String Class:

1. Operators
2. String Methods

A white sheet of paper with text

AI-generated content may be incorrect.

* String is a collection of characters in double quotes or single quotes.

s1="Hello"

s2='9848045789'

s3="my.id123@mail.com"

s4="my Book"

* Sequence of letters

+ve / -ve indices

S1🡪 H e l l o

0 1 2 3 4

-5 -4 -3 -2 -1

String length:

s1="Hello"

print(s1[0])

#H

print(s1[-5])

#H

print(s1[-1])

#o

print(s1[4])

#o

print(len(s1))

#5

#############

Traverse:

s1="Hello"

for i in s1:

    print(i,end=' ')

#H e l l o

print()

for i in range(len(s1)):

    print(s1[i],end=" ")

#H e l l o

########### STRING LITERALS

s1="Hello"

s2='Hello'

s3="""Hello1

Hello2

Hello3

"""

Immu

s5='Clark\"s'

s3="Clark's"

s3='Clark"s'

print(s3) # Clark"s

INDEXING and SLICING

1. String indexing
2. Immutable string
3. Strin slicing

Slicing:

s1="Hello World"

print(s1[0])

# H

print(s1[-7])

# o

# s1[0]='w'

# print(s1)

#TypeError: 'str' object does not support item assignment

###### String slicing

# []

# s1[start:stop:step]

# it s like range(start,stop,step)

#indexing and slicing for string lists tuple--> Sequence type

s1="Hello world"

print(s1[6])

# w

print(s1[-5])

# w

s1="Hello world"

print(s1[1:7])

#ello w

print(s1[3:7])

# lo w

print(s1[:7]) #0,7

# Hello w

print(s1[:])

# Entire string

# Hello world

print(s1[6:])

# end of string

# world

print(s1[-5:])

#world

print(s1[-5:-2])

#wor

print(s1[-11:-2])

# Hello wor

# forward direction

# Step

print(s1[0:11:1])

# By default step is 1

#Hello world

print(s1[0:11:2])

# Hlowrd

print(s1[::])

# Hello world

print(s1[::-1])

# dlrow olleH

s1="Hello world"

s2=s1[::]

print(s2[::-1])

# dlrow olleH (REVERSE)

print(s2[-3::-1])

# row olleH

########

s2="Hello world"

print(s2[-3:9:-1])

# no print . forward direction only

print(s2[8:2:-1])

#row ol

STRING operations

Arithmetic and String Concatenation

Concatenation : +

s1="abc"

s2=" CDE"

print(s1+s2)

#abc CDE

Repetation: \*

s1="abc "

print(s1\*5)

# abc abc abc abc abc

Member Ship: in , not in

print("a" in s1)

#True

print("2" in s1)

# False

String comparison: <.<=,>,<=,==,!=

s1="software"

s2="Hardware"

# s1 > s2 🡪True

s1="python"

s2="pycharm"

#s1<s2--> False

#s1==s2 🡪 True

s1="printer"

s2="print"

# s1>s2

Class String:

Class and Object

s1="Hello"

print(type(s1))

# <class 'str'>

A person pointing at a diagram

AI-generated content may be incorrect.

print(dir(s1))

$ python class\_string.py

['\_\_add\_\_', '\_\_class\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_getitem\_\_', '\_\_getnewargs\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_init\_\_', '\_\_init\_subclass\_\_', '\_\_iter\_\_', '\_\_le\_\_', '\_\_len\_\_', '\_\_lt\_\_', '\_\_mod\_\_', '\_\_mul\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_rmod\_\_', '\_\_rmul\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format\_map', 'index', 'isalnum', 'isalpha', 'isascii', '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']

FIND and INDEX:

Find:

s1="Hello how are you"

print(s1.find('o'))

#4

print(s1.find("how"))

#6

print(s1.find('k'))

# -1

print(s1.find('o',5))

#7

print(s1.find('o,5,7'))

#-1

print(s1.find("how",0,5))

# -1

print(s1.find("how",0,9))

#6

Rfind:

s1="Hello how are you"

rfind(sub,start,end)

print(s1.rfind('o'))

#15

print(s1.rfind('o',0,15))

#7

print(s1.rfind('kite'))

#-1

Index:

#index(sub,start,end)

print(s1.index('o'))

#4

print(s1.index("how"))

#6

print(s1.index('k'))

#ValueError: substring not found

Rindex:

# #rindex(sub,start,end)

print(s1.rindex('o'))

#15

print(s1.rindex('o',0,15))

#7

count(sub,start,end)

print(s1.count('o'))

#3

print(s1.count("me"))

#0

String alignment and padding

* ljust(width,fillchar)

H e l l o \* \* \* \* \*

* rjust(width,fillchar)

A computer screen shot of words

AI-generated content may be incorrect.

* center(width,fillchar)

A screen shot of a computer

AI-generated content may be incorrect.

* zfill(width)

A white background with red and blue text

AI-generated content may be incorrect.

STRIP:

A computer screen shot of a word

AI-generated content may be incorrect.

Lstrip🡪 will remove trailing spaces

Rstrip:

A hand pointing at a white board

AI-generated content may be incorrect.

Strip:

A white background with red and blue text

AI-generated content may be incorrect.

No new string .

Ljust:

#ljust(width,fillchar)

s="Hello"

print(s.ljust(10,"\*"))

# #no new string will be create

# # Hello\*\*\*\*\*

Rjust:

s="Hello"

print(s.rjust(10))

#     Hello

print(s.rjust(10,"\_"))

#\_\_\_\_\_Hello

Center:

s="Hello"

print(s.center(10,"\*"))

# \*\*Hello\*\*\*

Zfill:

#zfill(width)

print(s.zfill(10))

#00000Hello

STRIP METHODS:

# lstrip

s="     Hello"

print(s.lstrip())

#by default will be strip " "

# Hello

s="$$Hello"

print(s.lstrip("$"))

#Hello

Rstrip:

s="Hello     "

print(s.rstrip())

#Hello

s="$$$$Hello$$$$"

print(s.strip("$"))

#Hello

s="#! Hello  $\*"

x=s.strip("#!$\*")

print(x)

#Hello

JOING and SPLIITING:

Find and Replace

A computer screen shot of a computer code

AI-generated content may be incorrect.

# replace(old,new,count)

s1="a-b-c-d-e"

s2=s1.replace("-",",")

print(s2)

#a,b,c,d,e

print(s1.replace("-",",",3))

#a,b,c,d-e

print(s1.replace("k","m"))

#a-b-c-d-e (No changes on string if no matching "m")

s1="narendra.nya@gmail.com"

print(s1.replace("gmail","yahoo"))

#narendra.nya@yahoo.com

Join(iterable)

A close-up of a computer screen

AI-generated content may be incorrect.

s1="xyz"

s2="abc"

print(s1.join(s2))

#axyzbxyzc

s1="/"

s2="abc"

print(s1.join(s2))

#a/b/c

Split

A close-up of a computer screen

AI-generated content may be incorrect.

#split

s1="John Smith Ajay"

s2=s1.split()

print(s2)

print(type(s2))

#['John', 'Smith', 'Ajay']

# <class 'list'>

s2=s1.split("h")

print(s2)

#['Jo', 'n Smit', ' Ajay']

s1="John-Smith-Ajay-Khan-James"

print(s1.split(" "))

# ['John-Smith-Ajay-Khan-James']

print(s1.split("-"))

# ['John', 'Smith', 'Ajay', 'Khan', 'James']

print(s1.split("-",3))

#['John', 'Smith', 'Ajay', 'Khan-James']

Rsplit:

A close up of a text

AI-generated content may be incorrect.

Same way

A close up of a text

AI-generated content may be incorrect.

Multi line

A screen shot of a computer

AI-generated content may be incorrect.

s1="python is very easy"

print(s1.startswith("python"))

#True

print(s1.startswith("is"))

#False

print(s1.startswith("is",7))

#True

print(s1.endswith("easy"))

#True

s1="abc@gmail.com"

print(s1.endswith(".com"))

#True

A close-up of a computer screen

AI-generated content may be incorrect.

s1="python is easy"

s2=s1.partition("is")

print(s2)

#('python ', 'is', ' easy')

print(type(s2))

#('python ', 'is', ' easy')

# <class 'tuple'>

A close-up of a computer screen

AI-generated content may be incorrect.

A close-up of a computer screen

AI-generated content may be incorrect.

s1="python is easy"

print(s1.partition("s"))

#('python i', 's', ' easy')

A screen shot of a computer

AI-generated content may be incorrect.

Case Conversion Methods:

A close-up of a computer screen

AI-generated content may be incorrect.

#capitlize

s1="hello world"

print(s1.capitalize())

# Hello world

s2="hello Dear"

print(s2.capitalize())

#Hello dear

A computer screen shot of a word

AI-generated content may be incorrect.

s1="hello"

print(s1.upper())

#HELLO

A computer screen shot of a word

AI-generated content may be incorrect.

s2="HELLO"

print(s2.lower())

#hello

A computer screen shot of a computer

AI-generated content may be incorrect.

#title

s1="hello how are you"

print(s1.title())

#Hello How Are You

A screen shot of a computer

AI-generated content may be incorrect.

## swapcase

s1="helLo how ARe yoU"

print(s1.swapcase())

#HELlO HOW arE YOu

A screen shot of a computer

AI-generated content may be incorrect.

#casefold

S1="Hello how are you"

print(s1.casefold())

# hello how are you

Casefold:

Inquiry Methods:

Isalpha()🡪 Only alphabets

Islower() 🡪 Lower letters

Isupper() 🡪 Upper letters

*Istitle() 🡪 title or not*

s1="Hello"

print(s1.isalpha())

#True

s1="hello"

print(s1.islower())

#True

s1="Hello"

print(s1.isupper())

#False

s1="Hello how are you"

print(s1.istitle())

#False

s=" "

print(s.isspace())

#True

s=""

print(s.isspace())

#False

s=""

print(len(s))

# 0

s="\n\r\r\f"

print(s.isspace())

#True --> Whitespaces

###### printable

s="\n"

print(s.isprintable())

#False--> Escape seq

# \n\r\f\b\a esc'

# isidentifier

s1="1abc"

print(s1.isidentifier())

#False

s2="abc1"

print(s2.identifier())

#Valid variable name

Some more Enquiry methods:

Isascii

A white paper with black text

AI-generated content may be incorrect.

Isalphanum()

A close-up of a white box

AI-generated content may be incorrect.