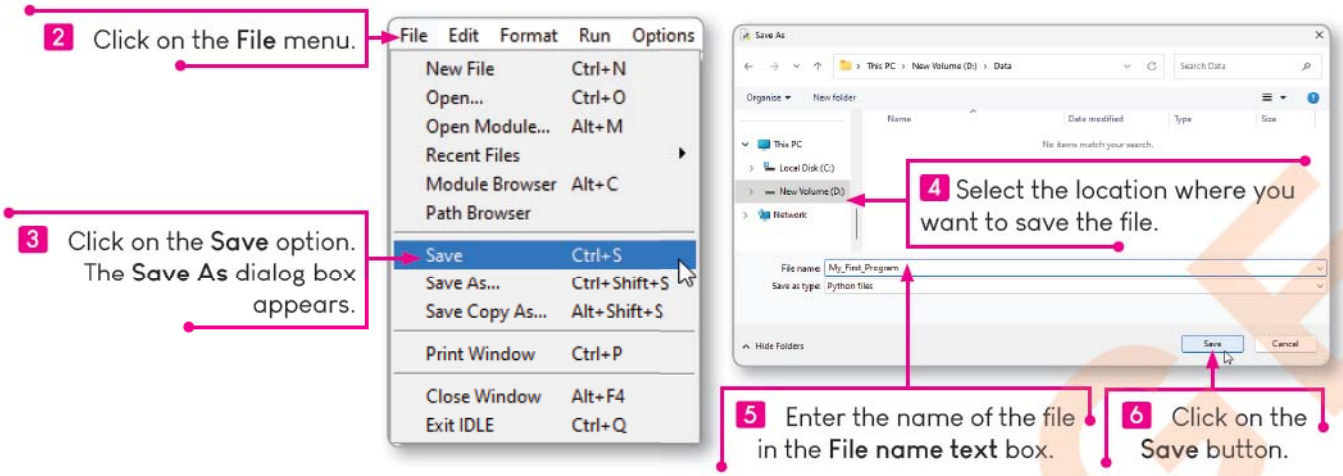
List some of the most popular applications or tools that were created using Python.

+ | Study



SHORT SIGN

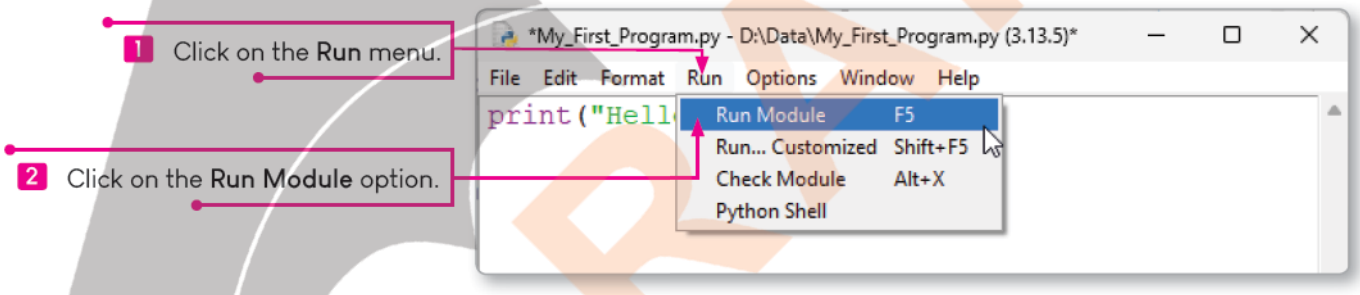
To save the current file: **Ctrl + S**



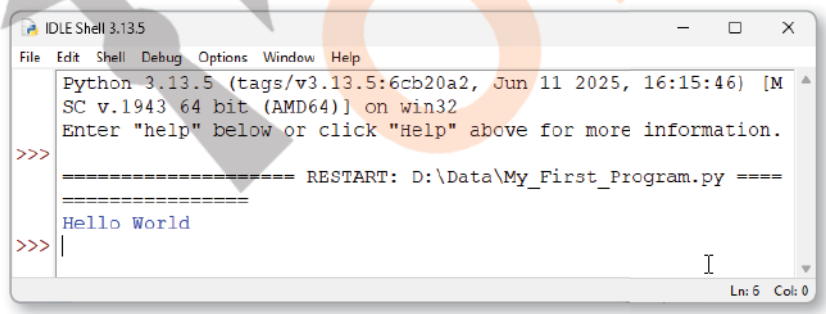
The Python program is now saved on your computer, ready to be executed.

Executing a Program

After writing and saving the Python program, you can run it to see the output. To execute the Python program, follow the given steps:



The program will execute and the output will appear in the Shell window.



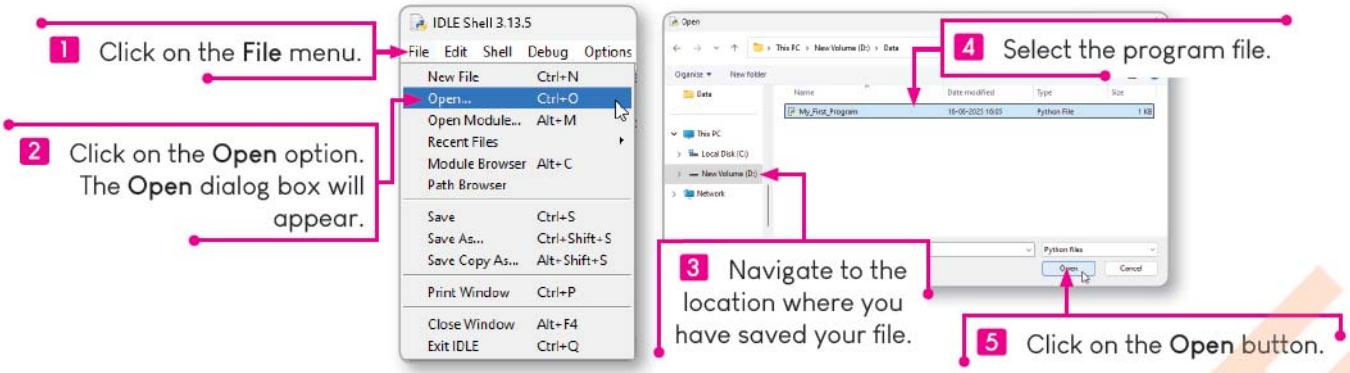
SHORT SIGN

To run the program: **F5**

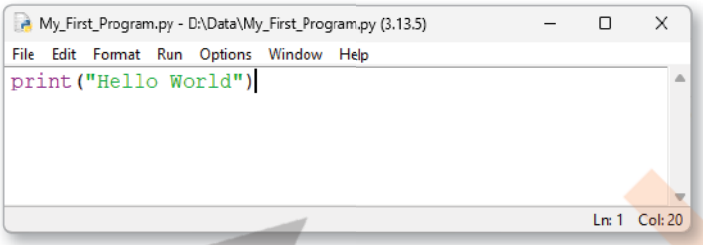
Running a program helps to test and debug code effectively.

OPENING AN EXISTING PROGRAM

You can open an existing Python program to make changes or run it again. To open an existing Python program, follow the given steps:



The selected Python program is now open in the editor, ready for you to modify or execute.



INPUT AND OUTPUT

Python provides built-in functions to interact with users and handle input and output operations.

THE INPUT( ) FUNCTION

The **input()** function is used to take input from the user. It reads a line of text entered by the user, which is then returned as a string.

Syntax of input() function:

```
input(<prompt>)
```

Here, the prompt is the string or message you want to display on the screen.

For example:

```
name = input("Enter your name: ")
>>> Enter your name: Ankit
```

The text 'Ankit' entered by the user will be stored in the variable **name**.

THE PRINT( ) FUNCTION

The **print()** function is used to display output on the screen in Python. It takes the specified text, variable or value and prints it in the Python shell or terminal.

Syntax of print() function:

```
print(<value>)
```

Here, the value can be an expression, string or variable that you want to display. You can also print multiple items by separating them with commas.

For example:

```
print("Let's code together!")
>>> Let's code together!
```

Program 1 To use the input( ) and print( ) functions.

Program1.py

File Edit Format Run Options Window Help

```
name = input("Enter your name: ")
print("Your name is", name)
```

Output

```
Enter your name: Ankit
Your name is Ankit
```

RAPID RECALL Tick (✓) if you know this.

1. The print() function is used to display output on the screen in Python.
2. The input() function is used to take input from the user.

CHARACTER SET

A **character set** is the collection of characters a programming language recognises. Python accepts all ASCII and Unicode characters, including:

- **Letters of the alphabet:** All capital letters (A-Z) and small letters (a-z).
- **Digits:** All digits from 0 to 9.
- **Special symbols:** Python supports a wide range of special symbols such as:
  " ' ; : ! ` ~ @ # $ % ^ & * ( ) _ + - = { } [ ]
- **White spaces:** White spaces like tab space, blank space, newline (\n) and carriage return (\r).

KEYWORDS AND IDENTIFIERS

In Python, **keywords** and **identifiers** are two important concepts that define the structure and naming conventions of a program.

KEYWORDS

Keywords are reserved words with special meanings in Python, used for tasks like control flow (if, for, while), functions (def) and classes (class). They cannot be used as variable names. Here is a list of Python keywords:

FACT File

Python has **35** reserved keywords that cannot be used as variable names.

`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`	

You cannot use keywords as variable names. For example, you cannot name a variable 'if' or 'for'.

IDENTIFIERS

An **identifier** is a name used to uniquely identify a variable, function, class or object in Python. It represents the values you store, like numbers or text.

Examples of Identifiers:

- **Variable identifiers:** `age, score, temperature`
- **Function identifiers:** `print_message(), calculate_sum()`
- **Class identifiers:** `Person, Student`

VARIABLES IN PYTHON

In Python, **variables** store values that can be accessed or changed later. Their names are identifiers and you don't need to specify their type because Python is dynamically typed.

RULES FOR NAMING VARIABLES

When creating variable names, make sure to follow these rules:

- Variable names can contain letters (a-z, A-Z), digits (0-9) and underscores (_).
- Variable names must begin with a letter or an underscore (_), but not a number.
- Variable names are case-sensitive, so 'age' and 'Age' are considered two different variables.
- Variable names cannot use Python keywords (such as `if, else, for`) as variable names.
- Variable names cannot contain spaces.

DECLARING AND INITIALISING A VARIABLE

In Python, you can declare and initialise a variable in one step. For example:

```
x = 5
y = 10
print("x =", x)
print("y =", y)
```

In this example, the variables **x** and y are initialised with the values 5 and 10, respectively. The output will be:

```
x = 5
y = 10
```

You can also assign the same value to multiple variables in a single line:

```
a = b = 30
print("a =", a)
print("b =", b)
```

In this case, both **a** and **b** will store the value **30** and the output will be:

```
a = 30
b = 30
```

Additionally, you can assign multiple values to multiple variables in a single line. For example:

```
name, age, city = "John", 25, "London"
print("Name:", name)
print("Age:", age)
print("City:", city)
```

The output will be:

```
Name: John
Age: 25
City: London
```

DATA TYPES

A **data type** defines the kind of value a variable can hold and the operations allowed on it.

For example, names are strings (text) and can be changed to uppercase or lowercase, while age, price and marks are numbers, either integers or floats and can be used in calculations.

Let's look at some commonly used data types in Python:

- **int:** It represents positive or negative whole numbers (without fractions).

  For example:

  ```
  a = 9
  ```

- **float:** It represents real number, where a fraction is denoted by a decimal point.

  For example:

  ```
  a = 7.5
  ```

- **string:** A sequence of one or more characters enclosed in single or double quotes.

  For example:

  ```
  a = "hello"
  ```

In Python, the input() function always returns a value as a string. To use the input as a number, you must convert it into the appropriate data type. This can be done using the following functions:

- **int():** Converts the input value into an integer.

  For example:

  ```
  a = int(input("Enter a number: "))
  ```

  This code converts the user's input into an integer before storing it in the variable a.

- **float():** Converts the input value into a floating-point number.

  For example:

  ```
  b = float(input("Enter a number: "))
  ```

  This code converts the user's input into a floating-point value before storing it in the variable b.

COMMENTS IN PYTHON

Comments explain code for users and are ignored by Python during execution. They can be single-line or multiline.

SINGLE-LINE COMMENT

A single-line comment is used to add brief notes or explanations to a single line of code. It starts with the **hash symbol (#)**.

Syntax of a single-line comment:

```
# This is a single-line comment
```

MULTILINE COMMENT

A multiline comment is used when the comment spans more than one line. Python does not have a specific syntax for multiline comments, but you can use either consecutive single-line comments or triple quotes (''' or """) to create a multiline comment.

By writing multiple single-line comments one after another:

```
# This is a multiline comment
# using single-line comment syntax
```

By enclosing the text within triple quotes (''' or """):

```
'''
This is a
multiline comment
'''
```

OR

```
"""
This is a
multiline comment
"""
```

Program 2 To demonstrate how to use both single-line and multiline comments.

Program2.py

File Edit Format Run Options Window Help

```
#This is a single line comment.
print("Orange Education")
'''
This is a
multiline comment
using triple quotes
'''
print("I love Python")
```

LIVE LEARNING

Write a Python program to display comments about your best friend in multi line comments and print his name using the print statement.

Output

```
Orange Education
I love Python
```

OPERATORS

Operators are symbols that perform mathematical, logical or relational operations on values called operands. For example, in 15 + 7, 15 and 7 are operands and + is the operator. Python has different categories of operators.

ARITHMETIC OPERATORS

Arithmetic operators perform arithmetic operations between two operands. The arithmetic operators are defined in the following table:

Operator	Name	Description	Example (x = 5, y = 3)	Output
+	Addition	Adds values on either side of the operator.	x + y	8
-	Subtraction	Subtracts the right operand from the left operand.	x - y	2
*	Multiplication	Multiplies values on either side of the operator.	x * y	15
/	Division	Divides the left operand by the right and returns a float.	x / y	1.6667
%	Modulus	Divides the left operand by the right and returns the remainder.	x % y	2

Operator	Name	Description	Example (x = 5, y = 3)	Output
**	Exponentiation	Raises the left operand to the power of the right operand.	x ** y	125
//	Floor division	Divides the left operand by the right and discards the fraction.	x // y	1

Program 3 To demonstrate the use of all the arithmetic operations.

Program3.py

File Edit Format Run Options Window Help

```
x = 5
y = 3

result_add = x + y # Addition
print("Addition:", result_add)

result_sub = x - y # Subtraction
print("Subtraction:", result_sub)

result_mul = x * y # Multiplication
print("Multiplication:", result_mul)

result_div = x / y # Division
print("Division:", result_div)

result_mod = x % y # Modulus
print("Modulus:", result_mod)

result_exp = x ** y # Exponentiation
print("Exponentiation:", result_exp)

result_floor = x // y # Floor Division
print("Floor Division:", result_floor)
```

LIVE LEARNING

Write a Python program to calculate simple interest: Use the formula (P * R * T) / 100, where P is the principal, R is the rate and T is the time.

Output

```
Addition: 8
Subtraction: 2
Multiplication: 15
Division: 1.6666666666666667
Modulus: 2
Exponentiation: 125
Floor Division: 1
```

RELATIONAL OPERATORS

Relational operators compare values on both sides and return True or False.

The following table describes the relational operators:

Operator	Name	Description	Example (x = 5, y = 3)	Output
==	Equal to	Returns True if both operands are equal.	x == y	False
!=	Not equal to	Returns True if both operands are not equal.	x != y	True
>	Greater than	Returns True if the left operand is greater than the right.	x > y	True

Operator	Name	Description	Example (x = 5, y = 3)	Output
<	Less than	Returns True if the left operand is less than the right operand.	x < y	False
>=	Greater than or equal to	Returns True if the left operand is greater than or equal to the right.	x >= y	True
<=	Less than or equal to	Returns True if the left operand is less than or equal to the right operand.	x <= y	False

Program 4 To demonstrate the use of all relational operators.

Program4.py

File Edit Format Run Options Window Help

```
# Assign values to variables
x = 5
y = 3

# Equal to (==)
print("x == y:", x == y)

# Not equal to (!=)
print("x != y:", x != y)

# Greater than (>)
print("x > y:", x > y)

# Less than (<)
print("x < y:", x < y)

# Greater than or equal to (>=)
print("x >= y:", x >= y)

# Less than or equal to (<=)
print("x <= y:", x <= y)
```

LIVE LEARNING

Write a Python program to calculate the value of the following expression if x = 8, y = 4 and z = 10:
x(x+y)*(z-y)//x

Output

```
x == y: False
x != y: True
x > y: True
x < y: False
x >= y: True
x <= y: False
```

LOGICAL OPERATORS

Logical operators are used to combine conditional statements and return a **Boolean value** (True or False) based on the conditions provided.

Python supports the following logical operators:

Operator	Name	Description	Example (x = 5, y = 3)	Output
not	NOT	Reverses the result, returns True if the condition is False or vice versa.	not(x > y)	False
and	AND	Returns True if both conditions are True.	x > 3 and y < 4	True
or	OR	Returns True if at least one condition is True.	x > 3 or y > 4	True

Program 5 To demonstrate the use of all the logical operators.

Program5.py

File Edit Format Run Options Window Help

```
# Assign values to variables
x = 5
y = 3

# Logical NOT (not)
print("not (x > y):", not (x > y))

# Logical AND (and)
print("x > 3 and y < 4:", x > 3 and y < 4)

# Logical OR (or)
print("x > 3 or y > 4:", x > 3 or y > 4)
```

HINTS & HACKS

In case of the OR operator, the second condition is checked only if the first is False.

Output

```
not (x > y): False
x > 3 and y < 4: True
x > 3 or y > 4: True
```

ASSIGNMENT OPERATORS

Assignment operators are used to assign the value of the right expression to the left operand. The assignment operators are described in the following table:

Operator	Name	Description	Example (x = 5, y = 3)
=	Assignment	Assigns the value of the operand on the right side to the left side operand.	x = y
+=	Addition assignment	Adds the right operand to the left operand and assigns the result to the left operand. x += y is equivalent to x = x + y.	x += y
-=	Subtraction assignment	Subtracts the right operand from the left operand and assigns the result to the left operand. x -= y is equivalent to x = x - y.	x -= y
*=	Multiplication assignment	Multiplies the right operand with the left operand and assigns the result to the left operand. x *= y is equivalent to x = x * y.	x*=y
/=	Division assignment	Divides the left operand by the right operand and assigns the result to the left operand. x /= y is equivalent to x = x / y.	x /= y
%=	Modulus assignment	Takes the modulus of the left operand by the right operand and assigns the result to the left operand. x %= y is equivalent to x = x % y.	x %= y
//=	Floor division assignment	Performs floor division on the operands and assigns the result to the left operand. x //= y is equivalent to x = x // y.	x //= y
**=	Exponentiation assignment	Raises the left operand to the power of the right operand and assigns the result to the left operand. x **= y is equivalent to x = x ** y.	x **= y

Program 6 To demonstrate the use of all the assignment operators.

Program6.py

File Edit Format Run Options Window Help

```
# Assign values to variables
a = 5
y = 3
# Assignment (=)
x = a
print("Assignment: x =", x)
# Addition assignment (+=)
x += a
print("Addition Assignment (+=): x =", x)
# Subtraction assignment (-=)
x -= y
print("Subtraction Assignment (-=): x =", x)
# Multiplication assignment (*=)
x *= y
print("Multiplication Assignment (*=): x =", x)
# Division assignment (/=)
x /= y
print("Division Assignment (/=): x =", x)
# Modulus assignment (%=)
x %= y
print("Modulus Assignment (%=): x =", x)
# Floor division assignment (//=)
x //= y
print("Floor Division Assignment (//=): x =", x)
# Exponentiation assignment (**=)
x **= y
print("Exponentiation Assignment (**=): x =", x)
```

Output

```
Assignment: x = 5
Addition Assignment (+=): x = 10
Subtraction Assignment (-=): x = 7
Multiplication Assignment (*=): x = 21
Division Assignment (/=): x = 7.0
Modulus Assignment (%=): x = 1.0
Floor Division Assignment (//=): x = 0.0
Exponentiation Assignment (**=): x = 0.0
```

INTERDISCIPLINARY LEARNING

21st Century Skills #Critical Thinking

Write a Python program to calculate the area of a square.

MATHEMATICS | LAB ACTIVITY

PRECEDENCE OF OPERATORS

Operator precedence determines the order in which operations are performed. Operators with higher precedence are evaluated first. When operators have the same precedence, Python evaluates them from left to right (associativity). The following table shows the operator precedence, starting with the highest precedence at the top:

Operator	Name
()	Parentheses
**	Exponentiation
*, /, //, %	Multiplication, Division, Floor Division, Modulus
+, –	Addition, Subtraction
<, <=, >, >=, !=, ==	Relational operators
not	Logical operators
and	
or	
=, %=, /=, //=, -=, +=, *=, **=	Assignment operators

RAPID RECALL

Tick (✓) if you know this.

1. A single-line comment begins with the hash symbol (#).
2. Keywords are reserved, so you cannot use them as variable names or identifiers.

Program 7 To multiply two numbers entered by the user.

Program7.py

File Edit Format Run Options Window Help

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
print("The product of the two numbers is:", num1 * num2)
```

Output

```
Enter the first number: 23
Enter the second number: 17
The product of the two numbers is: 391.0
```

Program 8 To calculate the perimeter of a rectangle.

Program8.py

File Edit Format Run Options Window Help

```
length = float(input("Enter the length of the rectangle: "))
width = float(input("Enter the width of the rectangle: "))
perimeter = 2 * (length + width)
print("The perimeter of the rectangle is:", perimeter)
```

Output

```
Enter the length of the rectangle: 20
Enter the width of the rectangle: 40
The perimeter of the rectangle is: 120.0
```

Program 9 To calculate the average of three numbers.

Program9.py

File Edit Format Run Options Window Help

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))
average = (num1 + num2 + num3) / 3
print("The average of the three numbers is:", average)
```

Output

```
Enter the first number: 13
Enter the second number: 17
Enter the third number: 21
The average of the three numbers is: 17.0
```

Program 10 To calculate the cube of a number.

Program10.py

File Edit Format Run Options Window Help

```
num = float(input("Enter a number: "))
print("The cube of a number is:", num ** 3)
```

Output

```
Enter a number: 8
The cube of a number is: 512.0
```

TECH TERMS

- **Function:** A reusable block of code that performs a specific task.
- **Library:** A collection of pre-written code, including functions and modules, that helps perform common tasks without writing code from scratch.

REWIND RUN

- Python is a simple, powerful and versatile language with many applications.
- Shell Mode allows you to execute Python code interactively.
- Script Mode is used for writing longer Python programs that can be saved to your computer and run later.
- The input() function is used to take input from the user.
- The print() function is used to display output on the screen in Python.
- A character set is the collection of characters a programming language recognises.
- Keywords are reserved words with special meanings in Python.
- An identifier is a name used to uniquely identify a variable, function, class or object in Python.
- Operators are symbols that perform mathematical, logical or relational operations on values called operands.