string

July 20, 2024

1 String

A string is a sequence of characters enclosed within either single quotes ” or double quotes ””. String are immutable

1.1 Creating

[1]:
```
print(str())          # Create a empty string
print('Hello')        # Create a string within double quotes/
```

```

Hello
```

1.2 Manipulation

consider a= ‘hello world’

[2]:
```
a= 'hello World\t'
```

[3]:
```
print('Capitalize \t:'+a.capitalize())                          # Returns a␣
  ↪copy of first character capitalized string
print('Casefold \t:'+a.casefold())                            # Returns a copy␣
  ↪of casefolded string for case-insensitive comparison
print('Centre \t\t:'+a.center(20))                            # Return a␣
  ↪centered string of length width.
print(f'Encode \t\t:{a.encode(encoding='UTF-16')}')             # Encode the␣
  ↪string using the codec registered for encoding.
print('Expandtabs \t:'+a.expandtabs())                          # Returns copy␣
  ↪of string where all tab characters are replaced by spaces
print('Join \t\t:'+','.join((a,a)))                                 #␣
  ↪Concatinates strings from the iterables with the intial string being␣
  ↪separator.
print('Ljust \t\t:'+a.ljust(41))                             # Returns a left␣
  ↪justified string of length width with the original string
print('Rjust \t\t:'+a.rjust(40))                             # Returns a right␣
  ↪justified string of length width
```

```
Capitalize      :Hello world
Casefold        :hello world
Centre          :    hello World
Encode          :b'\xff\xfeh\x00e\x00l\x00l\x00o\x00
\x00W\x00o\x00r\x00l\x00d\x00\t\x00'
Expandtabs      :hello World
Join            :hello World    ,hello World
Ljust           :hello World
Rjust           :                            hello World
```

[4]:
```
a='    Strip example     '
print('Lstrip \t\t:'+a.lstrip())    #Return a copy of the string with leading␣
 ↪whitespace removed.
print('Rstrip \t\t:'+a.rstrip())    #Return a copy of the string with leading␣
 ↪whitespace removed.
print('Strip \t\t:'+a.strip())      #Return a copy of the string with leading␣
 ↪whitespace removed.
```

```
Lstrip          :Strip example
Rstrip          :    Strip example
Strip           :Strip example
```

[5]:
```
a='partition,replace'
print(f'Patition \t:{a.partition(',')}')        # Returns a tuple separating␣
 ↪the string in 3 parts
print('Replace \t:'+a.replace(',',' '))         # Retunrs a copy of the replace␣
 ↪string with all occurences
print(f'Split \t\t:{a.split(',')}')             # Return a list of the␣
 ↪substrings in the string, using sep as the separator string.
print('Title \t\t:'+a.title())                  # Returns a title cased version␣
 ↪of string
print('Swapcase \t:'+a.swapcase())              # Returns a string of uppercase␣
 ↪char converted to lowercase or viceverse
print('Zfill \t\t:'+a.zfill(20))                # Returns a string of zeros␣
 ↪filled on the left
```

```
Patition        :('partition', ',', 'replace')
Replace         :partition replace
Split           :['partition', 'replace']
Title           :Partition,Replace
Swapcase        :PARTITION,REPLACE
Zfill           :000partition,replace
```

1.3 Inspection

[6]:
```
a='Inspection of a  String'
print(f'Count  \t\t:{a.count('i')}')        # Returns the number of occurrences␣
 ↪of substring in the range
print(f'Endswith \t:{a.endswith('ng')}')    # Return True if S ends with the␣
 ↪specified suffix, False otherwise.
print(f'Find \t\t:{a.find('g')}')           # Return the lowest index in S␣
 ↪where substring sub is found.
print(f'Index \t\t:{a.index('o')}')         # Return the lowest index in S␣
 ↪where substring sub is found.
```

```
Count           :2
Endswith        :True
Find            :22
Index           :8
```

[7]:
```
a='Python'
b='Python123'
c='12'
print(f"String\t\t{a}\t{b}\t{c}")
print(f'Isalnum \t:{a.isalnum()}\t{b.isalnum()}\t\t{c.isalnum()}')              ␣
 ↪# Return True if the string is an alpha-numeric string, False otherwise.
print(f'Isalpha \t:{a.isalpha()}\t{b.isalpha()}\t\t{c.isalpha()}')              ␣
 ↪# Return True if the string is an alphabetic string, False otherwise.
print(f"Isascii \t:{a.isascii()}\t{b.isascii()}\t\t{c.isascii()}")              ␣
 ↪# Return True if all characters in the string are ASCII, False otherwise.
print(f"Isdecimal\t:{a.isdecimal()}\t{b.isdecimal()}\t\t{c.isdecimal()}")       ␣
 ↪# Return True if the string is a decimal string, False otherwise.
print(f"isdigit \t:{a.isdigit()}\t{b.isdigit()}\t\t{c.isdigit()}")              ␣
 ↪# Return True if the string is a digit string, False otherwise.
print(f"Isidentifier\t:{a.isidentifier()}\t{b.isidentifier()}\t\t{c.
 ↪isidentifier()}") # Return True if the string is a valid Python identifier,␣
 ↪False otherwise.
print(f"islower\t\t:{a.islower()}\t{b.islower()}\t\t{c.islower()}")
```

```
String          Python  Python123       12
Isalnum         :True   True            True
Isalpha         :True   False           False
Isascii         :True   True            True
Isdecimal       :False  False           True
isdigit         :False  False           True
Isidentifier    :True   True            False
islower         :False  False           False
```

[1]:
```
a='PYTHON'
b='Python123'
```

```
c='12'
print(f"String\t\t{a}\t{b}\t{c}")
print(f'isnumeric \t:{a.isnumeric()}\t{b.isnumeric()}\t\t{c.isnumeric()}')      ␣
 ↪        # Return True if the string is a numeric string, False otherwise.
print(f"isprintable \t:{a.isprintable()}\t{b.isprintable()}\t\t{c.
 ↪isprintable()}")      # Return True if the string is printable, False␣
 ↪otherwise.
print(f"isspace \t:{a.isspace()}\t{b.isspace()}\t\t{c.isspace()}")           ␣
 ↪        # Return True if the string is a whitespace string, False otherwise.
print(f"istite \t\t:{a.istitle()}\t{b.istitle()}\t\t{c.istitle()}")          ␣
 ↪        # Return True if the string is a title-cased string, False otherwise.
print(f"isupper \t:{a.isupper()}\t{b.isupper()}\t\t{c.isupper()}")           ␣
 ↪        # Return True if the string is an uppercase string, False otherwise.
print(f"stratswith\t:{a.startswith('P')}\t{b.startswith('p')}\t\t{c.
 ↪startswith('1')}")  # Return True if S starts with the specified prefix,␣
 ↪False otherwise.
```

```
String          PYTHON  Python123       12
isnumeric       :False  False           True
isprintable     :True   True            True
isspace         :False  False           False
istite          :False  True            False
isupper         :True   False           False
stratswith      :True   False           True
```

1.4 Transformation

[7]:
```
a='Hello world'
print(f'Lower \t\t:{a.lower()}')        # Return a copy of the string converted␣
 ↪to lowercase.
print(f'Upper \t\t:{a.upper()}')        # Return a copy of the string converted␣
 ↪to uppercase.
print(f"Swapcase\t:{a.swapcase()}")     # Convert uppercase characters to␣
 ↪lowercase and lowercase characters to uppercase.
print(f"Title \t\t:{a.title()}")        # Return a version of the string where␣
 ↪each word is titlecased.
```

```
Lower           :hello world
Upper           :HELLO WORLD
Swapcase        :hELLO WORLD
Title           :Hello World
```

1.5 Formatting

[9]:
```python
name = "Alice"
age = 30
formatted_string = "My name is {} and I am {} years old.".format(name, age)
print(formatted_string)
```

```
My name is Alice and I am 30 years old.
```

[10]:
```python
person = {'name': 'Alice', 'age': 30}
formatted_string = "My name is {name} and I am {age} years old.".
  ↪format_map(person)
print(formatted_string)
```

```
My name is Alice and I am 30 years old.
```

[12]:
```python
# Maketrans
# Creating a translation table to replace characters
table = str.maketrans('aeiou', '12345')
example_string = "hello world"
# Applying translation using translate() method
translated_string = example_string.translate(table)
print(translated_string)
```

```
h2ll4 w4rld
```

I hope you found this information helpful! Feel free to save this post for future reference. Let's continue to learn and grow together!

Rajendra Prasad