Strings

Text is a string data type. Any data type written as text is a string. Any data under single, double or triple quote are strings. There are different string methods and built-in functions to deal with string data types. To check the length of a string use the len() method.

Creating a String

```
letter = 'PKB'              # A string could be a single character or a
bunch of texts
print(letter)               # PKB
print(len(letter))          # 3
greeting = 'Hello, World!'  # keep in a string "Hello, World!"
print(greeting)             # Hello, World!
```

String Concatenation

We can connect strings together. Merging or connecting strings is called concatenation.

```
first_name = 'Prasanta'
last_name = 'Biswal'
space = ' '
full_name = first_name  +  space + last_name
print(full_name) # Prasanta Biswal
```

Escape Sequences in Strings

In Python and other programming languages \ followed by a character is an escape sequence. Let us see the most common escape characters:

- \n: new line
- \t: Tab means(8 spaces)
- \\: Back slash
- \': Single quote (')
- \": Double quote (")

```
print('I hope everyone is enjoying the Python Course.\nAre you ?') # line
break
print('Days\tTopics\tExercises') # adding tab space or 4 spaces
print('Day 1\t5\t5')
print('Day 2\t6\t20')
print('Day 3\t5\t23')
print('Day 4\t1\t35')
print('This is a backslash  symbol (\\)') # To write a backslash
print('In every programming language it starts with \"Hello, World!\"') #
to write a double quote inside a single quote

# output
I hope every one is enjoying the Python Course.
Are you ?
Days    Topics  Exercises
Day 1   5           5
Day 2   6           20
Day 3   5           23
```

```
Day 4   1           35
This is a backslash  symbol (\)
In every programming language it starts with "Hello, World!"
```

String formatting

New Style String Formatting (str.format)

```
first_name = 'Prasanta'
last_name = 'Biswal'
language = 'Python'
formated_string = 'I am {} {}. I am learning {}'.format(first_name,
last_name, language)
print(formated_string)
#output
I am Prasanta Biswal. I am learning Python

a = 4
b = 3

print('{} + {} = {}'.format(a, b, a + b))
print('{} - {} = {}'.format(a, b, a - b))

# output
4 + 3 = 7
4 - 3 = 1
```

Accessing Characters in Strings by Index

In programming counting starts from zero. Therefore the first letter of a string is at zero index and the last letter of a string is the length of a string minus one.

```
language = 'Python'
first_letter = language[0]
print(first_letter) # P
last_letter = language[-1]
print(last_letter) # n
```

Slicing Python Strings

In python we can slice strings into substrings.

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

Reversing a String

We can easily reverse strings in python.

```
greeting = 'Hello, World!'
print(greeting[::-1]) # !dlroW ,olleH
```

Skipping Characters While Slicing

It is possible to skip characters while slicing by passing step argument to slice method.

```
language = 'Python'
pto = language[0:6:2] #
print(pto) # Pto
```

String Methods

There are many string methods which allow us to format strings. See some of the string methods in the following example:

- **capitalize**(): Converts the first character of the string to capital letter

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

- **count**(): returns occurrences of substring in string, count(substring, start=.., end=..). The start is a starting indexing for counting and end is the last index to count.

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

- **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
```

- **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'
```

- **find**(): Returns the index of the first occurrence of a substring, if not found returns -1

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

- **rfind**(): Returns the index of the last occurrence of a substring, if not found returns -1

```
challenge = 'thirty days of python'
print(challenge.rfind('y'))  # 16
print(challenge.rfind('th')) # 17

radius = 10
```

```
pi = 3.14
area = pi * radius ** 2
result = 'The area of a circle with radius {} is {}'.format(str(radius),
str(area))
print(result) # The area of a circle with radius 10 is 314
```

- **index**(): Returns the lowest index of a substring, additional arguments indicate starting and ending index (default 0 and string length - 1). If the substring is not found it raises a valueError.

```
challenge = 'thirty days of python'
sub_string = 'da'
print(challenge.index(sub_string))  # 7
print(challenge.index(sub_string, 9)) # error
```

- **rindex**(): Returns the highest index of a substring, additional arguments indicate starting and ending index (default 0 and string length - 1)

```
challenge = 'thirty days of python'
sub_string = 'da'
print(challenge.rindex(sub_string))  # 7
print(challenge.rindex(sub_string, 9)) # error
print(challenge.rindex('on', 8)) # 19
```

- **isalnum**(): Checks alphanumeric character

```
challenge = 'ThirtyDaysPython'
print(challenge.isalnum()) # True

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

challenge = 'thirty days of python'
print(challenge.isalnum()) # False, space is not an alphanumeric character

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

- **isalpha**(): Checks if all string elements are alphabet characters (a-z and A-Z)

```
challenge = 'thirty days of python'
print(challenge.isalpha()) # False, space is once again excluded
challenge = 'ThirtyDaysPython'
print(challenge.isalpha()) # True
num = '123'
print(num.isalpha())      # False
```

- **isdecimal**(): Checks if all characters in a string are decimal (0-9)

```
challenge = 'thirty days of python'
print(challenge.isdecimal())  # False
challenge = '123'
print(challenge.isdecimal())  # True
challenge = '\u00B2'
print(challenge.isdigit())   # False
challenge = '12 3'
print(challenge.isdecimal())  # False, space not allowed
```

- **isdigit**(): Checks if all characters in a string are numbers (0-9 and some other unicode characters for numbers)

```
challenge = 'Thirty'
print(challenge.isdigit()) # False
challenge = '30'
print(challenge.isdigit())   # True
challenge = '\u00B2'
print(challenge.isdigit())   # True
```

- **isnumeric**(): Checks if all characters in a string are numbers or number related (just like isdigit(), just accepts more symbols, like ½)

```
num = '10'
print(num.isnumeric()) # True
num = '\u00BD' # ½
print(num.isnumeric()) # True
num = '10.5'
print(num.isnumeric()) # False
```

- **isidentifier**(): Checks for a valid identifier - it checks 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
```

- **islower**(): Checks if all alphabet characters in the string are lowercase

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

- **isupper**(): Checks if all alphabet characters in the string are uppercase

```
challenge = 'thirty days of python'
print(challenge.isupper()) #  False
challenge = 'THIRTY DAYS OF PYTHON'
print(challenge.isupper()) # True
```

- **join**(): Returns a concatenated string

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

- **strip**(): Removes all given characters starting from the beginning and end of the string

```
challenge = 'thirty days of pythoonnn'
print(challenge.strip('noth')) # 'irty days of py'
```

- **replace**(): Replaces substring with a given string

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

- **split**(): Splits the string, using given string or space as a separator

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

- **title**(): Returns a title cased string

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

- **swapcase**(): Converts all uppercase characters to lowercase and all lowercase characters to uppercase characters

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

- **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
```