FREE EDITION · NOTES + 3 SAMPLE MCQS

CUET · COMPUTER SCIENCE · CLASS XI · CODE 308

Tuples and Dictionaries

CUET unit: Tuples and Dictionaries

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

WWW.UNIDRILL.IN

📌 Snapshot

- Two important Python data structures appear here: **tuples** (immutable ordered sequences) and **dictionaries** (mutable key-value mappings), both essential for efficient data storage and retrieval.
- Tuples are contrasted with lists: tuples are immutable (cannot be modified after creation), making them safer for fixed data; lists are mutable. Iterating through a tuple is faster than through a list.
- Dictionaries store data as key-value pairs enclosed in curly braces; keys must be unique and immutable (number, string, or tuple), while values can be of any type and can repeat.
- CUET tests these structures frequently — Python-specific data types, operations, built-in methods, and the distinction between mutable and immutable structures are all common MCQ territory.
- Practical programs (swapping variables, returning multiple values from functions, character frequency count) demonstrate real-world use of these structures and are a source of case/code-based questions.

📖 Detailed Notes

2.1 Core concepts

- **Tuple definition:** A tuple is an ordered sequence of elements of different data types (integer, float, string, list, or even another tuple). Elements are enclosed in parentheses and separated by commas. Indexing starts from 0, and negative indexing is supported. `(NCERT §10.1, p. 207)`
- **Single-element tuple:** If a tuple has only one element, that element must be followed by a comma — e.g., `(20,)`. Without the trailing comma, Python treats it as the data type of the element, not as a tuple. A sequence of comma-separated values without parentheses is also treated as a tuple by default. `(NCERT §10.1, p. 207-208)`
- **Accessing elements:** Elements of a tuple are accessed using indexing (`tuple1[0]`) and slicing (`tuple1[2:7]`), exactly like strings and lists. Accessing an index out of range raises `IndexError`. `(NCERT §10.1.1, p. 208)`
- **Tuple immutability:** Tuples are immutable — once created, individual elements cannot be changed. Attempting `tuple1[4] = 10` raises `TypeError: 'tuple' object`

`does not support item assignment`. However, if an element is itself a mutable type (e.g., a list), that element's contents can be modified. `(NCERT §10.1.2, p. 208-209)`

- **Concatenation (`+`):** Two tuples can be joined using the `+` operator to produce a new tuple. Extending an existing tuple via concatenation creates a new tuple rather than modifying the original. `(NCERT §10.2.1, p. 209)`
- **Repetition (`*`):** The `*` operator repeats the elements of a tuple a specified number of times. The first operand must be a tuple and the second must be an integer. `(NCERT §10.2.2, p. 210)`
- **Membership (`in` , `not in`):** The `in` operator returns `True` if an element is present in the tuple; `not in` returns `True` if it is absent. `(NCERT §10.2.3, p. 210)`
- **Slicing:** Slicing extracts a sub-tuple using `tuple[start:stop:step]`. Negative step reverses the tuple (e.g., `tuple1[::-1]`). `(NCERT §10.2.4, p. 210-211)`
- **Tuple methods and built-in functions:** Table 10.1 lists: `len()` , `tuple()` , `count()` , `index()` , `sorted()` , `min()` , `max()` , `sum()` . `count()` returns the frequency of a given element; `index()` returns the position of its first occurrence (raises `ValueError` if not found); `sorted()` returns a new sorted list without changing the original tuple. `(NCERT §10.3, p. 211-212)`
- **Tuple assignment (unpacking):** Python allows a tuple of variables on the left side of `=` to be assigned values from a tuple on the right. The number of variables must equal the number of elements, otherwise a `ValueError` is raised. Expressions on the right side are evaluated first. `(NCERT §10.4, p. 212-213)`
- **Nested tuples:** A tuple inside another tuple is called a nested tuple. Useful for storing structured records, e.g., student roll number, name, and marks. `(NCERT §10.5, p. 213)`
- **Tuple handling programs:** Tuples enable elegant operations such as swapping two variables without a temporary variable — `(num1, num2) = (num2, num1)` — and functions returning multiple values as a tuple. `(NCERT §10.6, p. 213-214)`
- **Introduction to dictionaries:** A dictionary is a mapping data type — a mapping between a set of keys and a set of values. Each key-value pair is called an item; key and value are separated by a colon (`:`); items are separated by commas and enclosed in curly braces. Items in a dictionary are ordered (Python 3.7+). `(NCERT §10.7, p. 215)`
- **Creating a dictionary:** Keys must be unique and immutable (number, string, or tuple). Values can be of any type and can repeat. An empty dictionary is created with `{}` or `dict()` . `(NCERT §10.7.1, p. 215-216)`
- **Accessing dictionary items:** Items are accessed by key (not by position). `dict1['key']` returns the corresponding value; accessing a non-existent key raises `KeyError` . `(NCERT §10.7.2, p. 216)`

- **Dictionaries are mutable:** New items can be added by assigning to a new key (`dict1['Meena'] = 78`). Existing items are modified by reassigning to an existing key. `(NCERT §10.8, p. 216-217)`
- **Dictionary operations:** The `in` operator checks if a key is present (`True` / `False`); `not in` does the opposite. Dictionary traversal using a `for` loop iterates over keys (Method 1) or key-value pairs using `.items()` (Method 2). `(NCERT §10.9-10.10, p. 217-218)`
- **Dictionary methods:** Table 10.2 lists: `len()` , `dict()` , `keys()` , `values()` , `items()` , `get()` , `update()` , `del` , `clear()` . `get()` returns the value for a key or `None` if the key is absent (no `KeyError`). `update()` merges another dictionary into the current one. `del dict1['key']` deletes a specific item; `del dict1` deletes the entire dictionary. `clear()` removes all items but the dictionary object remains. `(NCERT §10.11, p. 218-219)`
- **Manipulating dictionaries (programs):** Programs demonstrate: building a frequency-count dictionary of characters in a string (Program 10-7), converting digits to word-names using a dictionary (Python's idiomatic lookup-table pattern), and creating/operating on an odd-numbers dictionary (Program 10-5). `(NCERT §10.12, p. 219-222)`
- **Tuples are hashable (NCERT §10.7.1 cross-link).** Because tuples are immutable, they can act as dictionary keys or as members of a set. A tuple of all-immutable elements is fully hashable; a tuple containing a list is not hashable.
- **Iteration over a dictionary defaults to keys (NCERT §10.10, p. 218).** `for k in d:` yields each key. To iterate over values, use `for v in d.values()` ; for pairs, use `for k, v in d.items()` . The `items()` view always returns (key, value) tuples in insertion order in Python 3.7+.
- **Order preservation (NCERT note, p. 215).** From Python 3.7 onwards, dictionaries preserve insertion order. NCERT mentions this — earlier versions did not guarantee order. NTA distractors based on outdated information may claim dictionaries are "unordered" in modern Python — this is misleading.
- **Heterogeneous values (NCERT §10.7.1, p. 215).** A single dictionary may map keys to values of different types — e.g., `{'name': 'Akansha', 'marks': [98, 95, 99]}` mixes string and list values. This makes dictionaries versatile for structured record storage.
- **Common idiom: "Frequency counting".** NCERT Program 10-7 uses the pattern `d[ch] = d.get(ch, 0) + 1` — a Pythonic way to safely increment a possibly-missing key. Without `get()` an extra `if ch in d:` check would be needed.
- **The `len()` builtin works on tuples and dictionaries (NCERT §10.3, §10.11).** It counts elements in tuples and key-value pairs in dictionaries. Same builtin applies uniformly across sequence and mapping types.

- **Dictionaries can be nested (NCERT §10.7.1).** A value in a dictionary can itself be another dictionary — `{'alice': {'age': 16, 'class': 'XI'}, 'bob': {'age': 17, 'class': 'XII'}}`. Access uses chained brackets: `d['alice']['age']`.

2.2 Definitions to memorise

Term	Definition	Page
Tuple	An ordered, immutable sequence of elements of different data types, enclosed in parentheses separated by commas.	207
Immutable	A data type whose elements cannot be changed after creation.	208
Nested tuple	A tuple that contains another tuple as one of its elements.	213
Tuple assignment (unpacking)	Assigning values from a right-hand tuple to a matching number of variables on the left-hand side of the assignment operator.	212
Dictionary	A mapping (non-scalar) data type consisting of key-value pairs enclosed in curly braces; keys are unique and immutable.	215
Item (dictionary)	A key-value pair in a dictionary, with key and value separated by a colon.	215
KeyError	Error raised when a key that does not exist is accessed in a dictionary.	216
Mutable	A data type whose elements can be changed after creation (e.g., list, dictionary).	216
get()	Dictionary method that returns the value for a given key, or `None` if the key is absent, without raising `KeyError`.	219
update()	Dictionary method that appends key-value pairs from a second dictionary into the first.	219
clear()	Dictionary method that deletes all items from the dictionary, leaving an empty dictionary object.	219
`keys()`	Dictionary method returning a view of all keys	218
`values()`	Dictionary method returning a view of all values	218
`items()`	Dictionary method returning a view of all (key, value) tuple pairs	218
`del` keyword	Deletes a specific key (`del d['k']`) or the whole dictionary (`del d`)	219
`count()` (tuple)	Returns the number of times an element appears in the tuple	211
`index()` (tuple)	Returns the first index of an element; raises ValueError if absent	211
`min()`, `max()`, `sum()`	Built-ins returning smallest, largest, and sum of tuple elements	212
Mapping data type	Data type where each value is accessed via an immutable key (Python's dict is the standard example)	215

Term	Definition	Page
Single-element tuple	Tuple containing one element written with a trailing comma e.g., `(20,)`	207
Swap via tuple	Pythonic idiom `(a, b) = (b, a)` to exchange two variables without a temporary	213

2.3 Diagrams / processes to remember

- **Table 10.1 — Built-in functions and methods for tuples** (p. 211–212): A reference table covering `len()`, `tuple()`, `count()`, `index()`, `sorted()`, `min()`, `max()`, `sum()` with syntax and examples. Students should be able to predict output for each function given a sample tuple.
- **Table 10.2 — Built-in functions and methods for dictionary** (p. 218–219): A reference table covering `len()`, `dict()`, `keys()`, `values()`, `items()`, `get()`, `update()`, `del`, `clear()`. Particularly focus on the difference between `del dict1['key']` (removes one item), `del dict1` (removes the entire object), and `clear()` (empties the dictionary but object persists).
- **Program 10-1 — Nested tuple for student records** (p. 213): Shows how `st[i][0]`, `st[i][1]`, `st[i][2]` accesses roll number, name, and marks respectively from a nested tuple; useful model for 2D data access questions.
- **Program 10-3 — Function returning a tuple** (p. 214): A function `circle(r)` returns `(area, circumference)` as a tuple — demonstrates how Python functions can return multiple values.
- **Slicing summary** (p. 210–211): `tuple[start:stop]`, `tuple[:stop]`, `tuple[start:]`, `tuple[::step]`, `tuple[::-1]` (reverse). Understand that slicing on a tuple returns a tuple.

2.4 Common confusions / NTA trap points

- **Trailing comma for single-element tuple:** `(20)` is an integer; `(20,)` is a tuple. NTA may give `type((20))` and `type((20,))` as options — only the second is `<class 'tuple'>`.
- `sorted()` **returns a list, not a tuple:** `sorted(tuple1)` returns a new **list** in ascending order and does not modify the original tuple. A common distractor says it returns a sorted tuple or modifies the tuple in place.
- `in` **operator on dictionaries checks keys, not values:** `'Suhel' in dict1` is `True` if `'Suhel'` is a key. NTA may frame questions where students assume it checks values.
- `index()` **vs** `count()` **confusion:** `index()` returns the position of the first occurrence; `count()` returns the number of occurrences. Both raise errors/return 0 differently when the element is absent (`ValueError` for `index()`, `0` for `count()`).

- **Mutable element inside an immutable tuple:** Even though a tuple is immutable, if it contains a list, that list's elements can be changed. `tuple2[3][1] = 10` is valid if `tuple2[3]` is a list. NTA may test whether this raises an error.
- `del dict1` **vs** `dict1.clear()` **(NCERT Table 10.2, p. 219).** After `del dict1`, the name `dict1` no longer exists (`NameError` if referenced again). After `dict1.clear()`, `dict1` still exists as an empty dictionary `{}`.
- **Dictionary keys must be immutable (NCERT §10.7.1, p. 215).** Lists cannot be used as keys (they are mutable). Tuples and strings can. NTA distractor: claims any object can be a dictionary key.
- `get()` **returns None for missing keys (NCERT Table 10.2, p. 219).** Unlike `dict1['x']` which raises KeyError, `dict1.get('x')` silently returns None. NTA distractor swaps these behaviours.
- **Tuples allow nested mutable elements (NCERT §10.1.2, p. 209).** While the tuple's structure is immutable, mutable elements inside can be modified. NTA tests this nuance.
- **Tuple unpacking requires exact match (NCERT §10.4, p. 213).** `a, b = (1, 2, 3)` raises ValueError. NTA distractor: claims extras are silently ignored.
- **Concatenation creates a new tuple (NCERT §10.2.1, p. 209).** `t1 + t2` does not modify `t1` or `t2` — the result is a fresh tuple.

🎯 Practice MCQs

Q1. What will be the output of the following Python code? ```python >>> tuple5 = (20) >>> type(tuple5) ```

A. `<class 'tuple'>`

B. `<class 'list'>`

C. `<class 'int'>`

D. `<class 'str'>`

Q2. Consider the following tuple: `tuple1 = (10, 20, 30, 10, 40, 10, 50)`. What is the output of `tuple1.count(10)`?

A. 1

B. 2

C. 3

D. 0

Q3. Which of the following statements about tuples and dictionaries in Python is CORRECT?

A. Tuples are mutable and dictionaries are immutable.

B. Tuples use curly braces and dictionaries use parentheses.

C. Dictionary keys must be of immutable data types such as number, string, or tuple.

D. Dictionary values must be unique and of immutable types only.

📊 PYQ Alignment

Tuples and dictionaries are tested regularly in CUET Computer Science, typically contributing questions on tuple immutability versus list mutability, dictionary key constraints, output-prediction of built-in methods (`count()` , `index()` , `get()` , `keys()`), and code-trace questions involving nested tuples or tuple assignment. Questions are most often direct recall or short code-output based, making Table 10.1 and Table 10.2 the highest-priority revision items. See PYQ archive for Computer Science.