Day4_Strings_&_Methods

May 21, 2025

1 Day 4: Strings and String Methods in Python

Today, I explored Python **strings** in depth. I learned how to define strings using single or double quotes, and how to handle multi-line strings using triple quotes. I practiced **string concatenation**, **indexing**, **slicing**, and **unpacking characters**.

I also learned about **escape sequences** such as `\n` (newline), `\t` (tab), and `\\` (backslash), which help format strings in a readable way.

A major part of today's learning was discovering **string methods**—powerful built-in functions like `capitalize()`, `count()`, `find()`, `replace()`, `split()`, `join()`, and many more. These methods help manipulate and analyze strings efficiently, which is a crucial skill in real-world Python programming, especially in data cleaning and user input processing.

2 Key Concepts Practiced:

- String declaration (`'Hello'`, `"World"`)
- Multiline strings (`'''...'''`, `"""..."""`)
- String length with `len()`
- Concatenation and formatting using `+` and `.format()`
- Indexing, slicing, and negative indexing
- Escape characters for formatting (`\n`, `\t`, etc.)
- String methods:
  - `capitalize()`, `count()`, `endswith()`, `find()`, `index()`
  - `isalnum()`, `isalpha()`, `isdigit()`, `isdecimal()`, `isidentifier()`
  - `islower()`, `isupper()`, `isnumeric()`, `strip()`, `replace()`
  - `split()`, `join()`, `title()`, `swapcase()`, `startswith()`

3 Single line comment

[4]:
```
# Single line comment
letter = 'P'                # A string could be a single character or a bunch␣
 ↪of texts
print(letter)               # P
```

```
print(len(letter))          # 1
greeting = 'Hello, World!'  # String could be  a single or double quote,"Hello,␣
 ↪World!"
print(greeting)             # Hello, World!
print(len(greeting))        # 13
sentence = "I hope you are enjoying 30 days of python challenge"
print(sentence)
```

```
P
1
Hello, World!
13
I hope you are enjoying 30 days of python challenge
```

4 Multiline String

[5]:
```
# Multiline String
multiline_string = '''I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python.'''
print(multiline_string)
```

```
I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python.
```

Another way of doing the same thing

[ ]:
```
# Another way of doing the same thing
multiline_string = """I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python."""
print(multiline_string)
```

[ ]:
```

```

[6]:
```
# String Concatenation
first_name = 'Akshay'
last_name = 'Bhujbal'
space = ' '
full_name = first_name  +  space + last_name
print(full_name) # Akshay Bhujbal
# Checking length of a string using len() builtin function
print(len(first_name))  # 6
print(len(last_name))   # 7
print(len(first_name) > len(last_name)) # False
print(len(full_name)) # 14
```

```
Akshay Bhujbal
6
7
False
14
```

5 Unpacking characters

[7]:
```python
#### Unpacking characters
language = 'Python'
a,b,c,d,e,f = language # unpacking sequence characters into variables
print(a) # P
print(b) # y
print(c) # t
print(d) # h
print(e) # o
print(f) # n
```

```
P
y
t
h
o
n
```

6 Accessing characters in strings by index

[8]:
```python
# Accessing characters in strings by index
language = 'Python'
first_letter = language[0]
print(first_letter) # P
second_letter = language[1]
print(second_letter) # y
last_index = len(language) - 1
last_letter = language[last_index]
print(last_letter) # n
```

```
P
y
n
```

[9]:
```python
# If we want to start from right end we can use negative indexing. -1 is the␣
↪last index
language = 'Python'
last_letter = language[-1]
print(last_letter) # n
second_last = language[-2]
```

```
print(second_last) # o
```

```
n
o
```

7 Slicing

[10]:
```
# Slicing
language = 'Python'
first_three = language[0:3] # starts at zero index and up to 3 but not include 3
last_three = language[3:6]
print(last_three) # hon
# Another way
last_three = language[-3:]
print(last_three)   # hon
last_three = language[3:]
print(last_three)   # hon
```

```
hon
hon
hon
```

[11]:
```
# Skipping character while splitting Python strings
language = 'Python'
pto = language[0:6:2] #
print(pto) # pto
```

```
Pto
```

8 Escape sequence

[12]:
```
# Escape sequence
print('I hope every one enjoying the python challenge.\nDo you ?') # line break
print('Days\tTopics\tExercises')
print('Day 1\t3\t5')
print('Day 2\t3\t5')
print('Day 3\t3\t5')
print('Day 4\t3\t5')
print('This is a back slash  symbol (\\)') # To write a back slash
print('In every programming language it starts with \"Hello, World!\"')
```

```
I hope every one enjoying the python challenge.
Do you ?
Days    Topics  Exercises
Day 1   3       5
Day 2   3       5
Day 3   3       5
Day 4   3       5
```

```
This is a back slash  symbol (\)
In every programming language it starts with "Hello, World!"
```

9 String Methods

9.0.1 capitalize(): Converts the first character the string to Capital Letter

[13]:
```
# capitalize(): Converts the first character the string to Capital Letter

challenge = 'thirty days of python'
print(challenge.capitalize()) # 'Thirty days of python'
```

```
Thirty days of python
```

9.0.2 count(): returns occurrences of substring in string, count(substring, start=.., end=..)

[14]:
```
# count(): returns occurrences of substring in string, count(substring, start=..
 ↪, end=..)

challenge = 'thirty days of python'
print(challenge.count('y')) # 3
print(challenge.count('y', 7, 14)) # 1
print(challenge.count('th')) # 2`
```

```
3
1
2
```

9.0.3 endswith(): Checks if a string ends with a specified ending

[15]:
```
# endswith(): Checks if a string ends with a specified ending

challenge = 'thirty days of python'
print(challenge.endswith('on'))   # True
print(challenge.endswith('tion')) # False
```

```
True
False
```

9.0.4 expandtabs(): Replaces tab character with spaces, default tab size is 8. It takes tab size argument

[16]:
```
# expandtabs(): Replaces tab character with spaces, default tab size is 8. It␣
 ↪takes tab size argument

challenge = 'thirty\tdays\tof\tpython'
print(challenge.expandtabs())   # 'thirty  days    of      python'
```

```
print(challenge.expandtabs(10)) # 'thirty    days      of        python'
```

```
thirty  days    of      python
thirty    days      of        python
```

9.0.5 find(): Returns the index of first occurrence of substring

[17]:
```
# find(): Returns the index of first occurrence of substring

challenge = 'thirty days of python'
print(challenge.find('y'))  # 5
print(challenge.find('th')) # 0
```

```
5
0
```

9.0.6 format() formats string into nicer output

[18]:
```
# format()        formats string into nicer output
first_name = 'Akshay'
last_name = 'Bhujbal'
job = 'Data Analyst'
country = 'Finland'
sentence = 'I am {} {}. I am a {}. I live in {}.'.format(first_name, last_name,␣
 ↪job, country)
print(sentence) # I am Akshay Bhujbal. I am a Data Analyst. I live in Finland.
```

```
I am Akshay Bhujbal. I am a Data Analyst. I live in Finland.
```

[19]:
```
radius = 10
pi = 3.14
area = pi # radius ## 2
result = 'The area of circle with {} is {}'.format(str(radius), str(area))
print(result) # The area of circle with 10 is 314.0
```

```
The area of circle with 10 is 3.14
```

9.0.7 index(): Returns the index of substring

[20]:
```
# index(): Returns the index of substring
challenge = 'thirty days of python'
print(challenge.find('y'))  # 5
print(challenge.find('th')) # 0
```

```
5
0
```

9.0.8 isalnum(): Checks alphanumeric character

[21]:
```
challenge = 'ThirtyDaysPython'
print(challenge.isalnum()) # True

challenge = '30DaysPython'
print(challenge.isalnum()) # True

challenge = 'thirty days of python'
print(challenge.isalnum()) # False

challenge = 'thirty days of python 2019'
print(challenge.isalnum()) # False
```

```
True
True
False
False
```

9.0.9 isalpha(): Checks if all characters are alphabets

[22]:
```
# isalpha(): Checks if all characters are alphabets

challenge = 'thirty days of python'
print(challenge.isalpha()) # True
num = '123'
print(num.isalpha())       # False
```

```
False
False
```

9.0.10 isdecimal(): Checks Decimal Characters

[23]:
```
# isdecimal(): Checks Decimal Characters

challenge = 'thirty days of python'
print(challenge.find('y'))  # 5
print(challenge.find('th')) # 0
```

```
5
0
```

9.0.11 isdigit(): Checks Digit Characters

[26]:
```
# isdigit(): Checks Digit Characters

challenge = 'Thirty'
print(challenge.isdigit()) # False
challenge = "30"
```

```
print(challenge.isdigit()) # True
```

```
False
True
```

9.0.12 isdecimal():Checks decimal characters

[27]:
```
# isdecimal():Checks decimal characters

num = '10'
print(num.isdecimal()) # True
num = '10.5'
print(num.isdecimal()) # False
```

```
True
False
```

9.0.13 isidentifier():Checks for valid identifier means it check if a string is a valid variable name

[28]:
```
# isidentifier():Checks for valid identifier means it check if a string is a␣
 ↪valid variable name

challenge = '30DaysOfPython'
print(challenge.isidentifier()) # False, because it starts with a number
challenge = 'thirty_days_of_python'
print(challenge.isidentifier()) # True
```

```
False
True
```

9.0.14 islower():Checks if all alphabets in a string are lowercase

[29]:
```
# islower():Checks if all alphabets in a string are lowercase

challenge = 'thirty days of python'
print(challenge.islower()) # True
challenge = 'Thirty days of python'
print(challenge.islower()) # False
```

```
True
False
```

9.0.15 isupper(): returns if all characters are uppercase characters

[30]:
```
# isupper(): returns if all characters are uppercase characters

challenge = 'thirty days of python'
print(challenge.isupper()) #  False
```

```
challenge = 'THIRTY DAYS OF PYTHON'
print(challenge.isupper()) # True
```

```
False
True
```

9.0.16 isnumeric():Checks numeric characters

[31]:
```
# isnumeric():Checks numeric characters

num = '10'
print(num.isnumeric())      # True
print('ten'.isnumeric())    # False
```

```
True
False
```

9.0.17 join(): Returns a concatenated string

[32]:
```
# join(): Returns a concatenated string

web_tech = ['HTML', 'CSS', 'JavaScript', 'React']
result = '#, '.join(web_tech)
print(result) # 'HTML# CSS# JavaScript# React'
```

```
HTML#, CSS#, JavaScript#, React
```

9.0.18 strip(): Removes both leading and trailing characters

[33]:
```
# strip(): Removes both leading and trailing characters

challenge = ' thirty days of python '
print(challenge.strip('y')) # 5
```

```
 thirty days of python
```

9.0.19 replace(): Replaces substring inside

[34]:
```
# replace(): Replaces substring inside

challenge = 'thirty days of python'
print(challenge.replace('python', 'coding')) # 'thirty days of coding'
```

```
thirty days of coding
```

9.0.20 split():Splits String from Left

[35]:
```
# split():Splits String from Left

challenge = 'thirty days of python'
print(challenge.split()) # ['thirty', 'days', 'of', 'python']
```

```
['thirty', 'days', 'of', 'python']
```

9.0.21 title(): Returns a Title Cased String

[36]:
```
# title(): Returns a Title Cased String

challenge = 'thirty days of python'
print(challenge.title()) # Thirty Days Of Python
```

```
Thirty Days Of Python
```

9.0.22 swapcase(): Checks if String Starts with the Specified String

[37]:
```
# swapcase(): Checks if String Starts with the Specified String

challenge = 'thirty days of python'
print(challenge.swapcase())   # THIRTY DAYS OF PYTHON
challenge = 'Thirty Days Of Python'
print(challenge.swapcase())  # tHIRTY dAYS oF pYTHON
```

```
THIRTY DAYS OF PYTHON
tHIRTY dAYS oF pYTHON
```

9.0.23 startswith(): Checks if String Starts with the Specified String

[38]:
```
# startswith(): Checks if String Starts with the Specified String

challenge = 'thirty days of python'
print(challenge.startswith('thirty')) # True
challenge = '30 days of python'
print(challenge.startswith('thirty')) # False
```

```
True
False
```

[ ]:
```

```