

# Chapter 3

## Chapter 3: Strings in Python – Theory Notes

### ✓ Introduction to Strings

```
name = "Prathamesh"
nameshort = name[0:7]
print("The short version of my name is", nameshort)

character1 = name[1]
print("The second letter of my name is", character1)
```

### 📖 Key Concepts:

- Strings in Python are **ordered sequences** of characters.
- They are **zero-indexed**, meaning the first character is at index `0`.
- `name[0:7]` includes characters from **index 0 to 6** (7 is excluded).
- `name[1]` gives the **second character** of the string.

🧠 **Learning:** Python's string slicing uses the format `string[start:end]`

It includes the start index but **excludes the end index**.

## Negative Indexing

```
name = "Prathamesh"
print(name[-10:-1]) # Output: rathames
print(name[0:9])    # Output: Prathames

print(name[:4])      # Same as name[0:4] → "Prat"
print(name[1:])      # Same as name[1:10] → "rathamesh"
print(name[3:])      # Same as name[3:10] → "thamesh"
```

### Key Concepts:

- Negative indexes start from the **end of the string**.
  - `1` is the **last character**, `2` is second last, and so on.
- `name[-10:-1]` is same as `name[0:9]` → This is helpful for **reverse-style slicing**.
- Python allows **partial slicing**:
  - `[:4]` → from start till index 3
  - `[3:]` → from index 3 to end

## Slicing with Skip Values

```
word = "amazing"
print(word[1:7:2]) # Output: mzn
```

### Key Concepts:

- Format: `string[start:end:step]`
- It selects characters from `start` to `end-1` **with a step (skip value)**.
- Example:
  - `word[1:7]` → `"mazing"`
  - Then it skips 1 character each time → `"m"`, `"z"`, `"n"`

### Formula:

string[start:end:step] ⇒ take characters from start to end (not including end) and skip step-1 characters

## String Functions

```
name = "prathamesh"
print(len(name))           # 10
print(name.endswith("esh")) # True
print(name.startswith("Pr")) # False (case-sensitive)
print(name.capitalize())   # Prathamesh

quote = "Prathamesh is a great person"
print(quote.replace("great", "awesome")) # Replaces all "great" with "awesome"
```

### Useful Built-in String Methods:

Function	Description
<code>len(s)</code>	Returns the length of the string
<code>s.endswith("str")</code>	Checks if string ends with given substring
<code>s.startswith("str")</code>	Checks if string starts with given substring (case-sensitive)
<code>s.capitalize()</code>	Capitalizes <b>first character</b> and makes rest lowercase
<code>s.replace(a, b)</code>	Replaces all instances of <code>a</code> with <code>b</code>


 **Note:** Strings in Python are **immutable** – functions like `replace()` return a new string.

## Escape Sequences in Strings

```
a = "Prathamesh is a good person\nbut he is more of a\n \"Great Person\""
print(a)
```

### Explanation of Escape Sequences:

Escape Code	Meaning	Example Output
<code>\n</code>	New line	Moves the text to next line
<code>\"</code>	Double quote	Allows " inside strings
<code>\'</code>	Single quote	Allows ' inside strings
<code>\\</code>	Backslash	Outputs a <code>\</code> character
<code>\t</code>	Horizontal tab	Adds a tab space
<code>\b</code>	Backspace	Removes previous character

 **Learning:** Use escape sequences to format your strings with **newlines**, **quotes**, or **spacing**.

## Chapter 3: Strings – Practice Problems & Notes

### Problem 1 – Greeting the User

```
name = input("Enter your Name: ")
print("Good Afternoon,", name)

# Modern and cleaner way using f-string
print(f"Good afternoon, {name}")
```

 **Learning:**

- `input()` takes string input by default.
- `f"..."` is an **f-string** → allows **inserting variables directly** inside strings.
- Easier and cleaner than using `+` or `,` for string concatenation.

### Problem 2 – Selection Letter Generator

```
name = input("Enter your Name: ")
date = input("Enter Today's Date: ")

letter = f'''Dear {name},
You are selected!
```

```
{date}'''  
print(letter)
```

### Learning:

- Multiline string with triple quotes `'''...'''` is great for letter formats or paragraphs.
- `f'''...'''` supports variables even across multiple lines.
- Perfect use case: emails, auto-generated messages, certificates.

## Problem 3 – Detect Double Spaces

```
value = input("Enter Anything: ")  
print(value.count(" "))
```

### Learning:

- `.count(" ")` counts **double spaces** in the string.
- Useful in cleaning text data or debugging bad formatting.

## Problem 4 – Find First Single Space

```
value = input("Enter Anything: ")  
print("The number of single spaces is", value.find(" "))
```

### Learning:

- `.find(" ")` returns the **index of the first space** found.
- If no space is found, it returns `-1`.
- Strings are **immutable** → you can't change them directly; you return a new one.

## Problem 5 – Format Letter with Escape Sequences

```
letter = "Dear Harry,\nThis python course is nice, \nThanks!"  
print(letter)
```

## Learning:

- `\n` → newline character. Adds structure to printed text.
- Makes console output easier to read and more professional.



## Chapter 3 – Summary Table

Concept	Description
<b>Indexing</b>	Strings are indexed from 0. Use <code>string[i]</code> to access characters.
<b>Slicing</b>	Use <code>string[start:end]</code> to get part of the string (end not included).
<b>Negative Indexing</b>	Use <code>-1, -2, ...</code> to access from the end of the string.
<b>Skip Values</b>	Use <code>string[start:end:step]</code> to skip characters while slicing.
<b>Common Methods</b>	<code>.find()</code> , <code>.count()</code> , <code>.replace()</code> , <code>.capitalize()</code> , <code>.startswith()</code> etc.
<b>String Length</b>	<code>len(string)</code> returns the total number of characters.
<b>String Immutability</b>	Strings cannot be changed in-place. Functions return new modified strings.
<b>f-strings</b>	Use <code>f"Hello {name}"</code> to format strings cleanly and efficiently.
<b>Escape Sequences</b>	<code>\n</code> , <code>\t</code> , <code>\\</code> , <code>\"</code> , etc. for formatting and special characters.
<b>Multiline Strings</b>	Use triple quotes <code>'''text'''</code> for multi-line strings.