FREE EDITION · NOTES + 3 SAMPLE MCQS

CUET · COMPUTER SCIENCE · CLASS XI · CODE 308

Getting Started with Python

CUET unit: Getting Started with Python

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

WWW.UNIDRILL.IN

UniDrill

📌 Snapshot

- Python 3 is a high-level, interpreted, open-source programming language; the Python interpreter executes programs line by line.
- The fundamental building blocks of any Python program are: keywords, identifiers, variables, comments, data types, operators, expressions, statements, input/output, and type conversion.
- Data types (numbers, sequences, sets, None, mappings) and operator precedence are directly tested in CUET through expression-evaluation and output-prediction questions.
- Debugging means handling three error types — syntax, logical, and runtime — which NTA uses in scenario-based MCQs asking students to identify error types.
- These fundamentals are prerequisite for all subsequent Python topics; CUET regularly draws 5–7 questions from this material.

📖 Detailed Notes

2.1 Core concepts

- **Programming language and Python introduction:** An ordered set of instructions executed by a computer is a program; the language used to specify those instructions is a programming language. Computers understand machine language (0s and 1s); high-level languages like Python need translators. Python uses an **interpreter** that translates and executes statements one by one, stopping at the first error. A compiler, by contrast, translates the entire source code at once before execution. (NCERT §5.1, p. 87)
- **Features of Python:** Python is high-level, free, open-source, interpreted, case-sensitive, portable, platform-independent, has a rich library of predefined functions, supports web development, and uses indentation for blocks and nested blocks. (NCERT §5.1.1, p. 88)
- **Working with Python / Python shell:** To run a Python program, a Python interpreter (also called Python shell) must be installed. The symbol `>>>` is the Python prompt indicating the interpreter is ready. (NCERT §5.1.2, p. 88)
- **Execution modes — Interactive vs Script:** In **interactive mode**, a single statement typed at `>>>` is executed immediately; statements cannot be saved for

future reuse. In **script mode**, multiple statements are written in a `.py` file, saved, and executed via Run > Run Module (F5) in IDLE, or by typing the file name at the prompt. (NCERT §5.1.3, p. 89)

- **Python keywords:** Keywords are reserved words with a fixed meaning in Python; they must be written in exact case (e.g., `True`, `False`, `None`, `and`, `or`, `not`, `if`, `elif`, `else`, `for`, `while`, `break`, `continue`, `pass`, `return`, `def`, `class`, `import`, `from`, `global`, `nonlocal`, `lambda`, `yield`, `try`, `except`, `finally`, `raise`, `assert`, `del`, `in`, `is`, `with`, `as`). There are 33 keywords in Python 3. (NCERT §5.2, p. 90–91)
- **Identifiers:** Names used to identify variables, functions, or other entities. Rules: must begin with a letter (a–z, A–Z) or underscore `_`; followed by letters, digits (0–9), or underscores; cannot start with a digit; cannot be a keyword; no special symbols (`!`, `@`, `#`, `$`, `%`). Identifiers are case-sensitive. (NCERT §5.3, p. 91)
- **Variables:** A variable is uniquely identified by a name (identifier) and refers to an object stored in memory. Variable declaration in Python is **implicit** — a variable is automatically created when it is first assigned a value. The interpreter replaces a variable name with its current value in expressions. (NCERT §5.4, p. 91–92)
- **Comments:** Begin with `#`; everything from `#` to the end of that line is ignored by the interpreter. Used to document code and improve readability. (NCERT §5.5, p. 92–93)
- **Everything is an object:** Every value in Python is an object with a unique identity (ID) equivalent to its memory address; `id()` returns this identity. When two variables hold the same value, they share the same object ID. (NCERT §5.6, p. 93–94)
- **Data types — Numbers:** `int` (integers, e.g., –12, 0, 125), `float` (real/floating-point, e.g., –2.04, 14.23), `complex` (e.g., 3+4j). `bool` is a subtype of `int`; `True` is non-zero/non-null/non-empty; `False` is zero. `type()` function returns the data type of a variable. (NCERT §5.7.1, p. 94–95)
- **Data types — Sequence:** An ordered collection indexed by integers. Three sequence types: **String** (group of characters enclosed in single or double quotes; numerical operations cannot be performed on strings), **List** (items separated by commas, enclosed in `[ ]`; mutable), **Tuple** (items separated by commas, enclosed in `( )`; immutable — cannot be changed once created). (NCERT §5.7.2, p. 95–96)
- **Data types — Set:** Unordered collection enclosed in `{ }`; no duplicate entries; elements cannot be changed once created. (NCERT §5.7.3, p. 96–97)
- **Data types — None:** Special type with a single value; represents absence of value; neither `False` nor `0`. (NCERT §5.7.4, p. 97)
- **Data types — Mapping / Dictionary:** Unordered; holds key-value pairs enclosed in `{ }`; key and value separated by `:` colon; value accessed via `dict[key]`; keys are usually strings; only standard mapping type in Python. (NCERT §5.7.5, p. 97)
- **Mutable vs Immutable:** Mutable data types (List, Set, Dictionary) allow values to be changed after creation. Immutable types (Integer, Float, Boolean, Complex, String,

Tuple) do not; updating an immutable variable destroys the old object and creates a new one in memory. (NCERT §5.7.6, p. 97–99)

- **Choosing data types:** Lists for frequently modified collections; Tuples when data should not change; Sets for uniqueness/no duplicates; Dictionaries for key-value fast lookup. (NCERT §5.7.7, p. 99)
- **Operators — Arithmetic:** `+` (addition/concatenation), `-` (subtraction), `*` (multiplication/repetition), `/` (division — always returns float), `%` (modulus — remainder), `//` (floor/integer division — removes decimal), `**` (exponentiation). (NCERT §5.8.1, p. 100)
- **Operators — Relational:** `==`, `!=`, `>`, `<`, `>=`, `<=`; compare operands and return `True` or `False`; can also compare strings lexicographically using ASCII values. (NCERT §5.8.2, p. 100–101)
- **Operators — Assignment:** `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `//=`, `**=`. (NCERT §5.8.3, p. 101–102)
- **Operators — Logical:** `and`, `or`, `not` (must be lowercase). Default: all values are logically `True` except `None`, `False`, `0`, empty string `""`, empty list `[]`, empty tuple `()`, empty set `{}`. (NCERT §5.8.4, p. 103)
- **Operators — Identity:** `is` (True if both variables point to the same memory object), `is not` (True if they point to different objects). (NCERT §5.8.5, p. 103–104)
- **Operators — Membership:** `in` (True if value found in sequence), `not in` (True if not found). (NCERT §5.8.6, p. 104)
- **Operator precedence (high to low):** `**` > unary `~`, `+`, `-` > `*`, `/`, `%`, `//` > `+`, `-` > relational (`<=`, `<`, `>`, `>=`, `==`, `!=`) > assignment > `is`, `is not` > `in`, `not in` > `not` > `and` > `or`. Parentheses override precedence; equal-precedence operators evaluate left to right. (NCERT §5.9.1, p. 105)
- **Expressions and Statements:** An expression combines constants, variables, and operators and always evaluates to a value. A statement is a unit of code the interpreter can execute (e.g., assignment statement, print statement). (NCERT §5.9–5.10, p. 104–106)
- **Input and Output:** `input([Prompt])` — takes keyboard input and returns it as a **string** by default; use `int()` or `float()` to convert to numeric types. `print(value [, ..., sep=' ', end='\n'])` — outputs to screen; default separator is space; default end is newline. Using `+` between different types causes `TypeError`; use commas (`,`) to mix types in `print()`. (NCERT §5.11, p. 107–108)
- **Type conversion — Explicit (type casting):** Programmer forces conversion using functions: `int(x)`, `float(x)`, `str(x)`, `chr(x)` (ASCII value to character), `ord(x)` (character to ASCII code). Risk of data loss (e.g., `int(20.67)` discards `.67`). (NCERT §5.12.1, p. 109–110)

- **Type conversion — Implicit (coercion):** Python automatically converts data type when needed; uses **type promotion** to the wider type (e.g., int + float → float) to avoid loss of information. (NCERT §5.12.2, p. 112)
- **Debugging:** Process of identifying and removing errors (bugs) from a program. Three error types: (i) **Syntax errors** — violate Python grammar rules; interpreter stops and shows error message; must be fixed before execution; (ii) **Logical errors** — program runs without abrupt termination but produces wrong output; hardest to detect; also called semantic errors; (iii) **Runtime errors** — syntactically correct but interpreter cannot execute (e.g., division by zero, invalid type input); cause abnormal termination during execution. (NCERT §5.13, p. 112–114)

2.2 Definitions to memorise

Term	Definition	Page
Program	An ordered set of instructions executed by a computer to carry out a specific task	87
Programming language	Language used to specify a set of instructions to a computer	87
Source code	A program written in a high-level language	87
Interpreter	Translates and executes program statements one by one; stops at first error	87
Compiler	Translates entire source code into object code at once; generates all errors after scanning the whole program	87
Python shell	Another name for the Python interpreter; prompt symbol is `>>>`	88
Script mode	Writing Python code in a `.py` file and executing it via the interpreter	89
Keyword	A reserved word with a specific fixed meaning in Python; cannot be used as an identifier	90
Identifier	A user-defined name for a variable, function, or other entity in a program	91
Variable	A named memory location that stores a value; implicitly declared in Python when first assigned	91
Comment	A non-executable remark in source code starting with `#`; ignored by the interpreter	92
id()	Built-in function that returns the unique identity (memory address) of an object	93
Mutable	Data type whose value can be changed after creation (List, Set, Dictionary)	98
Immutable	Data type whose value cannot be changed after creation (int, float, bool, complex, str, tuple)	98

Term	Definition	Page
None	Special Python data type with a single value; represents absence of value	97
Floor division (`//`)	Divides and returns the integer quotient by removing the decimal part	100
Modulus (`%`)	Returns the remainder of division	100
Explicit conversion	Type conversion forced by the programmer using functions like `int()` , `float()` , `str()`	109
Implicit conversion	Automatic type conversion by Python (coercion); uses type promotion to avoid data loss	112
Debugging	Process of identifying and removing errors (bugs) from a program	112
Syntax error	Error caused by violation of Python grammar rules; detected before execution	113
Logical error	Bug that causes wrong output without abrupt termination; also called semantic error	113
Runtime error	Error that occurs during program execution, causing abnormal termination	113
Type promotion	Automatic widening of a narrower type to the wider type to avoid information loss (e.g., int + float → float)	112
Object identity	Unique identifier of a Python object equivalent to its memory address; returned by `id()`	93
Type casting	Explicit conversion of one data type to another using functions like `int()` , `float()` , `str()`	109
Tuple	Immutable ordered sequence enclosed in `( )` ; cannot be changed after creation	96
List	Mutable ordered sequence enclosed in `[ ]` ; supports item updates	95
Set	Unordered mutable collection enclosed in `{ }` ; no duplicate elements	96
Dictionary	Unordered mutable collection of key-value pairs enclosed in `{ }` with `:` between key and value	97
String	Immutable sequence of characters enclosed in single or double quotes	95
Interactive mode	Mode where a Python statement is typed at the `>>>` prompt and executed immediately	89
`chr()`	Built-in that converts an ASCII code to its character	110
`ord()`	Built-in that converts a character to its ASCII code	110

Term	Definition	Page
`type()`	Built-in that returns the class/data type of a value or variable	95

2.3 Diagrams / processes to remember

- **Figure 5.6 (p. 94) — Data types hierarchy in Python:** Root: Data Types in Python → Numbers (Integer, Floating Point, Complex, Boolean), Sequences (Strings, Lists, Tuples), Sets, None, Mappings (Dictionaries). Students must know which type falls under which category.
- **Figure 5.7 (p. 98) — Mutable vs Immutable classification:** Immutable side: Integers, Float, Boolean, Complex, Strings, Tuples. Mutable side: Lists, Sets, Dictionary. This diagram is a direct source for match-the-following MCQs.
- **Figures 5.8–5.10 (p. 98–99) — Object identity and memory:** When two variables hold the same value, they share the same memory object (same `id()`). When an immutable variable is updated, a new object is created at a new memory address — the old variable name is rebound.
- **Table 5.9 (p. 105) — Operator precedence table (11 levels):** Memorise top-5: `**` > unary > `* / % //` > `+ -` > relational. This table is the direct source for expression-evaluation questions.
- **Figure 5.1 (p. 88) — Python shell screen:** Shows `>>>` prompt. Useful for interactive mode vs script mode distinction questions.

2.4 Common confusions / NTA trap points

- `/` **always returns float in Python 3:** `8/4` gives `2.0`, not `2`. Students often expect an integer result. `//` (floor division) gives the integer quotient: `8//4` gives `2`, but `9//4` gives `2` (not `2.25`).
- `input()` **always returns a string:** Even if the user types `19`, `type(age)` returns `<class 'str'>`. NTA frequently uses programs where students must spot that arithmetic on an `input()` value will behave as string repetition/concatenation unless explicitly converted with `int()` or `float()`.
- `True` **and** `False` **are Python keywords and must be capitalised:** `true` and `false` are not keywords — they would be treated as identifiers. Since Python is case-sensitive, `NUMBER` and `number` are different identifiers.
- **Tuple vs List delimiter:** Tuples use `( )` and are immutable; Lists use `[ ]` and are mutable. NTA often swaps these in options. Sets use `{ }` but so do Dictionaries — the distinguishing feature is key-value pairs (`:`) for dictionaries.
- **Logical error does NOT terminate the program (NCERT §5.13.1, p. 113).** Unlike syntax and runtime errors, a logical error produces wrong output silently. NTA uses this in assertion-reason or statement-based questions where students must identify which error type allows the program to complete execution.

- `is` **vs** `==` **(NCERT §5.8.5, p. 103-104).** `==` compares values; `is` compares object identity (memory address). For small integers `5 is 5` is True but `[1,2] is [1,2]` is False because they are different list objects.
- **Empty containers are falsy (NCERT §5.8.4, p. 103).** `bool([])`, `bool("")`, `bool({})`, `bool(())`, `bool(0)`, `bool(None)` all return False. NTA uses these in logical-expression evaluation traps.
- `bool` **is a subtype of** `int` **(NCERT §5.7.1, p. 95).** `True + True == 2`. NTA distractor: "Boolean is unrelated to int" — false.
- **Indentation is syntactic in Python (NCERT §5.1.1, p. 88).** Wrong indentation produces a syntax error (IndentationError). NTA may suggest indentation is "merely stylistic" — false.
- **Identifiers cannot start with a digit (NCERT §5.3, p. 91).** `1var` is invalid; `var1` is valid. Underscore-start is allowed: `_var` is valid.
- **Strings cannot be combined with numbers using** `+` **(NCERT §5.11, p. 108).** `print("Age: " + 19)` raises `TypeError`; use commas or `str(19)`.
- `int(20.67)` **truncates, doesn't round (NCERT §5.12.1, p. 110).** Result is 20, not 21. NTA traps with rounding expectations.

🎯 Practice MCQs

Q1. Which of the following is a correct statement about the Python interpreter?

A. It translates the entire source code into object code before execution begins.

B. It translates and executes program statements one by one, stopping at the first error.

C. It generates all error messages only after scanning the complete program.

D. It directly converts Python source code into machine language without any intermediate step.

Q2. Consider the following Python code: ``` num1 = input("Enter a number: ") result = num1 * 2 print(result) ``` If the user enters `5`, what will be the output?

A. 10

B. 5.0

C. 55

D. Error: cannot multiply sequence by non-int

Q3. Which of the following lists correctly classifies Python data types as mutable and immutable?

A. Mutable: List, Tuple, Dictionary | Immutable: Integer, Float, String, Set

B. Mutable: List, Set, Dictionary | Immutable: Integer, Float, Boolean, Complex, String, Tuple

C. Mutable: List, Set, String | Immutable: Integer, Float, Tuple, Dictionary

D. Mutable: List, Dictionary, Tuple | Immutable: Integer, Float, Boolean, String, Set

🔒 **12 more MCQs + answer key**
Get UniDrill Pro · ₹199/year · unidrill.in/pricing

📊 PYQ Alignment

Python fundamentals are one of the highest-weightage topics in CUET Computer Science (subject code 308), consistently contributing 5–7 questions per year; NTA particularly favours output-prediction questions on operator precedence and `input()` type behaviour, data type classification (mutable/immutable), and error-type identification (syntax vs logical vs runtime). Use the PYQ archive for Computer Science for further drilling.