Datatypes

Category	Types
Numbers	int, float, complex
Text	str
Boolean	bool
Collections	list, tuple, set, frozenset, dict
Binary	bytes, bytearray, memoryview
Special	NoneType, custom classes

Datatypes

In **Python**, you can use a variety of **data types** to work with different kinds of data.

Basic Built-in Data Types

Type	Description	Example
int	Integer numbers	5, -20, 0
float	Decimal numbers (floating-point)	3.14, -0.99, 2.0
bool	Boolean (True/False)	True, False
str	String (text)	"hello", 'Python'

Collection Data Types

Type	Description	Example
list	Ordered, changeable collection	[1, 2, 3], ['apple', 'banana']
tuple	Ordered, **unchangeable** collection	(1, 2, 3), ('a', 'b')
set	**Unordered**, no duplicates	{1, 2, 3}, {'red', 'blue'}
dict	Key-value pairs	{'name': 'Alice', 'age': 25}

Datatypes

None Types

Type	Description	Example
NoneType	Represents the absence of a value	None

Examples

```
age = 30                                # int
pi = 3.14159                            # float
is_happy = True                         # bool
name = "Python"                         # str
colors = ['red', 'green']               # list
coordinates = (10.0, 20.0)              # tuple
unique_ids = {1, 2, 3}                  # set
profile = {'name': 'Alice', 'age': 25}  # dict
no_value = None                         # NoneType
```

List(list)

A **list** is an **ordered**, **changeable (mutable)** collection of items.

- You can add, remove, change items.

- Lists allow **duplicates**.

- Defined with **square brackets** [].

- Example

    fruits = ['apple', 'banana', 'cherry']

    print(fruits[0])   # apple

Lists are **indexed** — first item is at index 0.

Common List Methods:

Method	Description	Example
append(item)	Add to end	fruits.append('orange')
insert(index, item)	Insert at position	fruits.insert(1, 'kiwi')
remove(item)	Remove first occurrence	fruits.remove('banana')
pop(index)	Remove by index	fruits.pop(1)
sort()	Sort list	fruits.sort()
reverse()	Reverse list	fruits.reverse()
len(list)	Number of items	len(fruits)

Tuple (tuple)

A **tuple** is an **ordered**, **unchangeable (immutable)** collection.

- Once created, you **cannot modify** it.
- Defined with **parentheses** ().

```
colors = ('red', 'green', 'blue')
print(colors[1])    # green
```

Tuples are **indexed** like lists.

Key Differences from List

List	Tuple
Mutable (changeable)	Immutable (fixed)
Slower	Faster
Use when items may change	Use when data should not change

Set(set)

A **set** is an **unordered**, **changeable**, and **no-duplicates allowed** collection.

- Defined with **curly braces** {}.

```
numbers = {1, 2, 3, 2, 1}
print(numbers)
```

Output

    {1, 2, 3}

Duplicates are **automatically removed**.

Common Set Methods

Method	Description	Example
add(item)	Add an item	numbers.add(4)
remove(item)	Remove an item	numbers.remove(2)
union(set)	Combine two sets	set1.union(set2)
intersection(set)	Common items	set1.intersection(set2)
difference(set)	Items in first but not second	set1.difference(set2)

Dictionary (dict)

A **dictionary** is an **unordered** collection of **key-value pairs**.

- Each item has a **key** and a **value**.

- Defined with **curly braces** {} and a colon :.

```
student = {
    'name': 'Alice',
    'age': 23,
    'course': 'Python'
}
print(student['name'])   # Alice
```

Keys are **unique** and used to **access values**.

Common Dictionary Methods

Method	Description	Example
keys()	Return all keys	student.keys()
values()	Return all values	student.values()
items()	Return all key-value pairs	student.items()
get(key)	Get value safely	student.get('age')
update(dict)	Update dictionary	student.update({'grade': 'A'})
pop(key)	Remove a key	student.pop('age')

Quick Real-Life Analogy

Type	Real World Example
List	A shopping list (order matters, can add/remove items)
Tuple	Your birthdate (fixed, never changes)
Set	Collection of unique lottery numbers (no duplicates)
Dictionary	A phonebook (name → phone number mapping)

Big Summary Table

Type	Ordered?	Changeable?	Allows Duplicates?	Example
List ([])	Yes	Yes	Yes	['a', 'b', 'c']
Tuple (())	Yes	No	Yes	('a', 'b', 'c')
Set ({})	No	Yes	No	{1, 2, 3}
Dictionary ({})	No	Yes	Keys: No, Values: Yes	{'name': 'Alice'}

Important Notes:

- Lists and tuples are **ordered** (you can access by index).

- Sets and dictionaries are **unordered** (you can't depend on order).

- Dictionaries map **keys** to **values**.

- Sets automatically **remove duplicates**.