FREE SAMPLE | Buy the complete pack at notesbazar.in | page 1 of 19

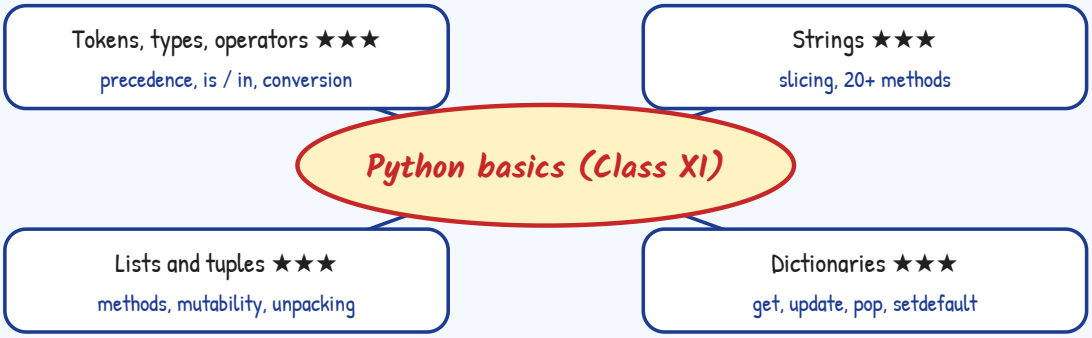
Chapter 1
Python Revision Tour

```
>>> print("Hi")
Hi
>>>
```

revise Class XI Python

Tokens · data types · operators · flow of control · strings · lists · tuples · dictionaries

Chapter map (★ = asked most often in the board exam)



Syllabus and weightage

Unit 1 **Computational Thinking and Programming – 2** carries **40 of 70 marks** and begins with a **revision of Class XI Python**. Revision topics are tested every year: about **10–13 one-mark questions** (outputs, valid/invalid statements, built-in methods) and 2-mark questions on dictionaries and strings.

Board paper	Items from this chapter
2024 (91, Set 4)	9 (Q3, 7, 8, 9, 10, 18, 22, 23, 26)
2025 (91, Set 4)	12 (Q1–8, 12, 23, 24)
2026 (91, Set 4)	13 (Q1–8, 20, 23, 24)

★ **Exam tip:** Most marks are lost on **output questions**: trace the code line by line on rough paper, write the exact separators (`sep`, `end`), quotes and brackets. A list prints with **[ ]**, a tuple with **( )**, a string without quotes.

In one line: Python stores values of different **types** (int, float, complex, bool, str, list, tuple, dict, None); **mutable** types (list, dict) can be changed in place, **immutable** ones (int, float, str, tuple) cannot; **operators** follow a fixed precedence; and **built-in methods** of strings, lists, tuples and dictionaries do most of the work in board programs.

1.1 Python basics ★★

Python is a **high-level, interpreted, dynamically typed, object-oriented** language. It is free and open source, platform independent, and uses **indentation** to mark blocks.

Execution modes

- **Interactive mode**: type a statement at the `>>>` prompt; it runs at once. Good for testing, but nothing is saved.
- **Script mode**: write a program in a `.py` file and run it (F5 in IDLE). The program is saved and can be run again.

Tokens (smallest units of a program)

Token	Meaning	Examples
Keyword	reserved word with a fixed meaning	`if, else, for, while, def, return, True, False, None, and, or, not, in, is, global, with, try, except, finally, import, break, continue, pass, lambda`
Identifier	name of a variable, function, class	`marks, _total, Break, sum2`
Literal	fixed value	`25, 3.14, 2+3j, "Hi", True, None`
Operator	performs an operation	`+ - * / // % ** == and in is`
Punctuator	symbol that organises code	`( ) [ ] { } , : . ; ' " #`

Rules for identifiers ★★★

- May contain letters, digits and underscore; **cannot start with a digit**.
- No spaces or special characters (`@ $ % -`).
- **Cannot be a keyword**; Python is **case sensitive**, so `Break` is valid but `break` is not.

Valid: `total_marks, _x, Sum2, Break`. Invalid: `2sum, my-name, for, total marks, global`.

Variables, l-value and r-value

A variable is a **name that refers to a value** stored in memory. In `x = y + 5`, **x** is the **l-value** (left side, must be a name) and **y + 5** is the **r-value** (an expression whose value is stored). `5 = x` is an error because a literal cannot be an l-value.

```
a = b = 10          # same value to many names
x, y = 3, 7         # multiple assignment
x, y = y, x         # swap: x is 7, y is 3
print(type(x), id(x))
```

Comments

Single-line comments start with `#`. A triple-quoted string (`'''...'''`) is used for multi-line notes and docstrings. Comments are ignored by the interpreter.

Input and output

```
name = input("Name: ")          # always returns a string
age  = int(input("Age: "))      # convert to int
print("Hi", name, sep="-", end="!\n")
print(1, 2, 3, sep="-", end="!\n")
```

Output

```
1-2-3!
```

`sep` is placed **between** values (default a space); `end` is printed **after** the last value (default a newline). Keyword arguments must come **after** positional ones, so `print("A", sep="*", 10)` is a **SyntaxError**.

Data types ★★★

Category	Types	Example
Number	int, float, complex	`74, 74.0, 3+4j`
Boolean	bool	`True, False`
Sequence	str, list, tuple	`"ab", [1,2], (1,2)`
Mapping	dict	`{"a":1}`
None	NoneType	`None` (absence of value)

> **Definition** ✎ **Mutable** objects can be changed in place after creation: **list, dict** (and set). **Immutable** objects cannot: **int, float, complex, bool, str, tuple**. "Changing" an immutable value actually creates a new object.

`type(74)` is `int` but `type(74.0)` is `float`: they are **different types** even though `74 == 74.0` is True.

1.2 Operators and expressions ★★★

Type	Operators	Example → result
Arithmetic	`+ - * / // % **`	`7/2→3.5, 7//2→3, 7%2→1, -7//2→-4, 2**10→1024`
Relational	`== != < > <= >=`	`5>3 → True`
Logical	`and or not`	`not 0 → True`
Assignment	`= += -= *= /= //= %= **=`	`x += 2` means `x = x + 2`
Identity	`is, is not`	True if both names refer to the **same object**
Membership	`in, not in`	`'a' in 'cat' → True`

Precedence (highest first) ★★★

1. `( )`
2. `**` (right to left: `2**3**2 = 2**9 = 512`)
3. unary `+x -x`

4. `* / // %` (left to right)
5. `+ -`
6. relational `== != < > <= >=`, `is`, `in`
7. `not`
8. `and`
9. `or`

`14 % 3**2 * 4` → `3**2` = 9 → 14 % 9 = 5 → 5 * 4 = **20**

`16*5/4*2/5-8` → 80/4 = 20.0 → ×2 = 40.0 → /5 = 8.0 → −8 = **0.0**

`/` always gives a **float**; `//` gives the **floor** (rounds towards −∞) and `%` takes the sign of the divisor: `-7//2 = -4`, `-7%2 = 1`.

Logical operators return an operand ★★

`and` returns the first false operand, otherwise the last one; `or` returns the first true operand, otherwise the last one.

```
print(5>3 and 7)    # 7
print(0 or "x")      # x
print(3 and 0)       # 0
print(None or 0)     # 0
```

So `3>1 and 2` evaluates to `2` (truthy), while `3>1 and 3>2` evaluates to `True`.

is versus ==

```
a = [1, 2]; b = [1, 2]
print(a == b, a is b)   # True False
c = a
print(c is a)           # True (same object)
```

Type conversion ★★

- **Implicit** (automatic): Python converts the smaller type in a mixed expression: `10 + 20.5 → 30.5` (int → float); `True + True → 2`.
- **Explicit** (type casting) by the programmer: `int("25")`, `float("2.5")`, `str(10)`, `int(9.99) → 9` (truncates), `list("abc") → ['a','b','c']`, `tuple([1,2])`, `bool("") → False`, `bool("0") → True`.

Errors ★★

Error	When	Example
Syntax error	rules of the language broken; program does not start	`print(2 ; 5)`, missing `:` or bracket
Run-time error (exception)	program starts but fails while running	`10/0` ZeroDivisionError, `"15"+3` TypeError, `int("a")` ValueError
Logical error	program runs but gives a wrong result	`avg = a + b / 2` instead of `(a+b)/2`

> ✗ **Don't do this:** Logical errors are **not** caught by `try…except`; only run-time errors (exceptions) are.

1.3 Flow of control ★★

Statements run **sequentially** unless a **conditional** (if) or **iterative** (for, while) statement changes the flow. A block is marked by the same **indentation** after a colon.

```
n = int(input("Number: "))
if n > 0:
    print("Positive")
elif n == 0:
    print("Zero")
else:
    print("Negative")
```

range(start, stop, step)

Generates start, start+step, … up to but **not including** stop. `range(5)` → 0 1 2 3 4; `range(2,11,3)` → 2 5 8; `range(5,0,-2)` → 5 3 1.

for and while

```
fact = 1                        # factorial
for i in range(1, 6):
    fact *= i
print(fact)                     # 120

n, rev = 1234, 0                # reverse digits
while n > 0:
    rev = rev*10 + n%10
    n //= 10
print(rev)                      # 4321
```

break, continue, pass, loop-else

- `break` leaves the loop at once; `continue` skips to the next iteration; `pass` does nothing (placeholder).
- The `else` of a loop runs only if the loop ends **without** `break`.

```
for n in range(2, 4):
    print(n, end=" ")
else:
    print("loop over")
```

Output

```
2 3 loop over
```

Nested loops: patterns ★

```
for i in range(1, 6):          # Pattern 1
    print("*" * i)
for i in range(5, 0, -1):      # 12345, 1234, ...
    for j in range(1, i+1):
        print(j, end="")
    print()
for i in range(1, 6):          # A, AB, ABC ...
    for j in range(i):
        print(chr(65+j), end="")
    print()
```

Standard programs (practical list) ★★

```
n = int(input("n: "))          # prime check
if n > 1:
    for d in range(2, int(n**0.5)+1):
        if n % d == 0:
            print("Composite"); break
    else:
        print("Prime")

a, b = 0, 1                    # Fibonacci, 8 terms
for _ in range(8):
    print(a, end=" "); a, b = b, a+b

def gcd(a, b):                 # GCD and LCM
    while b: a, b = b, a % b
    return a
x, y = 12, 18
print(gcd(x, y), x*y//gcd(x, y))   # 6 36

n = 153                        # Armstrong
s = sum(int(d)**len(str(n)) for d in str(n))
print(s == n)                  # True
n = 28                         # perfect number
print(sum(d for d in range(1, n) if n % d == 0) == n)  # True
```

Series $1 + x + x^2/2! + \ldots x^n/n!$:

```
x, n = 2, 4
s, term = 1, 1
for i in range(1, n+1):
    term = term * x / i        # x^i / i!
    s += term
print(round(s, 4))             # 7.0
```

1.4 Strings ★★★

A string is an **immutable sequence of characters** in single, double or triple quotes. Each character has a positive index (0 to len−1) and a negative index (−1 is the last).

char	C	o	m	p	u	t	e	r

+ve	0	1	2	3	4	5	6	7
-ve	-8	-7	-6	-5	-4	-3	-2	-1

Slicing s[start:stop:step] ★★★

```
s = "Computer Science"
s[3:8]    → 'puter'       s[:3]   → 'Com'
s[9:]     → 'Science'     s[::2]  → 'Cmue cec'
s[::-1]   → 'ecneicS retupmoC'
s[-7:-1]  → 'Scienc'
"PythonProgram"[-1:2:-2] → 'mroPo'
```

With a **negative step** the slice moves right to left, from start down to (but not including) stop. Slicing never gives an IndexError; indexing outside the range does.

Operations

Concatenation `"ab"+"cd" → 'abcd'`; repetition `"ab"*3 → 'ababab'`; membership `"th" in "Python" → True`; traversal `for ch in s:`. A string **cannot be changed in place**: `s[0] = "c"` → TypeError. `"15" + 3` → TypeError (str + int).

Traversal programs ★★

```
s = input("String: ")
v = c = up = lo = 0
for ch in s:
    if ch.isalpha():
        if ch in "aeiouAEIOU": v += 1
        else: c += 1
        if ch.isupper(): up += 1
        else: lo += 1
print(v, c, up, lo)
print("Palindrome" if s == s[::-1] else "Not palindrome")
print(s.swapcase())   # converts the case of each letter
```

Built-in functions and methods ★★★

Method	Example → result
`len(s)`	`len("Computer Science") → 16`
`capitalize()`	first letter upper, rest lower: `"the Truth" → 'The truth'`
`title()`	`"hello world" → 'Hello World'`
`lower(), upper()`	`'hello', 'HELLO'`
`count(sub)`	`"banana".count("a") → 3`
`find(sub)`	first index or **-1**: `"banana".find("na") → 2`, `find("x") → -1`
`index(sub)`	like find but raises **ValueError** if absent: `"Olympic2024".index("C")` → ValueError
`startswith(), endswith()`	`"data.csv".endswith(".csv") → True`
`isalnum()`	only letters/digits: `"abc123" → True`, `"abc 1" → False`
`isalpha(), isdigit()`	`"12.5".isdigit() → False`

`islower()`, `isupper()`, `isspace()`	`" ".isspace()` → `True`, `"".isspace()` → `False`
`lstrip()`, `rstrip()`, `strip()`	remove leading/trailing spaces (or given chars): `"**hi**".strip("*")` → `'hi'`
`replace(old,new)`	`"I like tea".replace("tea","coffee")`
`join(seq)`	`"-".join(["2026","03","15"])` → `'2026-03-15'`
`split(sep)`	returns a **list**: `"a,b,,c".split(",")` → `['a','b','','c']`; no argument splits on any whitespace
`partition(sep)`	returns a **3-tuple** (before, sep, after): `"name=Raj".partition("=")` → `('name','=','Raj')`; if sep absent → `('abc','','')`

★ ***Exam tip:*** Strings have no `sort()`, `append()` or `reverse()` methods: `"Hello".sort()` raises **AttributeError**. Use `sorted(s)` (returns a list) or slicing `s[::-1]`.

1.5 Lists ★★★

A list is a **mutable, ordered** sequence in `[ ]`; elements may be of **different types**: `[1, "a", 2.5, [3, 4]]`. Indexing and slicing work as for strings; `L[::-1]` gives a reversed copy.

```
L = [10, 20, 30]
L.append(40)         # [10, 20, 30, 40]
L.extend([50, 60])   # [10, 20, 30, 40, 50, 60]
L.insert(1, 15)      # [10, 15, 20, 30, 40, 50, 60]
L.pop()              # returns 60
L.pop(0)             # returns 10 → [15, 20, 30, 40, 50]
L.remove(30)         # removes first 30 → [15, 20, 40, 50]
L.reverse()          # [50, 40, 20, 15]
L.sort()             # [15, 20, 40, 50]
L.sort(reverse=True)  # [50, 40, 20, 15]
```

Function/method	*Note*
`append(x)` vs `extend(seq)`	`[1,2].append([3,4])` → `[1,2,[3,4]]`; `extend` → `[1,2,3,4]`
`pop(i)`	removes and **returns** item; IndexError if i out of range (`L.pop(6)` for a 6-item list)
`insert(i,x)`	never an error: i beyond the end just appends
`remove(x)`	ValueError if x absent
`index(x)`, `count(x)`	position of first x; number of x
`sort()` vs `sorted(L)`	sort() changes L and returns None; sorted() returns a **new** list, L unchanged
`min`, `max`, `sum`, `len`, `list()`	`sum([3,1,2])` → `6`
`del L[i]`, `del L[a:b]`	deletes by position

Aliasing and copying ★

```
A = [1, 2, 3]
B = A            # B is another name for the same list
B.append(4)
print(A)         # [1, 2, 3, 4]
C = A[:]         # copy (also list(A), A.copy())
C.append(5)
print(A, C)      # [1, 2, 3, 4] [1, 2, 3, 4, 5]
```

List programs ★★

```
L = [4, 9, 2, 9, 7]
print(max(L), min(L), sum(L)/len(L))    # 9 2 6.2
x = 9                                   # linear search
for i in range(len(L)):
    if L[i] == x:
        print("Found at", i); break
else:
    print("Not found")
freq = {}                               # frequency
for v in L: freq[v] = freq.get(v, 0) + 1
print(freq)          # {4: 1, 9: 2, 2: 1, 7: 1}
for i in range(0, len(L)-1, 2):         # swap even/odd positions
    L[i], L[i+1] = L[i+1], L[i]
print(L)             # [9, 4, 9, 2, 7]
```

1.6 Tuples ★★

A tuple is an **immutable, ordered** sequence in `( )`. A single-element tuple needs a **comma**: `(5)` is an int, `(5,)` is a tuple. Parentheses are optional: `t = 5, 6, 7`.

```
T = (1, 2, 3, 2)
T.count(2), T.index(3)     # 2, 2
T*2  → (1, 2, 3, 2, 1, 2, 3, 2)
T + (4,) → (1, 2, 3, 2, 4)
sorted((3, 1, 2)) → [1, 2, 3]   # a LIST
a, b, c = 5, 6, 7               # tuple assignment (unpacking)
t = tuple('tuple'); t += t[2],  # ('t','u','p','l','e','p')
```

`T[0] = 9` → TypeError. But `t += (x,)` works because it creates a **new** tuple and rebinds the name. Tuples are faster than lists and can be dictionary keys.

1.7 Dictionaries ★★★

A dictionary is a **mutable mapping** of **key : value** pairs in `{ }`. Keys must be **unique and immutable** (int, str, tuple); values can be anything. It is **not a sequence**: items are reached by key, not by position.

```
D = {"a": 1, "b": 2}
D["c"] = 3          # add  → {'a': 1, 'b': 2, 'c': 3}
D["a"] = 10         # modify (key exists)
for k in D:         # loop gives the KEYS
    print(k, D[k])
D.keys()   → dict_keys(['a', 'b', 'c'])
D.values() → dict_values([10, 2, 3])
D.items()  → dict_items([('a', 10), ('b', 2), ('c', 3)])
```

Method	Action → result
`get(k, default)`	value or None/default; no error: `D.get("z", 0)` → `0` (but `D["z"]` → KeyError)
`update(D2)`	adds/overwrites all pairs of D2
`pop(k)`	removes key k, returns its value
`popitem()`	removes and returns the last inserted pair
`setdefault(k, v)`	returns D[k] if present; else inserts k:v and returns v
`dict.fromkeys(seq, v)`	`dict.fromkeys(["x","y"], 0)` → `{'x': 0, 'y': 0}`
`copy()`	shallow copy (independent dict)
`del D[k]`, `clear()`	delete one pair / empty the dict
`len, max, min, sorted`	work on keys: `max({"p":3,"q":9})` → `'q'`
`dict()`	creates/returns a dictionary: `dict(a=1)`; `dict{}` is a SyntaxError

Dictionary programs ★★

```
s = "programming"            # count characters
C = {}
for ch in s:
    C[ch] = C.get(ch, 0) + 1
print(C)

emp = {}                     # employees and salary
n = int(input("How many? "))
for i in range(n):
    name = input("Name: ")
    emp[name] = float(input("Salary: "))
for k, v in emp.items():
    print(k, v)

S = {"AMIT":[92,86,64], "NAGMA":[65,42,43]}
for k, v in S.items():       # marks list as value
    avg = sum(v)/3
    print(k, "-", "A" if avg>=90 else "B" if avg>=60 else "C")
```

Checks: `"RNo" in D1` (key present?), `12 in D1.values()` (any value 12?).

1.8 Modules (revision) ★

```
import math
from statistics import mean, median, mode
math.sqrt(16) → 4.0        math.ceil(4.2) → 5
math.floor(-4.2) → -5      math.pow(2,3) → 8.0
math.fabs(-7) → 7.0        math.pi, math.e
mean([2,4,6,9]) → 5.25     median([3,1,4,2]) → 2.5
mode([1,2,2,3]) → 2
```

The random module (`random(), randint(a,b), randrange(a,b)`) is revised with board questions in Chapter 2, where "functions defined in modules" are taught.

CBSE PYQs 2024–2026 (fully solved) ★★★

CBSE 2025 State True or False: "A Python List must always contain all its elements of same data type."

> **Ans. False.** A list can hold elements of different types, e.g. `[1, "a", 2.5, [3]]`.

CBSE 2026 State True or False: In Python, data type of 74 is same as the data type of 74.0.

> **Ans. False.** `type(74)` is `int`; `type(74.0)` is `float`.

CBSE 2025 What will be the output of `print(14%3**2*4)`? (A) 16 (B) 64 (C) 20 (D) 256

> **Ans. (C) 20.** ** has the highest precedence: 3**2 = 9. Then % and * from left to right: 14 % 9 = 5; 5 * 4 = 20.

CBSE 2024 Output of `print(16*5/4*2/5-8)` : (a) –3.33 (b) 6.0 (c) 0.0 (d) –13.33

> **Ans. (c) 0.0.** Left to right: 16*5 = 80; 80/4 = 20.0; 20.0*2 = 40.0; 40.0/5 = 8.0; 8.0 – 8 = 0.0 (float, because / gives a float).

CBSE 2025 Identify the correct output: `game="Olympic2024"; print(game.index("C"))` (A) 0 (B) 6 (C) –1 (D) ValueError

> **Ans. (D) ValueError.** There is a lowercase "c" at index 6 but no uppercase "C"; `index()` raises ValueError when the substring is absent (`find()` would return –1).

CBSE 2025 Which is the correct identifier? (A) global (B) Break (C) def (D) with

> **Ans. (B) Break.** global, def and with are keywords. Python is case sensitive, so `Break` (capital B) is not the keyword `break`.

CBSE 2025 Identify the invalid statement: (A) print("A",10,end="*") (B) print("A",sep="*",10) (C) print("A",10,sep="*") (D) print("A"*10)

> **Ans. (B).** A positional argument (10) cannot follow a keyword argument (sep="*"); this is a SyntaxError.

CBSE 2025 `L=['TIC','TAC']; print(L[::-1])` (A) ['CIT','CAT'] (B) ['TIC','TAC'] (C) ['CAT','CIT'] (D) ['TAC','TIC']

> **Ans. (D) ['TAC', 'TIC'].** `[::-1]` reverses the order of the elements; the strings themselves are not reversed.

CBSE 2025 Which operator evaluates to True if the variables on either side point to the same memory location? (A) is (B) is not (C) and (D) or

Ans. (A) is, the identity operator (compares object identity, i.e. `id()`).

CBSE 2025 `D={'S01':95,'S02':96}`
`for I in D: print(I,end='#')` (A) S01#S02# (B) 95#96# (C) S01,95#S02,96# (D) S01#95#S02#96#

Ans. (A) S01#S02#. Looping over a dictionary gives its **keys**.

CBSE 2025 Which built-in function/method returns a dictionary? (A) dict() (B) keys() (C) values() (D) items()

Ans. (A) dict(). keys(), values() and items() return view objects (dict_keys, dict_values, dict_items), not dictionaries.

CBSE 2024 Identify the invalid statement: (a) d = dict() (b) e = {} (c) f = [] (d) g = dict{}

Ans. (d) g = dict{}. dict is a function and must be called with round brackets; `dict{}` is a SyntaxError.

CBSE 2024 `myStr = "MISSISSIPPI"; print(myStr[:4]+"#"+myStr[-5:])` (a) MISSI#SIPPI (b) MISS#SIPPI (c) MISS#IPPIS (d) MISSI#IPPIS

Ans. (b) MISS#SIPPI. `[:4]` = indices 0–3 = "MISS"; `[-5:]` = last five characters = "SIPPI".

CBSE 2024 Which statement will raise an error? (a) print("A"*3) (b) print(5*3) (c) print("15" + 3) (d) print("15" + "13")

Ans. (c). A str cannot be added to an int: TypeError. (a) gives AAA, (b) 15, (d) 1513.

CBSE 2024 `event="G20 Presidency@2023"; L=event.split(' '); print(L[::-2])` (a) 'G20' (b) ['Presidency@2023'] (c) ['G20'] (d) 'Presidency@2023'

Ans. (b) ['Presidency@2023']. L = ['G20', 'Presidency@2023']. `[::-2]` starts from the last element and moves two steps back, so only index 1 is taken; the result is a list.

CBSE 2024 A-R. (A): "Hello".sort() will give an error. (R): sort() does not exist as a method for strings.

Ans. (a) Both true and R explains A: strings are immutable and have no sort() method, so AttributeError is raised.

CBSE 2026 `s = "the Truth"; print(s.capitalize())` (A) The truth (B) THE TRUTH (C) The Truth (D) the Truth

Ans. (A) The truth. capitalize() makes the first character upper case and **all others lower case**.

CBSE 2026 Which expression evaluates to True? (A) 2>3 and 2<3 (B) 3>1 and 2 (C) 3>1 and 3>2 (D) 3>1 and 3<2

Ans. (C). True and True → True. (A) and (D) give False; (B) gives 2 (and returns its last operand), which is truthy but not the value True, so the scheme also accepted (B).

CBSE 2026 What is the output of the following code snippet? `s='War and Peace by Leo Tolstoy'; print(s.partition("by"))` (A) ('War and Peace ', 'by', ' Leo Tolstoy') (B) ['War and Peace ', 'by', ' Leo Tolstoy'] (C) ('War and Peace ', ' Leo Tolstoy') (D) ['War and Peace ', ' Leo Tolstoy']

Ans. (A) ('War and Peace ', 'by', ' Leo Tolstoy'). partition() returns a **tuple** of three parts: before the separator, the separator, after it (spaces kept).

CBSE 2026 What will be the output of `print("PythonProgram"[-1:2:-2])`? 1

Ans. Step −2 from index −1 (= 12, 'm') down to index 3 (excluded): indices 12, 10, 8, 6, 4 → 'm','r','o','P','o'. Output: **mroPo**.

CBSE 2026 What will be the output of the following code snippet? `t = tuple('tuple'); t2 = t[2],; t += t2; print(t)` (A) ('tuple') (B) ('tuple','p') (C) ('t','u','p','l','e','p') (D) ('t','u','p','l','e') 1

Ans. **(C) ('t', 'u', 'p', 'l', 'e', 'p').** tuple('tuple') splits the string into characters; `t[2],` is the one-element tuple ('p',); += creates a new, longer tuple.

CBSE 2026 True about dictionaries: (A) a sequence type (B) cannot have two elements with same key (C) cannot have two elements with same value (D) key and value cannot be the same 1

Ans. **(B).** Keys are unique (assigning to an existing key overwrites it). Values may repeat; a dictionary is a mapping, not a sequence.

CBSE 2026 If L has 6 elements, which raises an exception? (A) L.pop(1) (B) L.pop(6) (C) L.insert(1,6) (D) L.insert(6,1) 1

Ans. **(B) L.pop(6).** Valid indices are 0–5, so IndexError. insert() never fails: an index beyond the end simply appends.

CBSE 2026 A-R. (A): `[1,2,3]+'123'` is invalid. (R): a list cannot be concatenated with a string. 1

Ans. **(A)** Both true; R is the correct explanation (TypeError: can only concatenate list to list).

CBSE 2025 Write one example each of (i) Syntax Error (ii) Implicit type conversion. 2

Ans. (i) `print(2 ; 5)` or `if x>5 print(x)` (missing colon): the program does not run.
(ii) `x = 10; y = 20.5; print(x + y)` → 30.5: the int 10 is converted to float automatically.

CBSE 2025 D={"Suman":40, "Raj":55, "Raman":60}; D1={"Aditi":30, "Amit":90, "Raj":20} (built-in functions only).
(i)(a) Display the value for key "Raj" in D. OR (b) Display the length of D1.
(ii)(a) Append all pairs of D to D1. OR (b) Delete the item with key "Amit" from D1. 2

Ans. (i)(a) `print(D.get("Raj"))` or `print(D["Raj"])` → 55. (b) `print(len(D1))` → 3.
(ii)(a) `D1.update(D)`: D1 becomes {'Aditi': 30, 'Amit': 90, 'Raj': 55, 'Suman': 40, 'Raman': 60} (Raj's value is overwritten). (b) `D1.pop("Amit")` or `del D1["Amit"]`.

CBSE 2024 Output:
```
LS=["HIMALAYA","NILGIRI","ALASKA","ALPS"]; D={}
for S in LS: if len(S)%4==0: D[S]=len(S)
for K in D: print(K,D[K],sep="#")
```
2

Ans. Lengths: HIMALAYA 8, NILGIRI 7, ALASKA 6, ALPS 4. Only 8 and 4 are divisible by 4, so D = {'HIMALAYA': 8, 'ALPS': 4}.

Output
```
HIMALAYA#8
ALPS#4
```

CBSE 2024 (A) (i) Remove the item with key "NISHA" from dictionary Students. (ii) Display the number of occurrences of "is" in string message. OR (B) Convert tuple subject to a list and delete its last element. 2

Ans. (A)(i) `Students.pop("NISHA")` or `del Students["NISHA"]`. (ii) `print(message.count("is"))` → 3 for "This is his book" (This, is, his).

(B) `subject = list(subject)` then `subject.pop()` (or `del subject[-1]`).

CBSE 2024 Output: `S="Racecar Car Radar"; L=S.split()`

```
for W in L: x=W.upper()
  if x==x[::-1]: for I in x: print(I,end="*")
  else: for I in W: print(I,end="#")
  print()
```

3

Ans. "RACECAR" is a palindrome → printed in upper case with *. "CAR" is not → original "Car" with #. "RADAR" is a palindrome → *.

Output

```
R*A*C*E*C*A*R*
C#a#r#
R*A*D*A*R*
```

CBSE 2026 Write a statement (built-in functions/methods only): (i) to create a new list L1 with the elements of L in ascending order, without modifying L; (ii) to check whether the character ch is an alphabet or a number. 2

Ans. (i) `L1 = sorted(L)` (sorted returns a new list; `L.sort()` would change L). (ii) `ch.isalnum()` (or `ch.isalpha() or ch.isdigit()`).

CBSE 2026 Assuming D1 is a dictionary: (i)(a) Expression to check if key 'RNo' is present. OR (b) Check if any key in D1 has value 12. (ii)(a) A single built-in statement to add 'RNo':12 if 'RNo' is absent, but return its value if present. OR (b) Delete all elements of D1. 2

Ans. (i)(a) `'RNo' in D1` (b) `12 in D1.values()`
(ii)(a) `D1.setdefault('RNo', 12)` (b) `D1.clear()`

★ **Exam tip:** Board habit: for "built-in function/method only" questions, a loop gets no marks. Learn the one-line method for every common task (count, find, sorted, setdefault, update, pop, clear).

What the board asked from this chapter

Topic	2024	2025	2026
Operators, precedence	Q3	Q2, Q7	Q3
Identifiers, print, errors	Q9	Q4, Q5, Q23	Q1
Strings	Q8, Q10, Q18, Q26	Q3	Q2, Q4, Q5, Q23
Lists, tuples	Q23(B)	Q1, Q6	Q6, Q8, Q20
Dictionaries	Q7, Q22, Q23(A)	Q8, Q12, Q24	Q7, Q24

Practice set (answers at the end of the chapter)

A. Fill in the blanks (1 mark each)

1. The operator ____ returns the floor of a division.
2. `"banana".find("z")` returns ____.
3. A tuple with one element 7 is written as ____.

4. `print(2**3**2)` prints ____.
5. The method that removes and returns the last inserted pair of a dictionary is ____.
6. `"a-b-c".split("-")` returns ____.
7. Keys of a dictionary must be of ____ type.
8. `int(-3.9)` gives ____.

B. True or False (1 mark each)

1. A string can be modified in place.
2. `L.sort()` returns the sorted list.
3. `D.get("x")` raises KeyError if "x" is absent.
4. `[1,2] is [1,2]` is True.
5. `range(1,10,3)` generates 1, 4, 7.
6. The else of a loop runs even after break.
7. `"5"*3` gives "555".
8. `True + 2` gives 3.

C. Very short answer (1 mark each)

1. Name two mutable and two immutable types.
2. Output of `print("Hi", "All", sep="@", end="#")`?
3. Output of `print(-17 // 5, -17 % 5)`?
4. Difference between `find()` and `index()`.
5. Output of `"Hello World".title().swapcase()`?
6. Write a statement to sort list L in descending order.
7. Output of `print(len({"a":1,"b":2,"a":3}))`?
8. Why is `my-var` an invalid identifier?
9. What does `"abc".partition("b")` return?
10. Give an example of a logical error.

D. Short answer I (2 marks each)

1. Differentiate between `append()` and `extend()` with an example.
2. Differentiate between `sort()` and `sorted()`.
3. Explain implicit and explicit type conversion with one example each.
4. Predict: `L=[5,10,15]; M=L; M[0]=1; print(L)` and explain.
5. Write statements to (i) add key "Eng":90 to dict Marks, (ii) print all keys in sorted order.
6. Rewrite after removing errors: `for i in range(5)` / `if i%2=0:` / `print(i)`
7. What are l-value and r-value? Give an example.
8. Explain `is` vs `==` with an example.

E. Short answer II (3 marks each)

1. Write a program to count vowels, consonants, uppercase and lowercase letters in a string.
2. Write a program to input a list of numbers and swap elements at even and odd positions.

3. Write a program that reads n students' roll, name and marks into a dictionary and displays names with marks above 75.
4. Predict the output: `s="CBSE2026"; r=""` / `for ch in s:` / `if ch.isdigit(): r=ch+r` / `else: r=r+ch.lower()` / `print(r)`
5. Write a program to print the sum of the series $1 - x + x^2 - x^3 + \ldots \pm x^n$.
6. Explain mutable and immutable data types with two examples each.

F. Long answer (5 marks each)

1. Explain any five list methods with examples.
2. Explain any five dictionary methods with examples.
3. Write a menu-driven program for a dictionary of products and prices: add, search, update price, delete, display.
4. Explain the three types of errors in Python with examples.

G. HOTS

1. `t = ([1,2], 3); t[0].append(4); print(t)` : is this an error? Explain.
2. Why can a tuple be a dictionary key but a list cannot?
3. Output of `print([i*i for i in range(5) if i%2])` ?
4. Output of `D={}; D[(1,2)]=3; D[1,2]=4; print(D)` ?

Case study. A shop stores stock as `Stock = {"pen":[10, 25], "book":[5, 120], "bag":[2, 650]}` (item: [qty, price]). (i) Print the value of stock of "book". (ii) Add item "ink" with qty 20 and price 15. (iii) Print items whose quantity is below 5. (iv) Increase every price by 10%.

Multiple choice: 20 MCQs

1. Which is a valid identifier?
   (a) sum-2 (b) 2sum (c) _sum2 (d) for
2. Output of `print(7//2*2+7%2)` :
   (a) 7 (b) 6 (c) 8 (d) 7.0
3. Which is immutable?
   (a) list (b) dict (c) set (d) tuple
4. `type(3+4j)` is:
   (a) float (b) complex (c) str (d) int
5. Output of `"Python"[1:4]` :
   (a) 'Pyt' (b) 'yth' (c) 'ytho' (d) 'pyth'
6. `"apple".find("p")` returns:
   (a) 2 (b) 0 (c) -1 (d) 1
7. `[1,2,3].append([4])` makes the list:
   (a) [1,2,3,[4]] (b) [4,1,2,3] (c) [1,2,3,4] (d) error
8. Which returns None?
   (a) sorted(L) (b) len(L) (c) L.sort() (d) L.copy()
9. `bool("False")` is:
   (a) None (b) False (c) Error (d) True
10. Output of `print(10 > 5 or 1/0)` :
    (a) ZeroDivisionError (b) True (c) False (d) 1
11. `D={'a':1}; D.get('b',5)` gives:
    (a) KeyError (b) None (c) 5 (d) 1
12. `len("a\nb")` is:
    (a) 3 (b) 4 (c) 2 (d) 1
13. A one-element tuple:
    (a) (5) (b) (5,) (c) [5] (d) {5}
14. `max({"x":5,"y":2})` returns:
    (a) 5 (b) 'x' (c) 'y' (d) 2

15. `"a b c".split()` gives:
(a) ['a b c'] (b) ['a','b','','c'] (c) 'abc' (d) ['a','b','c']

16. Output of `print(2**-1)`:
(a) 0.5 (b) −2 (c) 0 (d) error

17. Which statement is a syntax error?
(a) 5 = x (b) x == 5 (c) x += 5 (d) x = 5

18. `"-".join("123")` gives:
(a) '123' (b) '-123' (c) '1-2-3-' (d) '1-2-3'

19. `T=(1,2,3); T[1]=5` gives:
(a) (1,5,3) (b) TypeError (c) (5,2,3) (d) IndexError

20. `list(range(10,0,-4))`:
(a) [10,6] (b) [10,6,2,−2] (c) [10,6,2] (d) []

1 (c) · 2 (a) · 3 (d) · 4 (b) · 5 (b) · 6 (d) · 7 (a) · 8 (c) · 9 (d) · 10 (b) · 11 (c) · 12 (a) · 13 (b) · 14 (c) · 15 (d) · 16 (a) · 17 (a) · 18 (d) · 19 (b) · 20 (c)

Assertion–Reason (6)

(a) Both A and R true, R explains A. (b) Both true, R does not explain A. (c) A true, R false. (d) A false, R true.

1. A: `s="abc"; s[0]="z"` raises an error. R: Strings are immutable.
2. A: `print(5/5)` displays 1. R: The / operator always returns a float.
3. A: A dictionary can have a tuple as a key. R: Keys of a dictionary must be immutable.
4. A: `L.sort()` and `sorted(L)` both return a sorted list. R: sorted() works on any iterable.
5. A: `3 and 7` evaluates to 7. R: `and` returns the last operand when all operands are true.
6. A: `"Hello".index("z")` returns −1. R: index() raises ValueError when the substring is not found.

Key: 1 (a) · 2 (d) · 3 (a) · 4 (d) · 5 (a) · 6 (d)

Answers to the practice set

A. 1. `//` 2. −1 3. `(7,)` 4. 512 5. `popitem()` 6. `['a','b','c']` 7. immutable 8. −3

B. 1. F 2. F (returns None) 3. F (returns None) 4. F (two different objects) 5. T 6. F 7. T 8. T

C. 1. Mutable: list, dict; immutable: str, tuple. 2. `Hi@All#` 3. `-4 3` (−17 = 5 × (−4) + 3) 4. find() returns −1 if absent; index() raises ValueError. 5. `hELLO wORLD` 6. `L.sort(reverse=True)` 7. 2 (the duplicate key "a" is overwritten) 8. "−" is an operator, not allowed in names. 9. `('a', 'b', 'c')` 10. `avg = a+b/2` instead of `(a+b)/2`.

D1. append(x) adds x as **one** element; extend(seq) adds each element of seq: `L=[1]; L.append([2,3])` → `[1, [2,3]]`; `L=[1]; L.extend([2,3])` → `[1,2,3]`.

D2. `L.sort()` sorts L in place and returns None; works only on lists. `sorted(x)` returns a new sorted list and leaves x unchanged; works on any iterable.

D3. Implicit: `5 + 2.0` → `7.0` (done by Python). Explicit: `int("42") + 8` → `50` (done by the programmer).

D4. `[1, 10, 15]`: M = L makes M an alias of the same list, so a change through M is seen through L.

D5. (i) `Marks["Eng"] = 90` (ii) `print(sorted(Marks))` or `for k in sorted(Marks): print(k)`.

D6.

```
for i in range(5):       # colon added
    if i%2 == 0:         # == for comparison
        print(i)         # indentation
```

D7. l-value: the name on the left of = that receives a value; r-value: the expression on the right whose value is assigned. In `total = price * qty`, total is the l-value and price*qty the r-value.

D8. `==` compares values; `is` compares identity. `a=[1]; b=[1]`: `a==b` True, `a is b` False.

E1. See the traversal program in section 1.4 (counts v, c, up, lo with isalpha/isupper).

E2.

```
L = eval(input("List: "))
for i in range(0, len(L)-1, 2):
    L[i], L[i+1] = L[i+1], L[i]
print(L)
```

E3.

```
n = int(input("Students: "))
R = {}
for i in range(n):
    roll = int(input("Roll: "))
    name = input("Name: ")
    marks = float(input("Marks: "))
    R[roll] = [name, marks]
for roll, (name, marks) in R.items():
    if marks > 75:
        print(name)
```

E4. Letters are added to the end in lower case; digits are added to the front: c, cb, cbs, cbse, then "2"+"cbse", "02cbse", "202cbse", "6202cbse". Output: `6202cbse`.

E5.

```
x = float(input("x: ")); n = int(input("n: "))
s = 0
for i in range(n+1):
    s += (-1)**i * x**i
print(s)
```

E6. Mutable: contents can change after creation without a new object, e.g. list (`L.append`), dict (`D[k]=v`). Immutable: any change creates a new object, e.g. str, tuple (`s[0]='x'` is an error).

F1–F2. Use the method tables of sections 1.5 and 1.7 (state the method, what it does, an example and its result for each).

F3.

```
P = {}
while True:
    ch = input("1 Add 2 Search 3 Update 4 Delete 5 Show 6 Exit: ")
    if ch == "1": P[input("Item: ")] = float(input("Price: "))
    elif ch == "2":
        k = input("Item: "); print(P.get(k, "Not found"))
    elif ch == "3":
        k = input("Item: ")
        if k in P: P[k] = float(input("New price: "))
    elif ch == "4": P.pop(input("Item: "), None)
    elif ch == "5":
        for k, v in P.items(): print(k, v)
    else: break
```

F4. Syntax, run-time and logical errors: see the table in section 1.2.

G1. No error: `([1, 2, 4], 3)`. The tuple still holds the same list object; the list itself is mutable. **G2.** Keys must be hashable (immutable); a list can change, so it cannot be a key. **G3.** `[1, 9]` (odd i: 1 and 3). **G4.** `{(1, 2): 4}`: `D[1,2]` is the same key (1, 2).

FREE SAMPLE | Buy the complete pack at notesbazar.in | page 19 of 19

Case. (i) `print(Stock["book"][0]*Stock["book"][1])` → 600 (ii) `Stock["ink"] = [20, 15]` (iii) `for k,v in Stock.items(): / if v[0] < 5: print(k)` → bag (iv) `for k in Stock: Stock[k][1] *= 1.10`

- **Last-night summary:** tokens (keyword, identifier, literal, operator, punctuator); identifier rules; mutable = list, dict; immutable = int, float, str, tuple; precedence ** (right to left) > unary > * / // % > + – > relational > not > and > or.
- Strings: slicing [start:stop:step], negative step reverses; find –1 vs index ValueError; capitalize vs title; split → list, partition → 3-tuple; no sort()/append() on str.
- Lists: append vs extend, pop(i) returns and may raise IndexError, insert never fails, sort() returns None vs sorted(); aliasing B = A.
- Tuples: (5,) for one element; immutable; unpacking a, b = b, a. Dictionaries: loop gives keys; get/update/pop/popitem/setdefault/fromkeys/clear; keys unique and immutable.