22/08/2025

Complex Data Type

- **Explanation:**
  The complex data type in Python is used to represent complex numbers, which have a real part and an imaginary part.
  Example: z = 3 + 4j → here, 3 is the real part, and 4j is the imaginary part.
- **Features:**
  - **Written as a + bj, where a = real, b = imaginary.**
  - **Real part accessed using .real, imaginary part using .imag.**
  - **Supports arithmetic operations (+, -, *, /).**
  - **Useful in scientific and mathematical computations.**

EX:1

Feature 1: Stores real and imaginary parts.

```
x = 3 + 4j
print(x.real, x.imag)   # Output: 3.0 4.0
```

Feature 2: Supports arithmetic operations.

```
a = 2 + 3j
b = 1 + 2j
print(a + b)   # Output: (3+5j)
```

Note : 'j' value is 1

---

2. List Data Type

MUTABLE DATA TYPE:

You can add, remove, or update elements in a mutable object.

The object stays in the same memory location, but its content is modified.

- **Explanation:**
  A list is an ordered, mutable (changeable) collection of items. Items can be of any data type.
  Example: my_list = [10, "apple", 3.5, True]
- **Features:**
  - **Ordered: Maintains insertion order.**
  - **Mutable: Items can be added, removed, or changed.**
  - **Heterogeneous: Can store different data types in one list.**
  - **Allows indexing & slicing (my_list[1], my_list[1:3]).**

- **Supports duplicate values.**

List declaration :

```
v=list()
print('data type:',type(v))
print('id:',id(v))
print(v)
```

ex 2:

```
j=[]
print('data type:',type(j))
print('id:',id(j))
print(j)
```

Feature 1: Ordered collection.

```
numbers = [10, 20, 30]
print(numbers[0])   # Output: 10
```

Feature 2: Mutable (can be changed).

```
fruits = ["apple", "banana"]
fruits[1] = "mango"
print(fruits)   # Output: ['apple', 'mango']
```

Feature 3: Allows duplicates.

```
nums = [1, 2, 2, 3]
print(nums)   # Output: [1, 2, 2, 3]
```

Feature 4: Supports different data types

```
mixed = [10, "apple", 3.5]
print(mixed)   # Output: [10, 'apple', 3.5]
```

---

3. Tuple Data Type

- **Explanation:**
  A tuple is an ordered, immutable (unchangeable) collection of items.
  Example: my_tuple = (10, "apple", 3.5, True)
- **Features:**
  - **Ordered: Maintains insertion order.**

- **Immutable: Once created, elements cannot be modified.**

- **Heterogeneous: Can store multiple data types.**
- **Supports indexing & slicing like lists.**
- **Can be used as keys in dictionaries (if elements are immutable).**

Declaration of tuple data type::

EX:1

```
import time
res=tuple()
print('data type:',type(res))
print('id:',id(res))
time.sleep(2)
print(res)
```

EX:2

```
import time
r=()
print(type(r))
print(id(r))
time.sleep(1.5)
print(r)
```

EX:3

```
res=([10,2,30,40],)
print(type(res))       # out put ➔ list data type
print(res)
```

Ex:4

```
res=(3.14)
print(type(res))   # out put ➔ float data type
```

Ex:5

```
import time
```

```
res=10,'python',3.1,True,3+2j


print(type(res))          # ➔ tuple data type
print('id:',id(res))
time.sleep(2)
print(res)
print(res[1])
```

Ex:6:

```
import time
v=10,True,200,'Developer',3e1
print('data type:',type(v))
print('id:',id(v))
time.sleep(1.5)
print('length:',len(v))
```

Ex:7

---

4. Set Data Type

- **Explanation:**
  A set is an unordered collection of unique elements.
  Example: my_set = {10, 20, 30, 10} → duplicates removed → {10, 20, 30}
- **Features:**
  - **Unordered: Does not maintain insertion order.**
  - **Unique elements only: Duplicates are automatically removed.**
  - **Mutable: You can add or remove elements, but items inside must be immutable.**
  - **Supports set operations like union, intersection, difference.**
  - **Does not support indexing or slicing.**

```
myset = {10, 20, 30}
# Using set() constructor
print(s)      # Output: {1, 2, 3}
```

Note: Empty set must be declared using set(), not {}.

```
empty = set()   #  correct
```

empty2 = {} # creates an empty dictionary, not a set

Feature 1: Unordered collection

```
s = {10, 20, 30}
print(s)   # Output may vary in order, e.g., {20, 10, 30}
```

Feature 2: Does not allow duplicates

```
s = {1, 2, 2, 3}
print(s)   # Output: {1, 2, 3}
```

Feature 3: Mutable (can be changed)

```
s = {10, 20}
s.add(30)   # Adding new element
print(s)    # Output: {10, 20, 30}

s.remove(20)   # Removing element
print(s)       # Output: {10, 30}
```

Feature 4: Can store heterogeneous data types

```
s = {10, "apple", 3.14}
print(s)   # Output: {10, 3.14, 'apple'}
```

invalid code

res={10,'python',3.12,[500,600,800]} ➔ **ERROR**

```
print("data type",type(res))
print(res)
```

NOTE:

Allowed: int, float, str, tuple, bool

Not allowed: list, set, dict (because they are mutable)