String Data Type

Premanand S

Assistant Professor
School of Electronics Engineering
Vellore Institute of Technology
Chennai Campus *premanand.s@vit.ac.in*

October 4, 2024

Introduction to Strings

- Strings represent sequences of characters.
- Strings are immutable (cannot be modified once created).
- Strings can be created using:
  - Single quotes: —'Hello'—
  - Double quotes: —"World"—
  - Triple quotes: —"'This is a multi-line string"'—
- We can convert numbers in a string into a number using int().

String Immutability

Example (String)

```
Strings are immutable.
You can create new strings but cannot modify existing ones.

s = "Hello"
s[0] = 'h'  # This raises an error
```

String Indexing and Slicing

Example (Indexing and Slicing)

```
Indexing starts from 0, negative indexing starts from
the end.

s = "Hello"
print(s[0])    # Output: H
print(s[-1])   # Output: o
print(s[1:4])  # Output: ell
```

Looking Inside Strings

Example (Indexing)

We can access any single character in a string using its index

```
>>> fruit = 'banana'
>>> letter = fruit[1]
>>> print(letter)  # a
```

A Character Too Far

Example (IndexError Example)

```
>>> zot = 'abc'
>>> print(zot[5])
Traceback (most recent call last):
  File "<stdin>" line 1 in <module>
IndexError: string index out of range
```

String Slicing

Example (Slicing Example)

We can access continuous sections of a string using slicing.

```
>>> s = 'Monty Python'
>>> print(s[0:4])  # Mont
>>> print(s[6:20])  # Python
```

String Concatenation and Repetition

Example (Concatenation and Repetition)

```
# Concatenate strings using the + operator.
# Repeat strings using the * operator.
s1 = "Hello"
s2 = "World"
print(s1 + " " + s2)  # Output: Hello World

>>> str3 = '123'
>>> str3 = str3 + 1
Traceback (most recent call last):
  File "<stdin>" line 1 in <module>
TypeError: cannot concatenate 'str' and 'int' objects

print(s1 * 3)         # Output: HelloHelloHello
```

Reading and Converting Strings

Example (Conversion Example)

```
>>> apple = input('Enter:')
Enter: 100
>>> x = int(apple) - 10
>>> print(x)  # 90
```

Strings Have Length

Example (Length of a String)

```
>>> fruit = 'banana'
>>> print(len(fruit))  # 6
```

Looping Through Strings (While Loop)

Example (While Loop Example)

```
fruit = 'banana'
index = 0
while index < len(fruit):
    letter = fruit[index]
    print(index, letter)
    index = index + 1
```

Looping Through Strings (For Loop)

Example (For Loop Example)

```
fruit = 'banana'
for letter in fruit:
    print(letter)
```

String Comparison

Example (Comparison Example)

```
word = input('Enter a word:')

if word == 'banana':
    print('All right bananas.')
elif word < 'banana':
    print('Your word ' + word + ' comes before banana.')
else:
    print('Your word ' + word + ' comes after banana.')
```

String Formatting

Example (String Formatting)

```
# Using format() method and f-strings for formatting:
name = "Premanand"
age = 37
print("My name is {} and I am {} years old".format(name, age))
print(f"My name is {name} and I am {age} years old")
```

Escape characters

- Escape characters in Python are used to represent special characters.
- They are preceded by a backslash ‘\‘ to escape their special meaning.

Common Escape Characters

Example (Python)

```
\'  : Single quote
\"  : Double quote
\\  : Backslash
\n  : Newline
\t  : Horizontal tab
\r  : Carriage return
\b  : Backspace
\f  : Form feed
\v  : Vertical tab
\a  : Bell (alert sound)
\0  : Null character
```

Octal and Hexadecimal Escape Characters

Example (Python)

```
# \ooo : Octal value
print("\101")

# \xhh : Hexadecimal value
print("\x41")
```

Unicode Escape Characters

Example (Python)

```
# \uXXXX : 16-bit Unicode character
print("\u2764")

# \UXXXXXXXX : 32-bit Unicode character
print("\U0001F600")

# \N{name} : Named Unicode character
print("\N{grinning face}")
```

Whitespace and Control Characters

Example (Python)

```
# Newline and Tab
print("Hello\nWorld!")

print("Hello\tWorld!")

# Carriage return
print("Hello\rWorld!")

# Form feed and vertical tab
print("Hello\fWorld!")
print("Hello\vWorld!")
```

Backspace

Example (Python)

```
print("He said, \"Python is awesome!\"")

path = "C:\\Users\\SPNK\\Documents"

path = r"C:\Users\JohnDoe\Documents"

# Backspace
print("Hello\bWorld!")
```

Exploring String Methods

Example (Type and dir() Example)

```
>>> stuff = 'Hello world'
>>> type(stuff)
<class 'str'>

>>> dir(stuff)
['capitalize', 'casefold', 'center', 'count', 'encode',
 'endswith', 'expandtabs', 'find', 'format', 'format_map',
 'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit',
 'isidentifier', 'islower', 'isnumeric', 'isprintable',
 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower',
 'lstrip', 'maketrans', 'partition', 'replace', 'rfind',
 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
 'splitlines', 'startswith', 'strip', 'swapcase', 'title',
 'translate', 'upper', 'zfill']
```

lower()

Example (Lowercase Example)

Converts all characters of the string to lowercase.

```
>>> s = "Hello World"
>>> print(s.lower())  # hello world
```

upper()

Example (Uppercase Example)

Converts all characters of the string to uppercase.

```
>>> s = "Hello World"
>>> print(s.upper())  # HELLO WORLD
```

capitalize()

Example (Capitalize Example)

Capitalizes the first character of the string.

```
>>> s = "hello world"
>>> print(s.capitalize())  # Hello world
```

title()

Example (Title Case Example)

Converts the first character of each word to uppercase.

```
>>> s = "hello world"
>>> print(s.title())  # Hello World
```

strip()

Example (Strip Example)

Removes leading and trailing whitespaces.

```
>>> s = "   hello world   "
>>> print(s.strip())  # "hello world"
```

replace(old, new)

Example (Replace Example)

Replaces occurrences of a substring with another substring.

```
>>> s = "I hate Programming"
>>> print(s.replace("hate", "love"))  # I love Programming
```

find(substring)

Example (Find Example)

```
Returns the lowest index where the substring is found, or
-1 if not found.

>>> s = "Hello World"
>>> print(s.find("World"))  # 6
>>> print(s.find("Python"))  # -1
```

count(substring)

Example (Count Example)

Returns the number of non-overlapping occurrences of a substring.

```
>>> s = "banana"
>>> print(s.count('a'))  # 3
```

startswith(substring) and endswith(substring)

Example (Startswith and Endswith Example)

Checks if the string starts or ends with the specified substring.

```
>>> s = "Hello World"
>>> print(s.startswith("Hello"))  # True
>>> print(s.endswith("World"))  # True
```

split(separator)

Example (Split Example)

Splits the string into a list of substrings based on a separator.

```
>>> s = "Hello,World,Python"
>>> print(s.split(","))  # ['Hello', 'World', 'Python']
```

join(iterable)

Example (Join Example)

Joins the elements of an iterable into a single string, separated by the string it's called on.

```
>>> lst = ['Hello', 'World', 'Python']
>>> print(','.join(lst))  # Hello,World,Python
```

isalpha()

Example (Isalpha Example)

```
Returns True if all characters in the string are alphabetic.

>>> s = "Hello"
>>> print(s.isalpha())  # True
```

isdigit()

Example (Isdigit Example)

Returns True if all characters in the string are digits.

```
>>> s = "12345"
>>> print(s.isdigit())  # True
```

swapcase()

Example (Swapcase Example)

```
Swaps the case of all characters in the string.

>>> s = "Hello World"
>>> print(s.swapcase())  # hELLO wORLD
```

zfill(width)

Example (Zfill Example)

Pads the string on the left with zeros until it reaches the sp

```
>>> s = "42"
>>> print(s.zfill(5))  # 00042
```

Extracting Host from Email String

Example (String Manipulation Example)

```
>>> data = 'From premanand.s@vit.ac.in Sat Jan  5 09:14:16 20(
>>> atpos = data.find('@')
>>> print(atpos)
17

>>> sppos = data.find(' ', atpos)
>>> print(sppos)
27

>>> host = data[atpos+1 : sppos]
>>> print(host)
vit.ac.in
```

Two Kinds of Strings in Python

- **str:**
  - The default string type in Python 3.
  - Represents a sequence of Unicode characters.
  - Supports all standard string operations.
  - Example:

```
s = "Hello, World!"
print(type(s))  # <class 'str'>
```

- **unicode:**
  - Explicitly required only in Python 2, where ‘unicode‘ and ‘str‘ are separate types.
  - In Python 3, ‘unicode‘ is merged with ‘str‘.
  - Unicode allows for a wider range of characters from different languages.
  - Example in Python 2:

```
s = u"Hello, World!"
print(type(s))  # <type 'unicode'>
```

Conclusion

- Strings are fundamental in Python for handling text data.
- They offer various methods and operations for manipulation.
- Understanding strings is key for text processing in Python.

Question: String Comparison in Python

Compare two strings: `"orange"` and `"apple"`.

- Write a Python program to compare the two strings.
- Print whether they are equal or which one comes first lexicographically.

Code: String Comparison in Python

Example (Python)

```
# Strings to compare
string1 = "orange"
string2 = "apple"

# Comparing the strings
if string1 == string2:
    print(f'The strings "{string1}" and
                    "{string2}" are equal.')
elif string1 < string2:
    print(f'The string "{string1}" comes
    before "{string2}" in lexicographical order.')
else:
    print(f'The string "{string1}" comes after
    "{string2}" in lexicographical order.')
```

Question: User Input and String Formatting

Extend the program to ask the user for their city. The output should be formatted like this:

- `"Hello [Name], you are [Age] years old, and you live in [City]."`

Code: User Input and String Formatting

Example (Python)

```
# Asking the user for their name, age, and city
name = input("Enter your name: ")
age = input("Enter your age: ")
city = input("Enter the city you live in: ")

# Printing a formatted message
print(f"Hello {name}, you are {age} years old, and
      you live in {city}.")
```

Question: String Slicing in Python

Given the string `"PythonProgramming"`, write a Python program that prints:

- The first 6 characters.
- The last 7 characters.
- Every second character from the string.

Code: String Slicing in Python

Example (Python)

```
# Given string
s = "PythonProgramming"

# Printing the first 6 characters
print("First 6 characters:", s[:6])

# Printing the last 7 characters
print("Last 7 characters:", s[-7:])

# Printing every second character
print("Every second character:", s[::2])
```

Question: Reverse a String Using Slicing

How would you reverse the entire string `"PythonProgramming"` using slicing?

Code: Reverse a String Using Slicing

Example (Python)

```
# Given string
s = "PythonProgramming"

# Reversing the string using slicing
reversed_string = s[::-1]

# Printing the reversed string
print("Reversed string:", reversed_string)
```

Question: Splitting a Sentence into Words

Write a Python program that splits the sentence `"Python is fun and powerful"` into individual words and prints them.

Code: Splitting a Sentence into Words

Example (Python)

```
# Given sentence
sentence = "Python is fun and powerful"

# Splitting the sentence into individual words
words = sentence.split()

# Printing each word
for word in words:
    print(word)
```

Question: Splitting a Sentence by Commas

Modify the program to split the sentence `"apple,banana,grape,orange"` based on commas (,) and print each fruit.

Code: Splitting a Sentence by Commas

Example (Python)

```
# Given sentence
sentence = "apple,banana,grape,orange"

# Splitting the sentence into individual words based on commas
fruits = sentence.split(',')

# Printing each fruit
for fruit in fruits:
    print(fruit)
```

Question: Remove Leading and Trailing Spaces

Given the string `"  Hello World!  "`, write a Python program that removes the leading and trailing spaces and prints the result.

Code: Remove Leading and Trailing Spaces

Example (Python)

```
# Given string with leading and trailing spaces
s = " Hello World! "

# Removing leading and trailing spaces
trimmed_string = s.strip()

# Printing the result
print("Trimmed string:", trimmed_string)
```

Question: Remove Specific Characters

Write a program that removes specific characters from the start and end of the string.
For example:

- `remove('---Hello---', '-')` should return `"Hello"`.

Code: Remove Specific Characters

Example (Python)

```
# Given string
s = '---Hello---'

# Removing specific characters from the start and end of the s
result = s.strip('-')

# Printing the result
print("Result:", result)
```

Question: String Length Comparison

Write a Python program that compares two strings and checks if they have the same length. Then, print whether one is longer or shorter than the other.

Code: String Length Comparison

Example (Python)

```
# Given strings
str1 = "apple"
str2 = "banana"

# Comparing the lengths of the strings
if len(str1) == len(str2):
    print(f'The strings "{str1}" and "{str2}" have
                    the same length.')
elif len(str1) > len(str2):
    print(f'The string "{str1}" is longer than "{str2}".')
else:
    print(f'The string "{str2}" is longer than "{str1}".')
```

Question: Currency Conversion

Write a Python program that takes an amount in USD and converts it to EUR using string formatting to display the result as:

- `"[Amount] USD is equivalent to [Converted Amount] EUR."`

Code: Currency Conversion

Example (Python)

```
# Amount in USD
usd_amount = float(input("Enter amount in USD: "))

# Example exchange rate from USD to EUR
exchange_rate = 0.91
eur_amount = usd_amount * exchange_rate

# Displaying the result using string formatting
print(f"{usd_amount:.2f} USD is equivalent
                    to {eur_amount:.2f} EUR.")
```

Question: Advanced Slicing

Write a Python program that:

- Extracts every third character from the string `"abcdefghijklmno"`.
- Given a string of your choice, slices it to print the middle third of the string.

Code: Advanced Slicing

Example (Python)

```
s = "abcdefghijklmno"
every_third_character = s[::3]
print("Every third character:", every_third_character)

example_string = "This is an example string for slicing."

length = len(example_string)
middle_third_start = length // 3
middle_third_end = middle_third_start * 2

middle_third = example_string[middle_third_start:middle_third_
print("Middle third of the string:", middle_third)
```

Question: Splitting and Joining Sentences

Given the sentence `"I love Python because it is versatile, easy, and powerful!"`, split the sentence into words, and then join them back together with hyphens (-) instead of spaces.

Code: Splitting and Joining Sentences

Example (Python)

```
# Given sentence
sentence = "I love Python because it is versatile, easy,
            and powerful!"

# Splitting the sentence into words
words = sentence.split()

# Joining words with hyphens
joined_sentence = '-'.join(words)

# Printing the result
print("Joined sentence with hyphens:", joined_sentence)
```