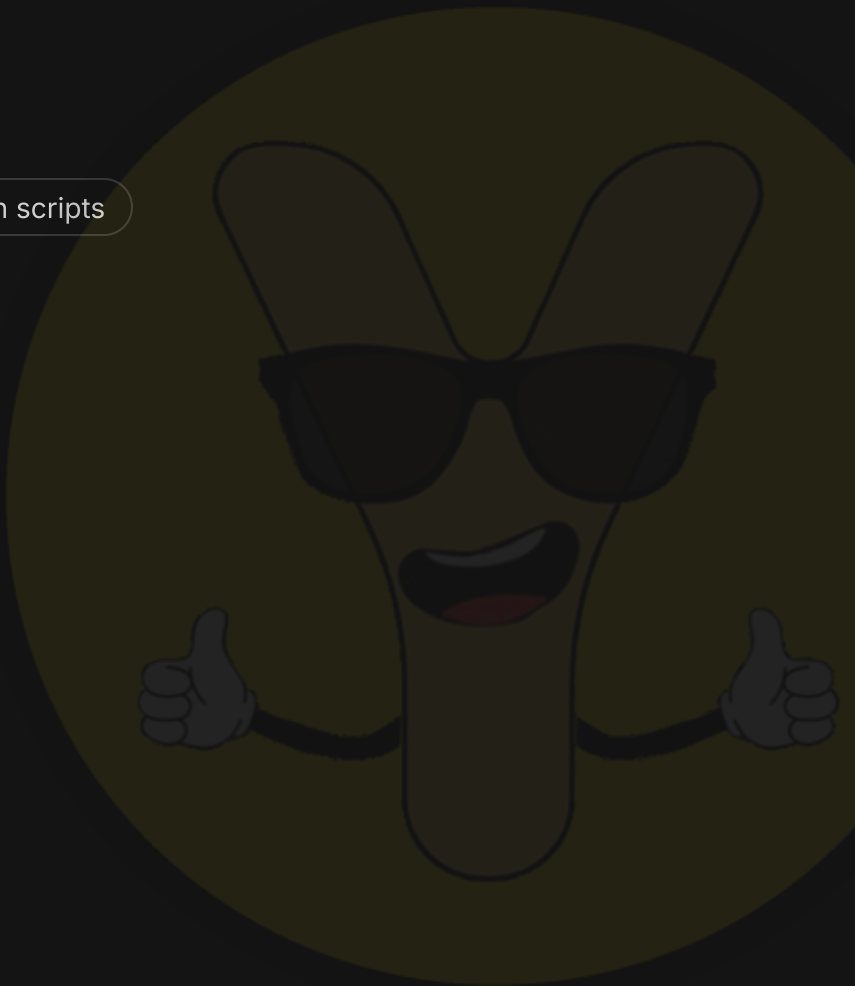
AKTU · 2ND YEAR · BCC302 / BCC402

Python Programming

Unit 1 — Introduction to Python

- The Python Programming Language
- History and features of Python
- Installing Python
- Running Python scripts
- Variables, expressions, statements
- Input/Output operations
- Operators and precedence
- Conditional statements: if, if-else, nested if-else
- Loops: for, while, break, continue, pass



DECODED BY

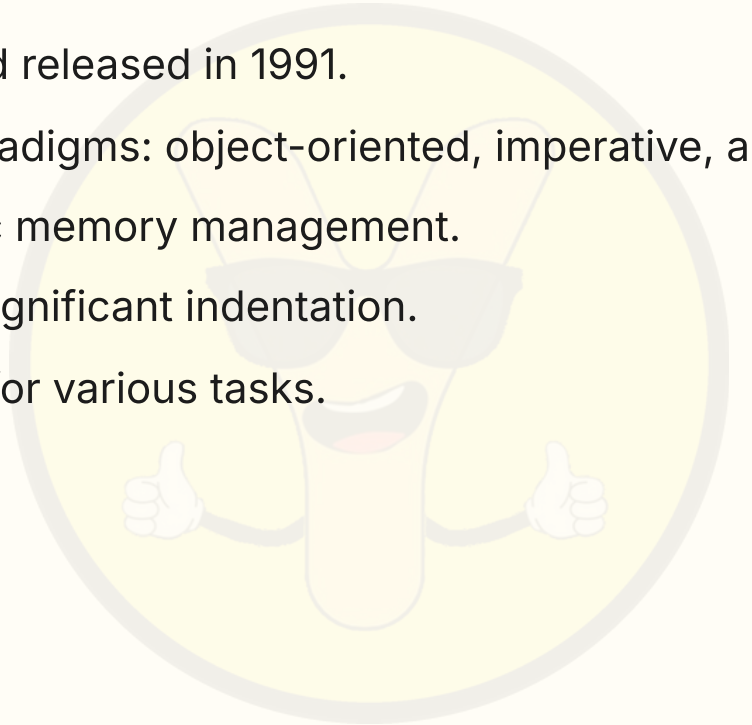
Chirag Jain

Founder, Ycotes · Gold Medalist in Artificial Intelligence, IIT Madras · taught 10,000+ students

Introduction to Python

Python is a high-level, interpreted, general-purpose programming language known for its readability.

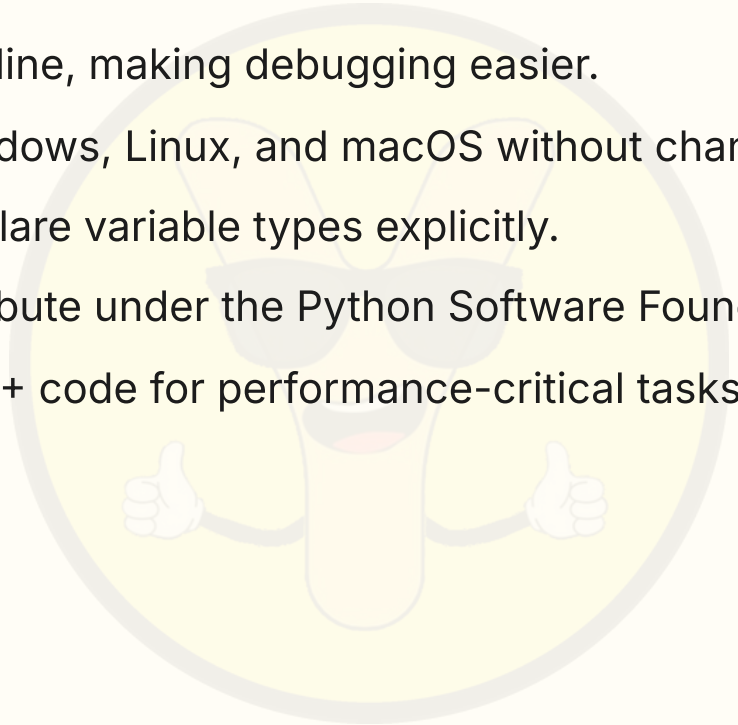
- Designed by Guido van Rossum and released in 1991.
- Supports multiple programming paradigms: object-oriented, imperative, and functional.
- Uses dynamic typing and automatic memory management.
- Emphasizes code readability with significant indentation.
- Extensive standard library support for various tasks.



History and Features

Python evolved from the ABC language and has become a dominant force in modern computing.

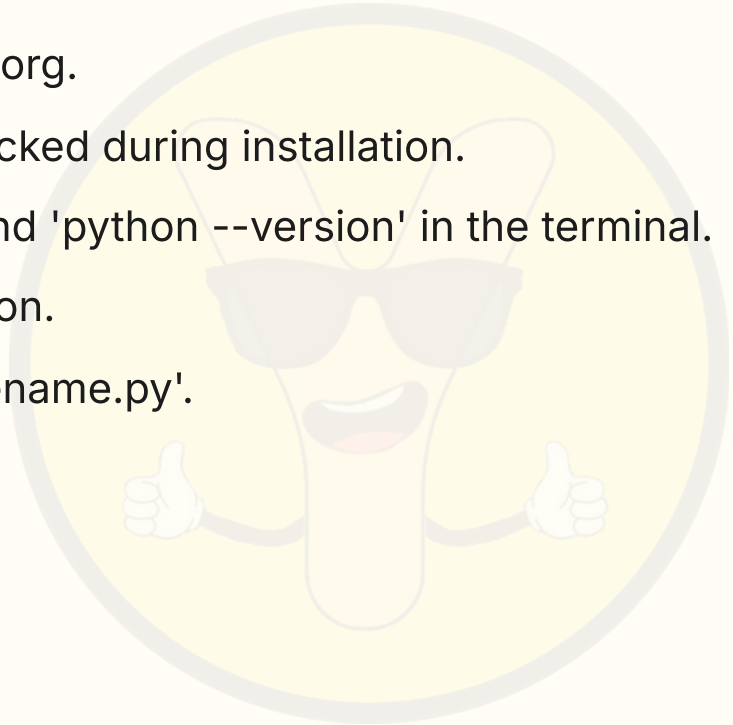
- Interpreted: Executes code line-by-line, making debugging easier.
- Platform Independent: Runs on Windows, Linux, and macOS without changes.
- Dynamically Typed: No need to declare variable types explicitly.
- Open Source: Free to use and distribute under the Python Software Foundation License.
- Extensible: Can integrate with C/C++ code for performance-critical tasks.



Installing and Running Python

Setting up the environment is the first step toward Python development.

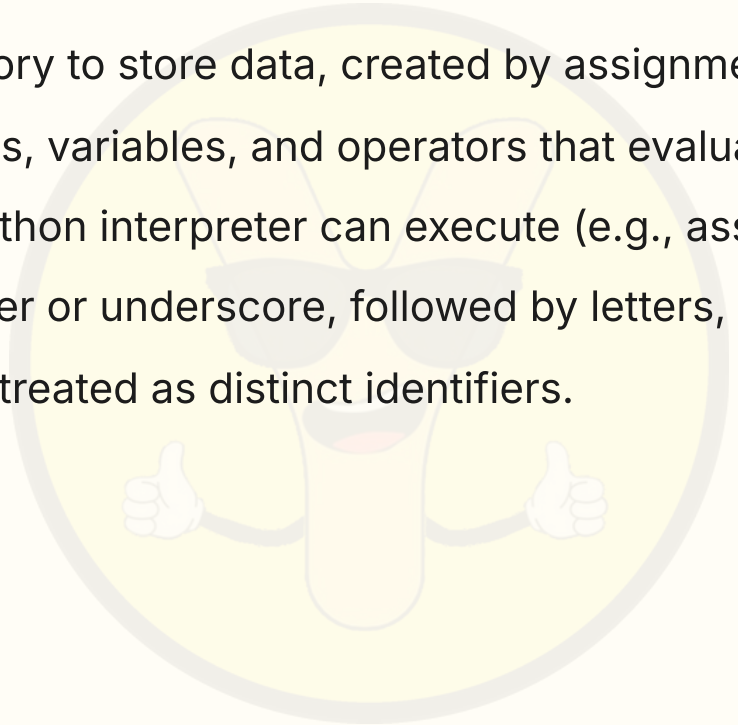
- Download the installer from python.org.
- Ensure 'Add Python to PATH' is checked during installation.
- Verify installation using the command 'python --version' in the terminal.
- Scripts are saved with a .py extension.
- Run scripts via terminal: 'python filename.py'.



Variables, Expressions, and Statements

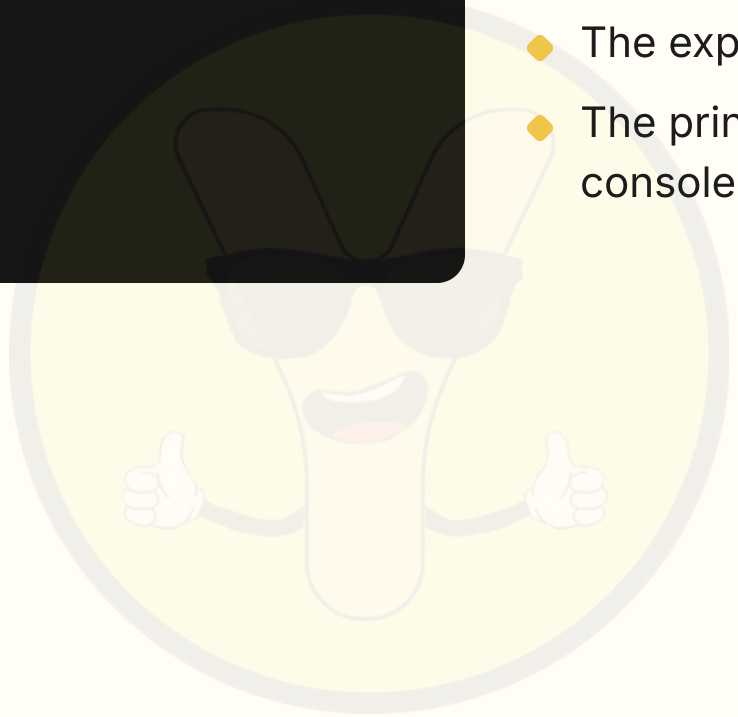
Fundamental building blocks of a Python program.

- Variables: Named locations in memory to store data, created by assignment.
- Expressions: Combinations of values, variables, and operators that evaluate to a single value.
- Statements: Instructions that the Python interpreter can execute (e.g., assignment, print).
- Naming Rules: Must start with a letter or underscore, followed by letters, digits, or underscores.
- Case Sensitivity: 'Var' and 'var' are treated as distinct identifiers.



Variables and Expressions Example

```
x = 10
y = 5
# Expression
result = (x + y) * 2
# Statement
print("Result is:", result)
```

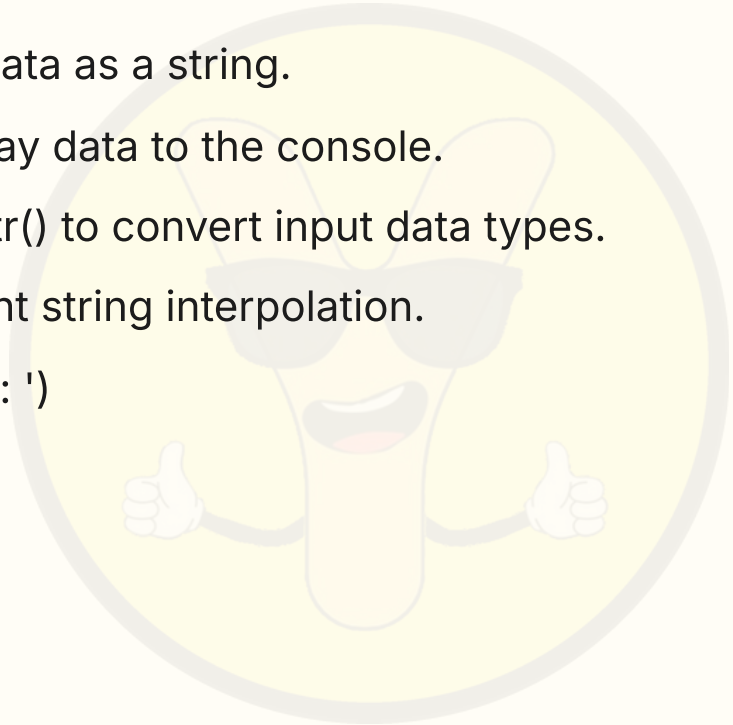


- x and y are variables holding integer values.
- The expression evaluates to 30.
- The print function outputs the result to the console.

Input/Output Operations

Interacting with the user via standard input and output devices.

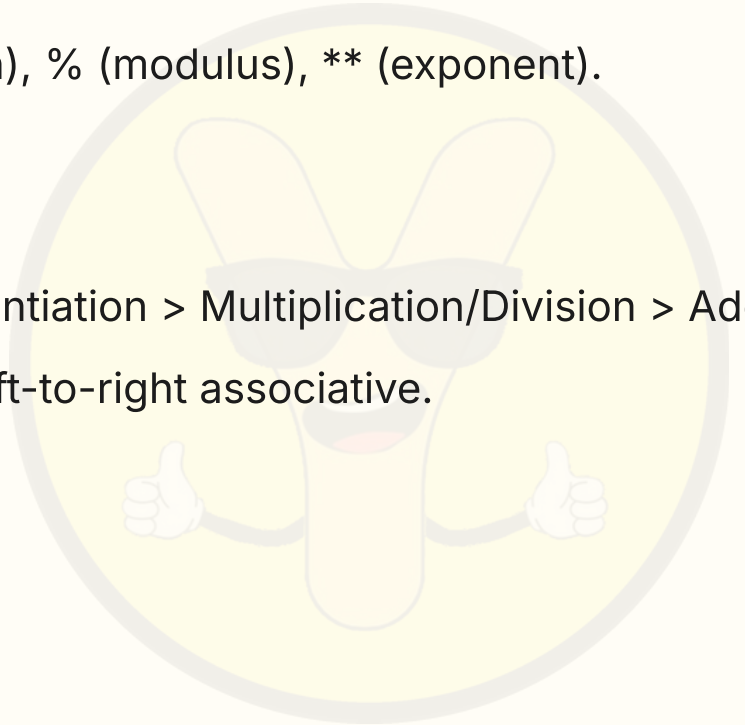
- Input: Use input() function to read data as a string.
- Output: Use print() function to display data to the console.
- Type Casting: Use int(), float(), or str() to convert input data types.
- Formatting: Use f-strings for efficient string interpolation.
- Example: name = input('Enter name: ')



Operators and Precedence

Operators perform operations on variables and values.

- Arithmetic: +, -, *, /, // (floor division), % (modulus), ** (exponent).
- Comparison: ==, !=, >, <, >=, <=
- Logical: and, or, not
- Precedence: Parentheses > Exponentiation > Multiplication/Division > Addition/Subtraction.
- Associativity: Most operators are left-to-right associative.



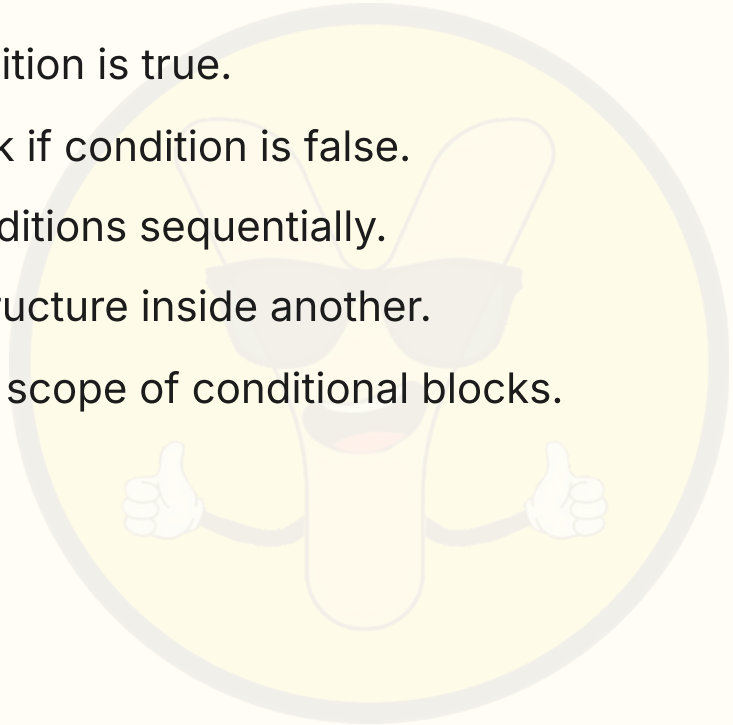
Operator Precedence Table

Operator	Description
()	Parentheses
**	Exponentiation
*, /, //, %	Multiplication/Division
+, -	Addition/Subtraction
<, <=, >, >=	Comparison
==, !=	Equality

Conditional Statements

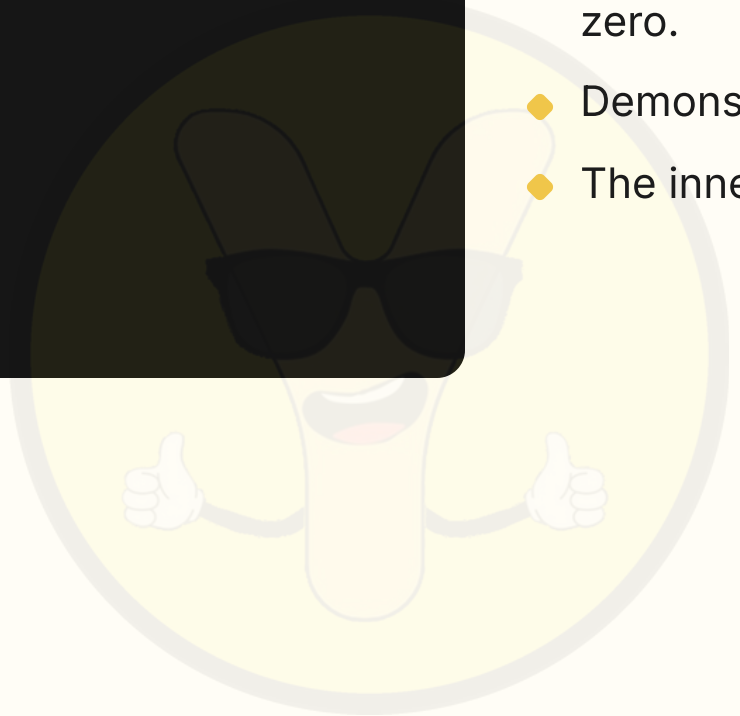
Control flow based on boolean conditions.

- if statement: Executes block if condition is true.
- if-else: Provides an alternative block if condition is false.
- elif: Used for checking multiple conditions sequentially.
- Nested if-else: Placing an if-else structure inside another.
- Indentation: Crucial for defining the scope of conditional blocks.



Nested if-else Example

```
num = 10
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive")
else:
    print("Negative")
```

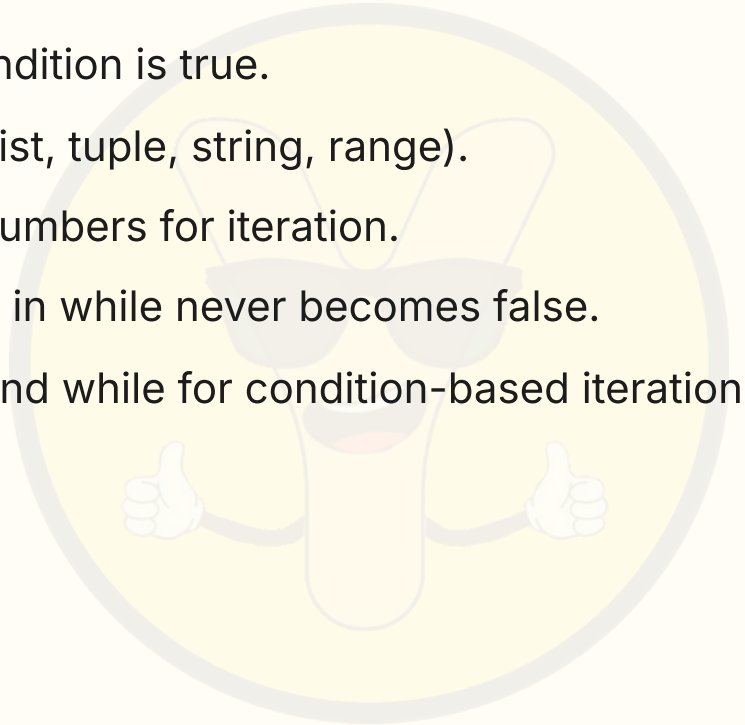


- Checks if a number is positive, negative, or zero.
- Demonstrates scope via indentation.
- The inner if checks for zero specifically.

Loops: for and while

Repeated execution of code blocks.

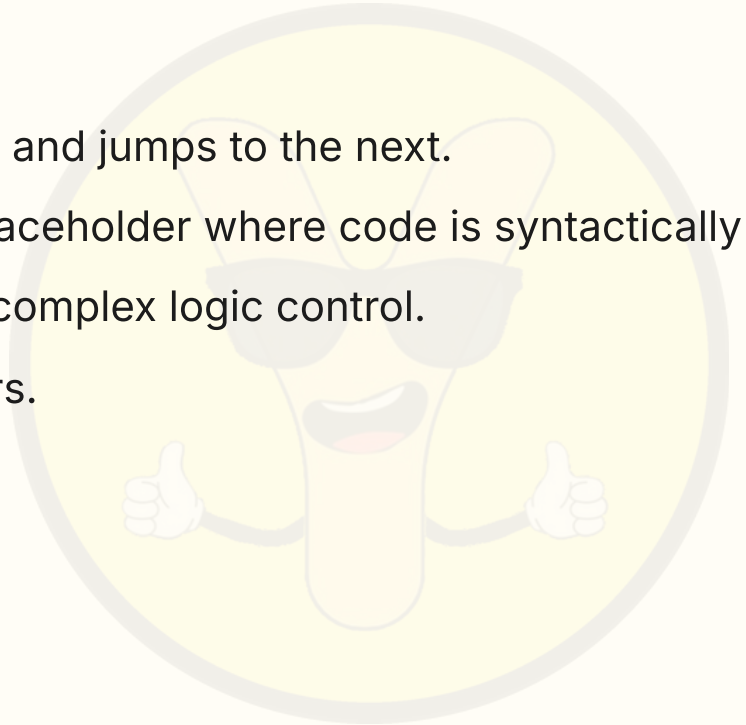
- while loop: Repeats as long as a condition is true.
- for loop: Iterates over a sequence (list, tuple, string, range).
- range(): Generates a sequence of numbers for iteration.
- Infinite loop: Occurs if the condition in while never becomes false.
- Use for loops for known iterations and while for condition-based iterations.



Loop Control Statements

Modifying the flow of loops.

- break: Exits the loop immediately.
- continue: Skips the current iteration and jumps to the next.
- pass: A null statement; used as a placeholder where code is syntactically required.
- These statements are essential for complex logic control.
- Improper use can lead to logic errors.



Loop Control Flow



Flowchart showing how loop control statements influence the execution path.

Exam corner

- **Q1** Explain the key features of Python that make it popular for developers.
- **Q2** Differentiate between interpreted and compiled languages.
- **Q3** What is the importance of indentation in Python? Explain with an example.
- **Q4** Explain operator precedence with an example expression.
- **Q5** Write a Python program to check if a number is prime using a loop.
- **Q6** Difference between break, continue, and pass statements.
- **Q7** How does the input() function work? Explain type casting.
- **Q8** Write a program to print the Fibonacci series up to n terms.

Unit 1 done. You are exam-ready.

Revise the Exam corner once more, then open Unit 2 on theycotes.com. Every AKTU subject, unit by unit, free. Notes by Chirag Jain, Ycotes.

