Chapter 9 - 4String

Theory:

This chapter covers the topic of 4String.

Code Example:

```
# ----------------------------------------
# ■ STRINGS IN PYTHON
# ----------------------------------------
# A string is a sequence of characters enclosed in quotes.
# You can use single (' '), double (" ") or triple (''' ''' or """ """) quotes.
name = 'milan' # Single quotes
Name = "milan" # Double quotes
nAME = '''MILAN
MILAN''' # Triple quotes (for multiline strings)
# ■ Strings are IMMUTABLE in Python
# → Once created, characters inside a string cannot be changed.
# ----------------------------------------
# ■ LENGTH OF A STRING
# ----------------------------------------
nameshort = len(name)
print("Length of 'name':", nameshort) # Output: 5
# ----------------------------------------
# ✂■ STRING SLICING
# ----------------------------------------
# Syntax: string[start_index : end_index]
# - Includes start_index
# - Excludes end_index
print(name[0:3]) # Output: 'mil'
print(name[1:3]) # Output: 'il'
# Full slice:
nameshort = name[0:5]
print("Full sliced name:", nameshort) # Output: 'milan'
# ----------------------------------------
# ■ CHARACTER ACCESS
# ----------------------------------------
char1 = name[1]
print("Character at index 1:", char1) # Output: 'i'
# ----------------------------------------
# ■ INDEXING TYPES
# ----------------------------------------
# Positive Indexing:
# m i l a n
# 0 1 2 3 4
# Negative Indexing:
# m i l a n
# -5 -4 -3 -2 -1
print(name[-1]) # Output: 'n'
print(name[-3:-1]) # Output: 'la'
# Negative slicing explained:
print(name[-4:-1]) # Output: 'ila' → Same as name[1:4]
# Convert negative to positive:
# name[-4:-1] = name[1:4]
# ----------------------------------------
```

```
# ■■ SLICING SHORTCUTS
# ----------------------------------------
print(name[:4]) # Output: 'mila' → name[0:4]
print(name[1:]) # Output: 'ilan' → name[1:len(name)]
print(name[1:5]) # Output: 'ilan'
# ----------------------------------------
# ■ SLICING WITH STEP VALUE
# ----------------------------------------
word = "amazing"
print(word[1:6:2]) # Output: 'mzn'
# Explanation: From index 1 to 5, skipping every 2nd character
# ----------------------------------------
# ■ COMMON STRING METHODS
# ----------------------------------------
name = "milan"
print(len(name)) # Output: 5
print(name.endswith("an")) # Output: True
print(name.endswith("lan")) # Output: True
print(name.startswith("lan")) # Output: False
print(name.startswith("mil")) # Output: True
name = "milan shgarma"
print(name.capitalize()) # Output: 'Milan shgarma'
# ----------------------------------------
# ■ STRING METHOD REFERENCE (with examples)
# ----------------------------------------
# ■ 1. len()
text = "hello"
print(len(text)) # Output: 5
# ■ 2. str()
number = 123
print(str(number)) # Output: '123'
# ■ 3. lower()
print("HeLLo".lower()) # Output: 'hello'
# ■ 4. upper()
print("hello".upper()) # Output: 'HELLO'
# ■ 5. capitalize()
print("python programming".capitalize()) # Output: 'Python programming'
# ■ 6. title()
print("hello world".title()) # Output: 'Hello World'
# ■ 7. strip()
print(" hello ".strip()) # Output: 'hello'
# ■ 8. replace()
print("banana".replace("a", "o")) # Output: 'bonono'
# ■ 9. find()
print("hello".find("l")) # Output: 2
# ■ 10. index()
print("hello".index("e")) # Output: 1
# print("hello".index("z")) # Error!
# ■ 11. count()
print("banana".count("a")) # Output: 3
# ■ 12. startswith()
print("hello world".startswith("hello")) # Output: True
# ■ 13. endswith()
```

```
print("hello world".endswith("world")) # Output: True
# ■ 14. 'in' keyword
print("th" in "python") # Output: True
# ■ 15. isalnum()
print("abc123".isalnum()) # Output: True
# ■ 16. isalpha()
print("hello".isalpha()) # Output: True
# ■ 17. isdigit()
print("12345".isdigit()) # Output: True
# ■ 18. islower()
print("hello".islower()) # Output: True
# ■ 19. isupper()
print("HELLO".isupper()) # Output: True
# ■ 20. isspace()
print(" \t\n".isspace()) # Output: True
# ■ 21. split()
print("a,b,c".split(",")) # Output: ['a', 'b', 'c']
# ■ 22. join()
words = ["I", "love", "Python"]
print(" ".join(words)) # Output: 'I love Python'
# ■ 23. format()
name = "Milan"
print("Hello, {}".format(name)) # Output: 'Hello, Milan'
# ■ 24. f-string (Python 3.6+)
age = 25
print(f"I am {age} years old") # Output: 'I am 25 years old'
# ----------------------------------------
# ■ ESCAPE SEQUENCE CHARACTERS IN PYTHON
# ----------------------------------------
# Escape sequences are special characters starting with a backslash (\)
# They're used to insert characters that are otherwise hard to include directly in strings.
# ■ Common Escape Sequences:
# \n → New line
# \t → Tab space
# \' → Single quote
# \" → Double quote
# \\ → Backslash
# ----------------------------------------
# ■ Examples
# ----------------------------------------
# \n → Adds a new line
a = "Milan is a good boy\nBut not a bad boy"
print(a)
# Output:
# Milan is a good boy
# But not a bad boy
# \t → Adds a horizontal tab
a = "Milan is a good boy\tBut not a bad boy"
print(a)
# Output: Milan is a good boy But not a bad boy
# \" → Allows double quotes inside double-quoted string
a = "Milan is a good \"boy\"\nBut not a bad boy"
print(a)
```

```
# Output:
# Milan is a good "boy"
# But not a bad boy
# \' → Allows single quotes inside double-quoted string
a = "Milan is a good \'boy\'\nBut not a bad boy"
print(a)
# Output:
# Milan is a good 'boy'
# But not a bad boy
# ■ If using single quotes to define the string,
# and also want to use a single quote inside → you MUST escape it:
# This will cause an error: 'Milan is a bad boy, 'yes'' ■
# Correct version:
correct = 'Milan is a bad boy, \'yes\''
print(correct)
# \\ → Prints a single backslash
a = "Milan is a good boy\\But not a bad boy"
print(a)
# Output: Milan is a good boy\But not a bad boy
```