Python Basics Manual- Session 2

Objective

This manual is designed to help learners understand the different sequence data types in Python. It provides a detailed overview, common functions, and practical examples to strengthen understanding and application.

Table of Contents

1. Sequence Data Types

In Python, **sequence data types** represent ordered collections of items. These items can be accessed using indices, sliced, and looped through.

There are three Sequence data types in the Python programming language:

- **String**: A string is a collection of one or more characters put in a single, double, or triple quote. In python there is no character data type, a character is a string of length one.
- **List**: List is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple**: Tuple is a collection which is ordered and unchangeable. Allows duplicate members.

2. String Data Type

A string is a collection of one or more characters put in a single quote, double-quote, or triple-quote. In python there is no character data type, a character is a string of length one. It is represented by str class.

Strings Manipulation

Strings are sequences of characters and are one of the most commonly used data types in Python.

Creating a String

- **Usage**: Define a string using single, double, or triple quotes.
- **Example**:

```
my_string = "Hello, World!"
another_string = 'Hello, Python!'
multiline_string = """This is a
                multiline string."""
```

String Methods:

Upper and Lower Case

- **Usage**: Converts a string to uppercase or lowercase.
- **Example**:

```
upper_string = my_string.upper()  # result is "HELLO, WORLD!"
lower_string = my_string.lower()  # result is "hello, world!"
```

Strip

- **Usage**: Removes whitespace from the beginning and end of a string.
- **Example**:

```
stripped_string = "   Hello, World!   ".strip()

# result is "Hello, World!"
```

Split

- **Usage**: Splits a string into a list of substrings based on a specified delimiter.
- **Example**:

```
split_string = my_string.split(",")
# result is ['Hello', ' World!']
```

Join

- **Usage**: Joins elements of a list into a single string, separated by a specified delimiter.
- **Example**:

```
joined_string = "-".join(["Hello", "World"])
# result is "Hello-World"
```

Find

- **Usage**: Returns the index of the first occurrence of a substring.
- **Example**:

```
index = my_string.find("World")  # result is 7
```

Replace

- **Usage**: Replaces a substring with another substring.
- **Example**:

```
replaced_string = my_string.replace("World", "Python")
# result is "Hello, Python!"
```

3. List

A list is an ordered, mutable collection of items. Lists are just like arrays, which is an ordered collection of data. It is very flexible as the items in a list do not need to be of the same type. Lists are defined using square brackets [].

Examples:

```
my_list = [1, 2, 3, 4, 5]
print(my_list[0])      # 1
my_list.append(6)      # Adds 6 to the end of the list
print(my_list)         # [1, 2, 3, 4, 5, 6]
my_list[2] = 10        # Changes the third element to 10
print(my_list)         # [1, 2, 10, 4, 5, 6]
```

Python has a set of built-in methods that you can use on lists/arrays.

Method	*Description*
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

4. Tuples

Tuple is a sequence of immutable Python objects. Tuples are just like lists with the exception that tuples cannot be changed once declared. Tuples are usually faster than lists. Tuples are defined using parentheses ().

Examples:

```
my_tuple = (1, 2, 3, 4, 5)
print(my_tuple[0])      # 1
# my_tuple[2] = 10    # This would raise an error since tuples
are immutable
print(my_tuple)         # (1, 2, 3, 4, 5)
```

- **Tuples are Ordered:**

When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.

- **Tuples are Unchangeable:**

Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.

Note: To create a tuple with only one item, you have to add a comma after the item, otherwise Python will not recognize it as a tuple.

5. Comparison between sequence data types

Feature	String	List	Tuple
Syntax	'Hello'	[1, 2, 3]	(1, 2, 3)
Ordered	Yes	Yes	Yes
Mutable	No	Yes	No
Indexing	Yes	Yes	Yes
Slicing	Yes	Yes	Yes
Use Case	Text data	Dynamic collections	Fixed data groups

6. Summary

- **Strings** are used for text and are immutable.
- **Lists** are versatile, mutable collections, ideal for dynamic data.
- **Tuples** are immutable and best for fixed-size grouped data.

Each type has unique strengths depending on the use case. Mastery of these three forms the foundation for working with Python collections effectively.