Data Structures

Python Course · Module 3

ElnurBDa

Data Structures – Overview

Python provides multiple built-in structures for organizing data:

- **Lists**
- **Strings**
- **Tuples**
- **Sets**
- **Dictionaries**
- Choosing the right structure

Each structure has unique properties and use-cases.

Lists

Lists store **ordered, changeable** sequences.

```
numbers = [10, 20, 30]

# Operations:
numbers.append(40)
numbers[1] = 99
numbers.remove(10)
print(numbers)

# List indexing:
print(numbers[0])
print(numbers[-1])
print(numbers[1:3])
```

———————— [finished] ————————

```
[99, 30, 40]
99
40
[30, 40]
```

List Comprehensions

List comprehensions are a **compact way** to build new lists from existing data.

Basic pattern:

```
numbers = [1, 2, 3, 4, 5]

squares = [n * n for n in numbers]
evens = [n for n in numbers if n % 2 == 0]

print("squares:", squares)
print("evens:", evens)
```

———————————— [finished] ————————————

```
squares: [1, 4, 9, 16, 25]
evens: [2, 4]
```

They are great for:

- simple transformations
- filtering values
- avoiding manual `for` + `.append()` boilerplate

Rule of thumb: if the logic fits in **one short line**, a comprehension is OK; otherwise, use a normal loop for readability.

List Characteristics

Advantages:

- Mutable
- Dynamic size
- Supports duplicates

Use-cases:

- Collections
- Queues
- Storing ordered items

Strings

Strings represent **text**.

Properties:

- Ordered
- Immutable
- Indexable
- Iterable

```
s = "Hello"

a = s[0]       # first character
b = s[-1]      # last character
c = s.upper()  # all upper chars
d = len(s)     # Length of the string

print(a, b, c, d)
```

```
──────────────────── [finished] ────────────────────
```

```
H o HELLO 5
```

Tuples

Tuples store **ordered, immutable** sequences.

```
point = (10, 20)
```

Benefits:

- Fast
- Secure from modification
- Memory-efficient

Tuple unpacking:

```
x, y = point
```

Tuple vs List

- Need modification → **List**
- Need fixed, reliable structure → **Tuple**
- Performance required → **Tuple**

Sets

Sets store **unordered, unique** elements.

```
colors = {"red", "blue", "green"}
```

Operations:

```
colors.add("yellow")
colors.remove("red")
```

Useful for:

- membership checks
- removing duplicates
- mathematical operations

Set Operations

```
a = {1, 2, 3}
b = {3, 4, 5}

x1 = a | b    # union
x2 = a & b    # intersection
x3 = a - b    # difference
x4 = a ^ b    # symmetric difference

print(x1, x2, x3, x4)
```

———————— [finished] ————————

```
{1, 2, 3, 4, 5} {3} {1, 2} {1, 2, 4, 5}
```

Dictionaries

Dictionaries store **key : value** pairs.

```
user = {
    "name": "Alice",
    "age": 25,
    "city": "Baku"
}

# Access:
n = user["name"]
print(n)
print()

# Modify:
user["age"] = 26
user["country"] = "Azerbaijan"
print(user)
```

———————————— [finished] ————————————

```
Alice

{'name': 'Alice', 'age': 26, 'city': 'Baku', 'country':
'Azerbaijan'}
```

Dictionary Operations

```
user = {
    "name": "Alice",
    "age": 25,
    "city": "Baku"
}

a = user.keys()
b = user.values()
c = user.items()
d = "city" in user  # check key existence
e = "Baku" in user

print(a)
print(b)
print(c)
print(d)
print(e)
```

```
———————————————— [finished] ————————————————
```

```
dict_keys(['name', 'age', 'city'])
dict_values(['Alice', 25, 'Baku'])
dict_items([('name', 'Alice'), ('age', 25), ('city',
'Baku')])
True
False
```

Dictionaries are ideal for structured data.

Example

```
student = {
    "name": "Elnur",
    "scores": [90, 85, 97],
    "passed": True
}

average = sum(student["scores"]) / len(student["scores"])

print(student["name"], "average:", average)
```

———————————————— [finished] ————————————————

```
Elnur average: 90.66666666666667
```

Choosing the Right Data Structure

Type	Ordered	Mutable	Unique	Best For
String	Yes	No	N/A	Text
List	Yes	Yes	No	General collections
Tuple	Yes	No	No	Fixed data
Set	No	Yes	Yes	Unique items
Dict	Keys no	Yes	Keys unique	Structured data

Mini Task

Create a script that:

1. Stores student info in a dictionary
2. Contains: name, age, list of grades
3. Calculates average grade
4. Prints a formatted summary
