FREE EDITION · NOTES + 3 SAMPLE MCQS

CUET · COMPUTER SCIENCE · CLASS XI · CODE 308

Brief Overview of Python

CUET unit: Brief Overview of Python

By **UniDrill** · NCERT-grounded study material

WWW.UNIDRILL.IN

📌 Snapshot

- Python is an open-source, high-level, interpreter-based programming language created by Guido van Rossum in 1991; its core syntax and fundamental constructs are covered here.
- The foundational vocabulary of Python programming — keywords, identifiers, variables, data types, operators, expressions, input/output, and control structures — is all heavily tested in CUET.
- CUET tests this material through direct recall of operator behaviour (precedence, floor division, modulus), data type classification, identifier validity rules, and error types, making it one of the highest-yield areas in the subject.
- Built-in functions, conditional statements (if/elif/else), the for loop with range(), and nested loops appear both as conceptual MCQs and as trace-the-output questions.
- These fundamentals are prerequisite for all subsequent Python topics; CUET frequently combines concepts from here (e.g., operator precedence + data types) into single questions.

📖 Detailed Notes

2.1 Core concepts

- **Introduction to Python (§3.1, p. 31):** A program is an ordered set of instructions executed by a computer. Python is a popular, easy-to-learn programming language created by Guido van Rossum in 1991. It is used in software development, web development, scientific computing, big data, and AI. Programs in the NCERT book use Python 3.7.0. (NCERT §3.1, p. 31)
- **Python Shell / Interpreter (§3.1.1, p. 32):** The Python interpreter is also called the Python shell. The symbol `>>>` is the Python prompt indicating readiness to receive instructions. (NCERT §3.1.1, p. 32)
- **Execution Modes (§3.1.2, p. 32–33):** There are two modes — (a) Interactive mode: statements typed directly at `>>>` are executed immediately; cannot save statements for future use. (b) Script mode: program written in a file with `.py` extension, saved and executed. Python has a built-in editor called IDLE (Integrated Development and Learning Environment) for creating scripts. (NCERT §3.1.2, p. 32–33)

- **Python Keywords (§3.2, p. 34):** Keywords are reserved words with specific meaning to the Python interpreter. Python is case-sensitive, so keywords must be written exactly as given (e.g., `True`, `False`, `None`, `if`, `else`, `elif`, `for`, `while`, `def`, `return`, `import`, etc. — 33 keywords listed in Table 3.1). (NCERT §3.2, p. 34)
- **Identifiers (§3.3, p. 34):** Identifiers are names used to identify variables, functions, or other entities. Rules: must begin with an uppercase/lowercase letter or underscore (`_`); followed by letters (a–z, A–Z), digits (0–9), or underscore; cannot start with a digit; cannot be a keyword; cannot contain special symbols like `!`, `@`, `#`, `$`, `%`. (NCERT §3.3, p. 34)
- **Variables (§3.4, p. 34–35):** A variable is an identifier whose value can change. Variables must be assigned a value before use; otherwise an error occurs. Assignment uses `=` (e.g., `gender = 'M'`, `price = 987.9`). Comments in Python start with `#` and are ignored by the interpreter. (NCERT §3.4, p. 34–35)
- **Data Types — Numbers (§3.5.1, p. 36):** Number data type stores numerical values classified into `int` (integer numbers, e.g., -12, 0, 123), `float` (floating-point numbers, e.g., -2.04, 14.23), and `complex` (complex numbers, e.g., 3+4i). `bool` is a subtype of integer with only two constants: `True` (non-zero) and `False` (zero). The built-in function `type()` returns the data type of a variable. (NCERT §3.5.1, p. 36)
- **Data Types — Sequence (§3.5.2, p. 36–37):** A Python sequence is an ordered collection indexed by integers. Three sequence types: (a) String — group of characters enclosed in single or double quotes; numerical operations cannot be performed on strings. (b) List — sequence of comma-separated items enclosed in square brackets `[ ]`; items may be of different data types. (c) Tuple — sequence enclosed in parentheses `( )`; unlike list, items cannot be changed once created. (NCERT §3.5.2, p. 36–37)
- **Data Types — Mapping (§3.5.3, p. 37–38):** Mapping is an unordered data type. The only standard mapping type in Python is Dictionary. Dictionary holds data in key-value pairs enclosed in curly brackets `{ }`. Key and value are separated by a colon (`:`). Keys are usually strings; values can be of any type. Values are accessed by specifying the key in square brackets. (NCERT §3.5.3, p. 37–38)
- **Arithmetic Operators (§3.6.1, p. 38–39):** Python supports `+` (addition/concatenation), `-` (subtraction), `*` (multiplication/repetition), `/` (division — returns float), `%` (modulus — returns remainder), `//` (floor division — returns quotient without decimal part, also called integer division), `**` (exponent). (NCERT §3.6.1, p. 38–39)
- **Relational Operators (§3.6.2, p. 39–40):** Compare operand values and return `True` or `False`. Operators: `==` (equal to), `!=` (not equal to), `>` (greater than), `<` (less than), `>=` (greater than or equal to), `<=` (less than or equal to). Python compares strings lexicographically using ASCII values. (NCERT §3.6.2, p. 39–40)
- **Assignment Operators (§3.6.3, p. 40):** Assign or change a variable's value. Basic: `=`. Compound: `+=`, `-=`, `*=`, `/=`, `%=`, `//=`, `**=`. (NCERT §3.6.3, p. 40)

- **Logical Operators (§3.6.4, p. 40–41):** Three logical operators written in lowercase: `and` (True if both operands are True), `or` (True if any operand is True), `not` (reverses logical state of operand). (NCERT §3.6.4, p. 40–41)
- **Membership Operators (§3.6.5, p. 41):** `in` returns `True` if value is found in the specified sequence; `not in` returns `True` if value is not found. (NCERT §3.6.5, p. 41)
- **Expressions and Operator Precedence (§3.7, §3.7.1, p. 41–42):** An expression is a combination of constants, variables, and operators that always evaluates to a value. When an expression has more than one operator, precedence determines evaluation order. Higher precedence operators are evaluated first. `*` and `/` have higher precedence than `+` and `-`. Parentheses override precedence; expressions inside `()` are evaluated first. Equal-precedence operators are evaluated left to right. (NCERT §3.7, §3.7.1, p. 41–42)
- **Input and Output (§3.8, p. 42–43):** `input()` function takes user input; accepts all input as a string. Syntax: `variable = input([Prompt])`. `int()` converts a string to integer (error if non-numeric). `print()` function outputs data to screen; evaluates expression before displaying. Syntax: `print(value)`. (NCERT §3.8, p. 42–43)
- **Debugging (§3.9, p. 43–44):** Three types of errors: (i) Syntax errors — violations of language rules, interpreter shows error and stops execution (e.g., missing parenthesis). (ii) Logical errors (semantic errors) — program runs but produces wrong output; difficult to identify (e.g., writing `10 + 12/2` instead of `(10+12)/2` for average). (iii) Runtime errors — syntactically correct but interpreter cannot execute (e.g., division by zero). Debugging is the process of identifying and removing logical and runtime errors. (NCERT §3.9, p. 43–44)
- **Functions (§3.10, p. 44–46):** A function is a set of statements grouped under a name that performs specific tasks. Functions are defined once and reused. Python has many built-in functions. A module is a Python file with multiple functions grouped together, imported using `import`. Key aspects: Function Name, Arguments (values passed to the function in parentheses), Return Value (result passed back to calling point). Built-in functions are categorised into Input/Output (`input()`, `print()`), Datatype Conversion (`int()`, `float()`, `str()`, `bool()`, `list()`, `tuple()`, `dict()`, etc.), Mathematical (`abs()`, `max()`, `min()`, `pow()`, `sum()`, `divmod()`), and Others (`len()`, `range()`, `type()`). (NCERT §3.10, p. 44–46)
- **if..else Statements (§3.11, p. 46–47):** Conditional statements allow different code paths based on conditions. Three forms: (a) `if` — executes block only when condition is true; (b) `if...else` — executes `if` block when true, `else` block when false; (c) `if...elif...else` — checks multiple conditions; `elif` means "else if". Python uses indentation (spaces/tabs) to define blocks; incorrect indentation causes syntax errors. (NCERT §3.11, p. 46–47)
- **for Loop (§3.12, p. 48–49):** Used to repeat statements for a known number of times; iterates over a range of values or a sequence (numeric, string, list, or tuple). Syntax: `for <control-variable> in <sequence/range>:`. When all items in range are

exhausted, loop ends and execution continues after the loop. Body of loop must be indented. (NCERT §3.12, p. 48)

- **range() Function (§3.12.1, p. 49):** Built-in function. Syntax: `range([start], stop[, step])`. Generates a sequence of integers from `start` (default 0) up to but not including `stop`, with increment `step` (default 1). All parameters must be integers; step can be positive or negative (not zero). (NCERT §3.12.1, p. 49)
- **Nested Loops (§3.13, p. 50):** A loop inside another loop is called a nested loop. For each iteration of the outer loop, the inner loop executes completely. (NCERT §3.13, p. 50)

2.2 Definitions to memorise

Term	Definition	Page
Program	An ordered set of instructions or commands to be executed by a computer	31
Programming language	Language used to specify instructions to a computer (e.g., Python, C, Java)	31
Python shell	The Python interpreter; `>>>` is the Python prompt	32
IDLE	Integrated Development and Learning Environment — Python's built-in editor	33
Interactive mode	Execution mode where statements are typed at `>>>` and executed immediately	32
Script mode	Execution mode where program is saved as a `.py` file and then executed	33
Keyword	Reserved word with specific meaning to the Python interpreter	34
Identifier	Name used to identify a variable, function, or other entity in a program	34
Variable	An identifier whose value can change during program execution	34
Comment	Non-executable statement starting with `#`; ignored by interpreter	35
Data type	Identifies the type of data a variable can hold and operations performable on it	35
bool	Subtype of integer with two constants: `True` (non-zero) and `False` (zero)	36
List	Ordered sequence of comma-separated items enclosed in square brackets `[ ]`	37
Tuple	Ordered sequence enclosed in parentheses `( )`; items cannot be changed once created	37

Term	Definition	Page
Dictionary	Unordered mapping data type holding key-value pairs enclosed in curly brackets `{ }`	38
Operand	Value(s) on which an operator works	38
Floor division (`//`)	Divides and returns quotient by removing decimal part; also called integer division	39
Modulus (`%`)	Divides and returns the remainder	39
Expression	Combination of constants, variables, and operators that evaluates to a value	41
Operator precedence	Order/hierarchy in which operators are evaluated in an expression	42
Syntax error	Error caused by violation of language rules; interpreter stops execution	44
Logical error	Semantic error; program runs but produces wrong output	44
Runtime error	Syntactically correct statement that interpreter cannot execute (e.g., division by zero)	44
Debugging	Process of identifying and removing logical and runtime errors	44
Function	Set of statements grouped under a name that performs specified tasks	44
Module	A Python file in which multiple functions are grouped together	45
Argument	Value(s) passed to a function, enclosed in parentheses when calling	45
Indentation	Leading whitespace (spaces/tabs) at the beginning of a statement defining code blocks	47
Nested loop	A loop inside another loop	50
`int()`	Built-in function that converts a value (string or float) to integer	43
`float()`	Built-in function that converts a value to a floating-point number	46
`str()`	Built-in function that converts a value to its string representation	46
`len()`	Built-in returning the count of items in a sequence	46
`range()`	Built-in producing a sequence of integers `[start, stop)` with given step	49
`type()`	Built-in returning the type/class of a value	36
`print()`	Built-in function that writes output to the screen	43
`input()`	Built-in function that reads user input as a string	43
`for` loop	Repetition construct iterating over a sequence	48

Term	Definition	Page
`if..elif..else`	Multi-way selection construct	47
Floor division `//`	Returns the integer quotient by removing decimal part	39
Membership operator	`in` / `not in` — tests sequence containment	41
Boolean values	`True` and `False` — capitalised; subtype of int	36

2.3 Diagrams / processes to remember

- **Figure 3.6 — Data Types in Python (p. 36):** Tree diagram showing top-level categories: Numbers (Integer, Floating Point, Complex; Boolean is subtype of Integer), Sequences (Strings, Lists, Tuples), Sets, None, and Mappings (Dictionaries). Memorise this hierarchy — CUET frequently asks which category a data type belongs to.
- **Table 3.3 — Arithmetic Operators (p. 39):** Seven operators with their symbols, operations, and examples. Pay special attention to `//` (floor division) vs `/` (true division) — `5//2 = 2` but `5/2 = 2.5`.
- **Table 3.8 — Built-in Functions (p. 46):** Four-column table categorising built-in functions into Input/Output, Datatype Conversion, Mathematical, and Others. Know which category each listed function belongs to.
- **if..elif..else flow (§3.11, p. 47):** Understand that `elif` conditions are checked sequentially; once one is true, remaining conditions are skipped and the `if` block terminates.
- **range() behaviour (§3.12.1, p. 49):** `range(10)` → 0 to 9; `range(2,10)` → 2 to 9; `range(0,30,5)` → 0,5,10,15,20,25; `range(0,-9,-1)` → 0,-1,-2,...,-8. The stop value is always excluded.

2.4 Common confusions / NTA trap points

- `/` **vs** `//`**:** The `/` operator always returns a float (e.g., `5/2 = 2.5`), while `//` returns an integer result by removing the decimal (e.g., `5//2 = 2`). NTA often puts both as options; students confuse them.
- `input()` **always returns a string:** Even if the user types `19`, the variable stores the string `'19'`, not the integer `19`. To use it as a number, wrap with `int()` or `float()`. A common distractor claims `input()` auto-detects type.
- **Identifier starting with digit is invalid:** `1st_Room` is invalid (starts with digit); `_Percentage` is valid (starts with underscore). NTA tests this with mixed lists of identifiers. Also, a keyword used as identifier (e.g., `True`) is invalid.
- **Logical error vs runtime error:** Logical errors do not stop execution — the program runs and gives wrong output. Runtime errors stop execution mid-way. NTA presents code snippets and asks students to classify the error type.

- **Tuple is immutable, List is mutable (NCERT §3.5.2, p. 37).** Items in a tuple cannot be changed after creation; list items can. Both use commas to separate items but tuple uses `( )` and list uses `[ ]`. A distractor often swaps the bracket types.
- `True` / `False` **must be capitalised (NCERT §3.2 & §3.5.1, p. 34, 36).** Python is case-sensitive — `true` is not a keyword.
- **Modulus on negative numbers (NCERT §3.6.1, p. 39).** Python's `%` follows the sign of the divisor; `-7 % 3 = 2`, not -1.
- `range()` **is half-open (NCERT §3.12.1, p. 49).** `range(2, 10)` includes 2 but excludes 10.
- **Indentation is structural (NCERT §3.11, p. 47).** Wrong indentation is a syntax error in Python — not a stylistic concern.
- **Dictionary keys must be unique (NCERT §3.5.3, p. 38).** Reassigning the same key overwrites the previous value.
- **NCERT uses Python 3.7.0 (NCERT §3.1, p. 31).** All syntax must conform to Python 3 — Python 2 idioms (like `print` statement without parentheses) are wrong.

🎯 Practice MCQs

Q1. Which of the following is a valid Python identifier?

A. 1st_Room

B. Total-Marks

C. _Percentage

D. Hundred$

Q2. Consider the following Python expression: `20 + 30 * 40`. What value will Python evaluate it to?

A. 2000

B. 1220

C. 50

D. 1200

Q3. Which of the following statements about the `//` operator in Python is correct?

A. It performs true division and always returns a float

B. It returns the remainder when one number is divided by another

C. It divides and returns the quotient by removing the decimal part

D. It raises the left operand to the power of the right operand

📊 PYQ Alignment

Python fundamentals are among the most heavily tested areas in CUET Computer Science, appearing consistently across 2023–2025 papers with questions on operator precedence, data type identification, identifier validity rules, and tracing output of short code snippets involving if-else and for loops. NTA particularly favours questions combining floor division/modulus with precedence, and error-type classification. See PYQ archive for Computer Science.