Python Programming: Strings in Depth

Comprehensive Study Material for Class 11

Subject: Computer Science
Target Revision: Final Exams

Table of Contents

Introduction to Strings

Welcome to the world of Python programming! As a Class 11 student, you have already learned about variables and basic data types. Today, we dive deep into one of the most powerful and frequently used data types: **Strings**.

1. What is a String?

In Python, a string is a sequence of characters. These characters can be letters, numbers, symbols, or even spaces. Essentially, anything you type inside quotes is treated as a string.

```
my_string = "Hello, Class 11!"
print(my_string)
```

2. Creating Strings

Python offers great flexibility in how we define strings. You can use single, double, or triple quotes.

- **Single Quotes (' '):** Used for short, single-line strings.
- **Double Quotes (" "):** Works exactly like single quotes, but useful if the string contains a single quote character.

- **Triple Quotes (''' ''' or """ """):** Used for multi-line strings or docstrings.

```
msg1 = 'Hello'
msg2 = "Python's power"
msg3 = """This is a
multi-line string"""
```

Section Summary: *Strings are sequences of characters enclosed in quotes. Triple quotes allow content to span multiple lines.*

Indexing and Slicing

1. String Indexing

Every character in a string has a specific position called an **index**. Python uses two types of indexing:

- **Positive Indexing:** Starts from 0 (left to right).
- **Negative Indexing:** Starts from -1 (right to left).

Character	P	Y	T	H	O	N
Pos Index	0	1	2	3	4	5
Neg Index	-6	-5	-4	-3	-2	-1

2. String Slicing

Slicing allows you to extract a part (substring) of a string using the syntax:

`string[start : stop : step]`

```
word = "COMPUTER"
print(word[0:4])    # Output: COMP
print(word[4:])     # Output: UTER
print(word[::-1])   # Output: RETUPMOC (Reverse)
```

Section Summary: *Indexing helps access single characters, while slicing extracts parts of strings. Indexing starts at 0.*

String Operations and Methods

1. Basic Operations

Python provides simple operators to manipulate strings:

- **Concatenation (+):** Joins two strings together.
- **Repetition (*):** Repeats a string a specified number of times.
- **Membership (in / not in):** Checks if a character or substring exists within a string.

```
a = "Python"
b = "is Great"
print(a + " " + b)  # Python is Great
print(a * 3)        # PythonPythonPython
print("Py" in a)    # True
```

2. Essential String Methods

Methods are built-in functions that work on strings. Here are the most common ones for Class 11:

- **upper() / lower():** Converts to uppercase or lowercase.
- **title():** Capitalizes the first letter of every word.

- **strip():** Removes leading and trailing whitespace.
- **find(substring):** Returns the lowest index of substring (or -1 if not found).

- **replace(old, new):** Replaces all occurrences of 'old' with 'new'.
- **count(char):** Counts how many times a character appears.
- **split():** Splits string into a list based on a delimiter.
- **join():** Joins elements of a list into a single string.

```
text = "  hello python  "
print(text.strip().upper()) # "HELLO PYTHON"
print(text.count('o'))       # 2
```

Section Summary: *Concatenation joins strings, while repetition repeats them. Methods like upper(), strip(), and split() are vital for text processing.*

Formatting, Iteration, and Immutability

1. Escape Sequences

Escape sequences allow us to include special characters in strings using a backslash (\).

- \n : Newline
- \t : Horizontal Tab
- \\ : Backslash
- \' : Single Quote
- \" : Double Quote

2. String Formatting (f-strings)

Modern Python uses f-strings for clean and readable formatting. Simply prefix the string with 'f' and use curly braces for variables.

```
name = "Aryan"
score = 95
print(f"Student {name} scored {score} marks.")
```

3. String Immutability

Crucial Concept: Strings in Python are *immutable*. This means once a string is created, its characters cannot be changed in place.

```
s = "Hello"
# s[0] = 'J'  # This will trigger a TypeError

# To 'change' it, create a new string:
s = "J" + s[1:]
print(s) # Jello
```

4. Iterating through Strings

```
word = "PYTHON"
for char in word:
    print(char, end="-")
# Output: P-Y-T-H-O-N-
```

Section Summary: *f-strings are the preferred way to format output. Immutability means strings cannot be modified after creation.*

Practical Programs for Practice

1. Reverse a String

```
s = input("Enter a string: ")
print("Reversed:", s[::-1])
```

2. Check Palindrome

```
s = input("Enter a word: ")
if s == s[::-1]:
    print("It is a palindrome")


else:
    print("Not a palindrome")
```

3. Count Vowels in a String

```
s = input("Enter string: ")
count = 0
for char in s.lower():
    if char in "aeiou":
        count += 1
print("Total vowels:", count)
```

4. Count Occurrences of a Character

```
s = "programming is fun"
char_to_find = 'r'
print(s.count(char_to_find))
```

5. Toggle Case (Upper to Lower and vice versa)

```
s = "PyThOn"
print(s.swapcase())
```

6. Remove All Spaces

```
s = "  Remove   Spaces  "
print(s.replace(" ", ""))
```

7. Find Substring

```
main_str = "Class 11 Python"
sub_str = "Python"
if sub_str in main_str:
    print("Found!")
```

8. Extract domain from an Email

```
email = "student@school.com"
domain = email.split('@')[1]
print("Domain:", domain)
```

Exam Oriented Questions & Final Summary

Multiple Choice Questions (MCQs)

- 1. What is the output of 'abc' * 2? (a) abc2 (b) abcabc (c) aabbcc (d) Error
- 2. Which method removes whitespace from both ends? (a) clean() (b) trim() (c) strip() (d) split()
- 3. Are Python strings mutable? (a) Yes (b) No (c) Only triple-quoted ones (d) Depends on OS

Short Answer Questions

- Q1: Explain the difference between find() and index().
- Q2: What is negative indexing? Give an example.
- Q3: Why is string formatting important?

Long Answer Questions

- Q1: Write a Python program to count the number of words in a sentence provided by the user.
- Q2: Explain any five string methods with examples.

- Q3: Elaborate on string slicing with the help of a diagram and code snippet.

Final Revision Notes

- Strings are sequences of Unicode characters.
- Indexing starts at 0 for positive and -1 for negative.
- Slicing [start:stop:step] excludes the 'stop' index.
- Common operators: +, *, in, not in.
- Strings are immutable; you cannot change a character at a specific index.
- f-strings are more efficient and readable for formatting.