PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Strings

Prof. Sindhu R Pai
PCPS Theory Anchor - 2024
Department of Computer Science and Engineering

Introduction

A **string** is a collection of characters or a sequence of characters.

Creating strings is as straightforward as assigning a value to a variable.

Ex:

```
var1 = “”      # empty string.

var1 = "python"

var1 = “””” this is python””””
```

Characteristics

- A string has zero or more characters.

```
>>> str1=" “                    >>> str1="p"
>>> str2="python“               >>> str3 = “’’python pro “’’
```

- Each character of a string can be accessed by using the index that starts from 0. Negative indices are allowed.

Example:-

```
>>> str2[0]
    'p’
>>> str2[-1]
    ‘n’
```

- The index value can be an expression that is computed.

```
>>> str2[2+3]
    'n’
```

Characteristics

- **Immutable** – cannot modify the individual characters in a string. One cannot **add, delete, or replace** characters of a string.

Example:-

```
str2="python"
>>> str2[2]='T'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Characteristics

- **Iterable:** Traversing elements one by one.

Example:-

```
str2="python"
for i in str2:
    print(i, end=" ")
```

Output: p y t h o n

Methods of String

All string operations that “modify” a string return a new string that is a modified version of the original string.

Common methods of String

- **The len(str)** returns the number of characters in a string.

```
>>> str1="python"
>>> len(str1)
6
```

- **s.index(chr)** returns the index of the first occurrence of chr in s.

```
>>> str1="python programming“
>>> str1.index(‘p’)
0
>>> str1.index('prog’)
7
```

- **count() -** Returns the number of times a specified value / character occurs in a string.

```
>>> str1="Welcome to Python Class“
>>> str1.count('s’)
2
```

Common methods of String

- **len(str)** returns the number of characters in a string.

```
>>> str1="python"
>>> len(str1)
6
```

min and max functions as applied to strings return the smallest and largest character respectively based on the underlying Unicode encoding.

For example, all lowercase letters are larger have a larger Unicode value than all uppercase letters.

- **Example:-**

```
>>> str1="Python“
>>> min(str1)
'P’
>>> max(str1)
'y'
```

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Common operators applied on string

Concatenation (+) operator can be used to add multiple strings together.

```
>>> str1="Python“          >>> str2="Language"
>>> str1+str2
'PythonLanguage’
```

Repetition (*) operator returns True if a sequence with the specified value is present in the string otherwise False.

```
>>> str1="Python Programming"
>>> str1*2
'Python ProgrammingPython Programming'
```

Common operators applied on string

Scope resolution (::) operator assigns the characters of string str1 into an another string str2.

```
>>> str1="Python Programming"
>>> str2=str1[::]      // same as str2=str1
>>> str2
'Python Programming’
>>> id(str1)
1534440206336
>>> id(str2)
1534440206336
```

Common operators applied on string

Membership (in) operator returns True if a sequence with the specified value is present in the string otherwise False.

```
>>> str2="Language"
>>> "Lang" in "Language"
True
>>> "Png" in "Language"
False
```

Membership (not in) operator is used to repeat the string to a certain length.

```
>>> str1="Language"
>>> "LAN" not in str1
True
>>> "Lang" not in str1
False
```

Common operators applied on string

The slice operator s[start:end] returns the substring starting with index start, up to but not including index end.

- **Example:-**

```
>>> s = 'Monty Python'
>>> s[0:4]
    Mont
>>> s[6:7]
    P
>>> s[6:20]
    Python
```

Specific methods of String

- **The dir() function** returns all properties and methods of the specified object.

```
>>>dir(str)
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__getitem__', '__getnewargs__', '__getstate__', '__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', 'removeprefix', 'removesuffix', 'replace', 'rfind',
'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate',
'upper', 'zfill']
```

Specific methods of String

- **index(chr)** returns the index of the first occurrence of chr in s.

```
>>> str1="python programming“
>>> str1.index('prog’)
>>> str1.index(‘p’)
0
```

- **count() -** Returns the number of times a specified value / character occurs in a string.

```
>>> str1="Welcome to Python Class“
>>> str1.count('s’)
2
```

Specific functions of string

startswith(prefix,start,end) returns True if string starts with the given prefix otherwise returns False.

prefix – A string that needs to be checked.

start – starting position where prefix is needed to be checked within the string.

end- Ending position where prefix is needed to be checked within the string.

Ex:-

```
>>> str1.startswith("W",0)
True
>>> str1.startswith("W",1)
False
>>> str1.startswith("App",5,8)
True
>>> str1.startswith("App",6,9)
False
```

Specific functions of string

endswith(search_string,start,end) returns True if original string ends with the search_string otherwise returns False.

search_string – A string to be searched.

start – starting position of the string from where the search_string is to be searched.

end- Ending position of the string to be considered for searching.

Ex:-

```
>>> str1="Welcome to Python Class“
>>> str1.endswith("Class")
>>> str1.endswith("Class")
True
>>> str1.endswith("Class",18)
True
>>> str1.endswith("Class",19)
False
>>> str1.endswith("Class",18,22)
False
>>> str1.endswith("Class",18,23)
True
```

Specific functions of string

- There are a number of methods specific to strings in addition to the general sequence operations.
- **String processing involves search**.
- The **find(substring,start,end) method** returns the index location of the first occurrence of a specified substring. If not found, returns -1.

  Substring- The substring to search for.

  start – where to start the search. Default is 0.

  end – where to end the search. Default is 0.

```
>>> s3="chocolate“
>>> s3.find("c")
0
>>> s3.find("z")
-1
```

```
>>>s3.find(“c”,4,9)
-1

>>>s3.find(“c”,3,9)
3
```

Specific functions of string

- The **rfind(substring,start,end) method** returns the index location of the **last occurrence** of a specified substring. If not found, returns -1.
  Substring- The substring to search for.
  start – where to start the search. Default is 0.
  end – where to end the search. Default is 0.

```
>>> s3="chocolate“
>>> s3.rfind("c")
3
>>> s3.rfind("c",-6,-1)
3
>>> s3.rfind("c",4,9)
-1
>>>str1.rfind("c",0)
3
```

```
>>> s3.rfind("c",1,2)
-1
```

Specific functions of string

- The **strip(characters) method** removes characters from both left and right based on the argument.
- The **lstrip(characters) method** removes characters from left based on the argument.
- The **rstrip(characters) method** removes characters from right based on the argument.

```
>>> str1="          Welcome to Python class          "
>>> str1.strip()
'Welcome to Python class’
 >>> str1.lstrip()
 'Welcome to Python class          ‘
 >>> str1.rstrip()
 '          Welcome to Python class'
```

Specific functions of string

- >>> str1="madam"
  >>> str1.strip('m')
  'ada'
- >>> str1.lstrip('m')
  'adam'
- >>> str1.rstrip('m')
  'mada'
- >>> str1.rstrip('ma')
  'mad'
- >>> str1.rstrip('maz')
  'mad'
- >>> str1.rstrip('mjaz')
  'mad'

Specific functions of string

The **replace method** produces a new string with every occurrence of a given substring within the original string replaced with another.

- >>> s1.replace("a","b")
  'whbtsbpp'
- >>> s1.replace("a","b",1)
  'whbtsapp'
- >>> s1.replace("a","b",2)
  'whbtsbpp'
- >>> s1.replace("a","b",0)
  'whatsapp'
- >>> s1.replace("at","b",1)
  'whbsapp'
- >>> s1.replace("at","b",2)
  'whbsapp'

Specific functions of string

PES UNIVERSITY

The **title() method** returns a string where the first character in every word is upper case.

- >>> str1="people education society"
  >>> str1.title()
  'People Education Society'

The **capitalize() method** returns a string where the first character is upper case, and the rest is lower case.

```
>>> s1="Python"
>>> s1.capitalize()
'Python'
```

Specific functions of string

The **join() method** takes all items in an iterable and joins them into one string.
A string must be specified as the separator.

Example:-

```
>>> s1="abc"
>>> s2="xyz"
>>> s1.join(s2)
'xabcyabcz’
```

Specific functions of string

The **split()** method splits a string into a list.

Example:-

- >>> str1="When someone is lost, dare to help them find the way."
  >>> str1.split()
  ['When', 'someone', 'is', 'lost,', 'dare', 'to', 'help', 'them', 'find', 'the', 'way.']
- >>> str1.split("i")
  ['When someone ', 's lost, dare to help them f', 'nd the way.']
- >>> str1.split("i",1)
  ['When someone ', 's lost, dare to help them find the way.']
- >>> str1.split("i",1,2)
  Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  TypeError: split() takes at most 2 arguments (3 given)

Specific functions of string

- The **isspace()** method returns "True" if all characters in the string are whitespace characters, Otherwise, It returns "False".

```
>>> s1="  "
>>> s1.isspace()
True
```

```
>>> s1="543a"
>>> s1.isspace()
False
```

- The **ljust()** method will left align the string, using a specified character and returns a new string

```
>>> s3="chocolate"
>>> s3.ljust(20)
'chocolate           '
>>> s3.ljust(20,"#")
'chocolate###########'
```

Specific functions of string

- The **rjust()** method will right align the string, using a specified character and returns a new string

```
>>> s3="chocolate"
>>> s3.rjust(20)
'          chocolate’
>>> s3.rjust(20,"$")
'$$$$$$$$$$chocolate'
```

- The **center()** method will center align the string, using a specified character and returns a new string.

```
>>> s3="chocolate"
>>> s3.center(20)
'    chocolate     ‘
>>> s3.center(20,'*’)
'*****chocolate******'
```

Specific functions of string

- The **zfill()** method adds zeros (0) at the beginning of the string, until it reaches the specified length.

```
>>> s3.zfill(10)
'0chocolate'
>>> s3.zfill(20)
'00000000000chocolate'
```

- The **isnumeric()** method returns True if all the characters are numeric (0-9), else False.

```
>>> s1="543"
>>> s1.isnumeric()
True
```

```
>>> s1="543a"
>>> s1.isnumeric()
False
```

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Specific functions of string

Checking the Contents of a String			
`str.isalpha()`	Returns True if `str` contains only letters.	`s = 'Hello'`	`s.isalpha()` → `True`
		`s = 'Hello!'`	`s.isalpha()` → `False`
`str.isdigit()`	Returns True if `str` contains only digits.	`s = '124'`	`s.isdigit()` → `True`
		`s = '124A'`	`s.isdigit()` → `False`
`str.islower()` `str.isupper()`	Returns True if `str` contains only lower (upper) case letters.	`s = 'hello'`	`s.islower()` → `True`
		`s = 'Hello'`	`s.isupper()` → `False`
`str.lower()` `str.upper()`	Return lower (upper) case version of `str`.	`s = 'Hello!'`	`s.lower()` → `'hello!'`
		`s = 'hello!'`	`s.upper()` → `'HELLO!'`
Searching the Contents of a String			
`str.find(w)`	Returns the index of the first occurrence of `w` in *str*. Returns -1 if not found.	`s = 'Hello!'`	`s.find('l')` → `2`
		`s = 'Goodbye'`	`s.find('l')` → `-1`
Replacing the Contents of a String			
`str.replace(w,t)`	Returns a copy of *str* with all occurrences of `w` replaced with t.	`s = 'Hello!'`	`s.replace('H', 'J')` → `'Jello'`
		`s = 'Hello'`	`s.replace('ll', 'r')` → `'Hero'`
Removing the Contents of a String			
`str.strip(w)`	Returns a copy of *str* with all leading and trailing characters that appear in w removed.	`s = ' Hello! '` `s = 'Hello\n'`	`s.strip(' !')` → `'Hello'` `s.strip('\n')` → `'Hello'`
Splitting a String			
`str.split(w)`	Returns a list containing all strings in `str` delimited by w.	`s = 'Lu, Chao'`	`s.split(',')` → `['Lu', 'Chao']`

Functions to Try

PES UNIVERSITY

Try out the following functions:

- encode()
- decode()
- maketrans()
- translate()
- partition()

THANK YOU

Department of Computer Science and Engineering

Dr. Shylaja S S, Director, CCBD & CDSAML, PESU
Prof. Sindhu R Pai – sindhurpai@pes.edu