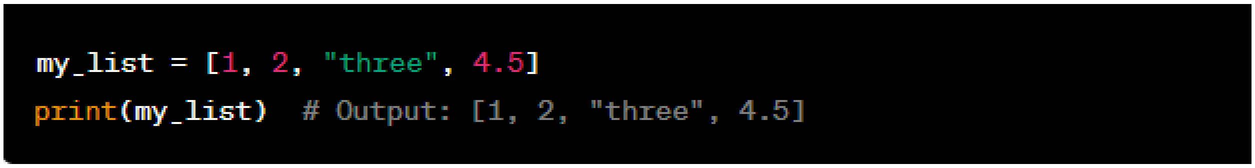
Data Structures

There are four built in data structures in python.

Lists:

Lists are ordered, mutable sequences that can hold elements of different data types. You can add, remove, or modify elements in a list. Lists are defined using square brackets ([]).



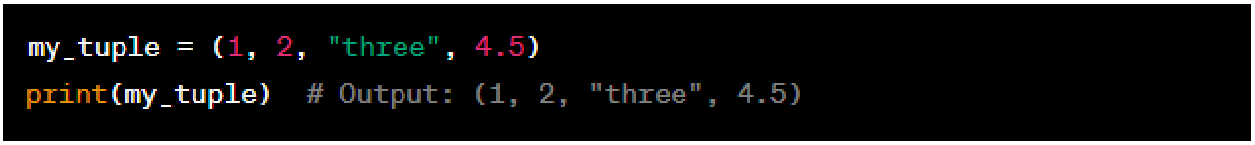
```
my_list = [1, 2, "three", 4.5]
print(my_list)  # Output: [1, 2, "three", 4.5]
```

Lists are mutable, which means they can be altered.

Tuples:

Tuples are ordered, immutable sequences that are similar to lists but cannot be modified once created. Tuples are defined using parentheses (()).

Example:

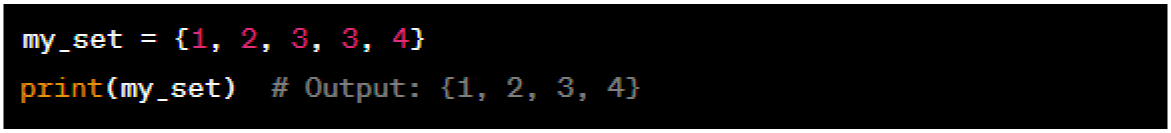


```
my_tuple = (1, 2, "three", 4.5)
print(my_tuple)  # Output: (1, 2, "three", 4.5)
```

Sets:

Sets are unordered collections of unique elements. They do not allow duplicate values. Sets are defined using curly braces ({}) or the **set()** function.

Example:

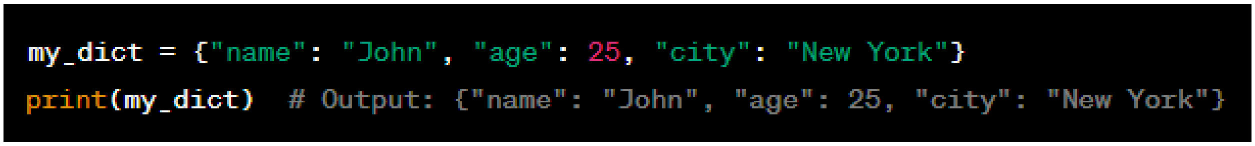


```
my_set = {1, 2, 3, 3, 4}
print(my_set)  # Output: {1, 2, 3, 4}
```

Dictionaries:

Dictionaries are key-value pairs where each value is associated with a unique key. They are unordered and mutable. Dictionaries are defined using curly braces ({}) and colons (:).

Example:



```
my_dict = {"name": "John", "age": 25, "city": "New York"}
print(my_dict)  # Output: {"name": "John", "age": 25, "city": "New York"}
```

These data structures serve different purposes, and choosing the right one depends on the requirements of your program. Lists are useful for storing ordered collections of elements. Tuples are suitable for situations where immutability is desired. Sets are handy when you need to store unique values or perform set operations like union, intersection, etc. Dictionaries are efficient for mapping keys to values, making it easy to retrieve values based on their keys.