MASTER OF COMPUTER APPLICATIONS (MCA)

Post Graduate Department of Computer Science and Technology

Sardar Patel University

Vallabh Vidyanagar, Gujarat, INDIA

PS01CMCA51 [Python Programming]

Unit-II

MR. BHARAT B. PATEL (ASSOCIATE PROFESSOR)

(✉) bb_patel@spuvvn.edu

Sequence Type in Python

- A sequence is a positionally ordered collection of items.
- And you can refer to any item in the sequence by using its index number e.g., s[0] and s[1].
- In Python, the sequence index starts at 0, not 1. So the first element is s[0] and the second element is s[1]. If the sequence s has n items, the last item is s[n-1].
- Python has the following built-in sequence types:

lists	strings	tuples	range
bytearrays	bytes		

Sequence Type in Python

- Python classifies sequence types as mutable and immutable.
- The mutable sequence types are lists and bytearrays while the immutable sequence types are strings, tuples, range, and bytes.
- A sequence can be homogeneous or heterogeneous. In a homogeneous sequence, all elements have the same type. For example, strings are homogeneous sequences where each element is of the same type.
- Lists, however, are heterogeneous sequences where you can store elements of different types including integer, strings, objects, etc.
- In general, homogeneous sequence types are more efficient than heterogeneous in terms of storage and operations.

Basic Sequence types in Python

- **There are three basic sequence types**

  [1] lists

  [2] tuples

  [3] range objects

Common operations on sequences

- **The operations in the following table are supported by most sequence types, both mutable and immutable.**

Operation	Result
x in s	True if an item of s is equal to x, else False
x not in s	False if an item of s is equal to x, else True
s + t	the concatenation of s and t
s * n or n * s	equivalent to adding s to itself n times
s[i]	I^{th} item of s, origin 0
s[i:j]	slice of s from i to j
s[i:j:k]	slice of s from i to j with step k
len(s)	length of s
min(s)	smallest item of s
max(s)	largest item of s
s.index(x[, i[, j]])	index of the first occurrence of x in s (at or after index i and before index j)
s.count(x)	total number of occurrences of x in s

Python – List

- In Python, the list is a mutable sequence type.
- A list object contains one or more items of different data types in the square brackets [] separated by a comma.
- A list can contain unlimited data depending upon the limitation of your computer's memory.
- List items can be accessed using a zero-based index in the square brackets [].
- Indexes start from zero and increment by one for each item.
- Accessing an item using a large index than the list's total items would result in IndexError.
- A list can contain multiple inner lists as items that can be accessed using indexes.

Python – List

- All the list objects are the objects of the list class in Python.
- Use the list() constructor to convert from other sequence types such as tuple, set, dictionary, string to list.
- A list items can be iterated using the for loop.

Python – List

Characteristics / features of List [DOMAIN]

- Lists are dynamic.
- Lists are ordered.
- Lists are mutable.
- Lists can contain arbitrary objects.
- List elements can be accessible via index.
- Lists can be nested. (One list inside another)

Python – List

- The underlying type of a list is the list class.
- All the list objects are the objects of the list class in Python.
- Use the list() constructor to convert from other sequence types such as tuple, set, dictionary, string to list.
- A list items can be iterated using the for loop.
- We can use various operators like +, *, [], [:], in, not in, etc. to perform different operations on list object.
- Declaration of List :

```
lst1 = []
lst2 = list()
lst3 = [True,1,"1",23.50,[1,2,3],"Hello",]
```

Python – List

- **Iterating list using for loop :**

```
>>> lst3 = [True,1,"1",23.50,[1,2,3],"Hello",]
>>> for ele in lst3:
        print ( ele)
```

- **Output :**

```
True
1
1
23.5
[1, 2, 3]
Hello
>>>
```

Python List Methods

List Method	Description
list.append()	Adds a new item at the end of the list.
list.clear()	Removes all the items from the list and make it empty.
list.copy()	Returns a shallow copy of a list.
list.count()	Returns the number of times an element occurs in the list.
list.extend()	Adds all the items of the specified iterable (list, tuple, set, dictionary, string) to the end of the list.
list.index()	Returns the index position of the first occurrence of the specified item. Raises a ValueError if there is no item found.
list.insert()	Inserts an item at a given position.

Python List Methods

List Method	Description
list.pop()	Returns an item from the specified index position and also removes it from the list. If no index is specified, the list.pop() method removes and returns the last item in the list.
list.remove()	Removes the first occurrence of the specified item from the list. It the specified item not found then throws a ValueError.
list.reverse()	Reverses the index positions of the elements in the list. The first element will be at the last index, the second element will be at second last index and so on.
list.sort()	Sorts the list items in ascending, descending, or in custom order.

Python – Tuple

- Tuple is an immutable (unchangeable) collection of elements of different data types.
- It is an ordered collection, so it preserves the order of elements in which they were defined.
- Tuples are defined by enclosing elements in parentheses (), separated by a comma. However, it is not necessary to enclose the tuple elements in parentheses. The tuple object can include elements separated by a comma without parentheses.
- Each element in the tuple is accessed by the index in the square brackets [].
- An index starts with zero and ends with (number of elements - 1).
- The tuple supports negative indexing also, the same as list type. The negative index for the first element starts from -number of elements and ends with -1 for the last element.

Python – Tuple

- If the element at the specified index does not exist, then the error "index out of range" will be thrown.
- Tuple is unchangeable. So, once a tuple is created, any operation that seeks to change its contents is not allowed.
- A tuple items can be iterated using the for loop.

Python – Tuple

Characteristics / features of Tuple [INDIAO]

- Tuples are immutable.
- Tuples can be nested. (tuple inside another)
- Tuples are dynamic.
- Tuples elements can be accessible via index.
- Tuples can contain arbitrary objects.
- Tuples are ordered.

Python – Tuple

- The underlying type of a tuple is the tuple class.
- All the list objects are the objects of the tuple class in Python.
- The tuple() constructor is used to convert any iterable to tuple type.
- A tuple items can be iterated using the for loop.
- We can use various operators like +, *, [], [:], in, not in, etc. to perform different operations on tuple object.
- Declaration of Tuple :

```
tpl1 = ()
tpl2 = tuple()
tpl3 = (True,1,"1",23.50,[1,2,3],"Hello",)
tpl4 = True,1,"1",23.50,[1,2,3],"Hello“,
```

Python – Tuple

- **Iterating tuple using for loop :**

```
>>>
>>> tpl1 = 1,"1",False, ( 1,2)   ,[5,6],
>>> for ele in tpl1:
            print ( ele)
```

- **Output :**

```
1
1
False
 ( 1, 2)
[5, 6]
```

Python Tuple Methods

List Method	Description
tuple.count()	Returns the number of times an element occurs in the tuple.
tuple.index()	Returns the index position of the first occurrence of the specified item. Raises a ValueError if there is no item found.

Tuple's count() method

- The count() method of Tuple returns the number of times the given element appears in the tuple.
- Syntax : tuple.count(element)

  element - the element that is to be counted.

- Example-1:

```
>>> tpl1= ( 11,13,14,15,11,10+1)
>>> tpl1.count ( 11)
3
>>>
```

Example-2 :

```
>>> tpl2 = 11, [11,12], ( 9,10)  ,True,False,True,False,  ( 10,9)
>>> tpl2.count ( True or False)
2
>>> tpl2.count (  ( 9,10)  )
1
```

Tuple's index() method

- The Index() method returns the first occurrence of the given element from the tuple.
- Syntax : tuple.index(element, start, end)

  element - The element to be searched.

  start (Optional) - The starting index from where the searching is started

  end (Optional) - The ending index till where the searching is done
- Example-1:

```
>>> tpl1= ( 1,2,1+2,3,4,5,6,3,3,3)
>>> tpl1.index ( 3)
2
```

This method raises a ValueError if the element is not found in the tuple.

Example-2 :

```
>>> tpl2 = 11, [11,12], ( 9,10)  ,True,False,True,False,  ( 10,9)
>>> tpl2.index ( True)
3
>>> tpl2.index ( "True")
Traceback  ( most recent call last)  :
  File "<pyshell#49>", line 1, in <module>
    tpl2.index ( "True")
ValueError: tuple.index ( x)  : x not in tuple
```

Python – Range

What Is Range In Python?

- ***It is an in-built function in Python which returns a sequence of numbers starting from 0 and increments to 1 until it reaches a specified number.***
- **The most common use of range function is to iterate sequence type.**
- **It is most commonly used in for and while loops.**
- **The range() constructor method returns an object of the range class, which is an immutable sequence type.**
- **Syntax :** **range(stop)**
- **range(start, stop, [step])**
- **The range() method returns the immutable sequence numbers between the specified start and the stop parameter, increment by step parameter.**

Python – Range

- **start** : integer starting from which the sequence of integers is to be returned
- **stop** : integer before which the sequence of integers is to be returned. The range of integers end at stop – 1.
- **Step** : integer value which determines the increment between each integer in the sequence.
- **Return Value:** The range() returns an immutable sequence object of numbers.

Python – Range

- **Examples :**

```
>>> range ( 10)
range ( 0, 10)
>>> range ( 1,10,1)
range ( 1, 10)
>>> for ele in range ( 3)   :
            print ( ele)


0
1
2
```

```
>>> for ele in range ( 1,3,1)   :
            print ( ele)


1
2
>>>
```

Python – String

- **In Python, string is an immutable sequence data type.**
- ***String is the sequence of Unicode characters wrapped inside single, double, or triple quotes.***
- **A character is simply a symbol.**
- **Strings are Arrays Like many other popular programming languages, strings in Python are arrays of bytes representing Unicode characters. However, Python does not have a character data type, a single character is simply a string with a length of 1. Square brackets can be used to access elements of the string.**
- **Strings are immutable. This means that elements of a string cannot be changed once they have been assigned.**
- **We cannot delete or remove characters from a string.**

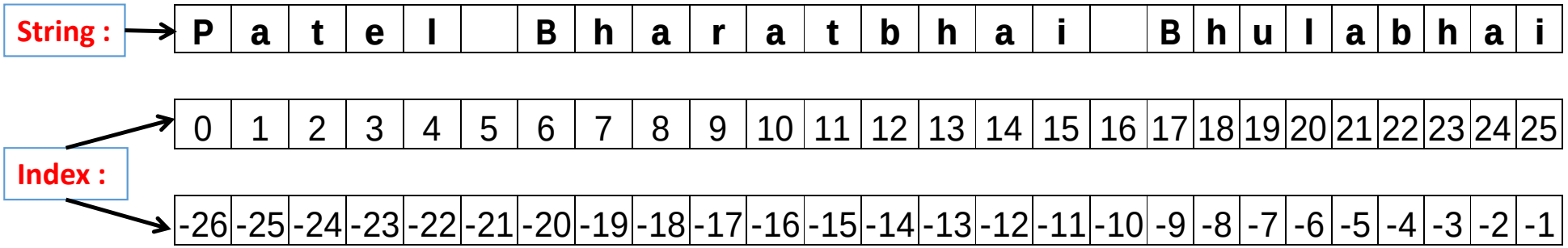
Python – String

- **The followings are valid string literals in Python.**

```
>>> # Examples of string in Python
>>> 'I am string in Python'
'I am string in Python'
>>> "I am string in Python"
'I am string in Python'
>>> ""I am string in Python""
'I am string in Python'
>>> """I am string in Python"""
'I am string in Python'
>>>
```

Python - String

- In Python, string value is represented using a pair of single quotes (' ') or double quotes (" "). [in case of docstring , it is represented using ''' or """ (three times single or double quotes).
- For example, 'Python' "Python"



- To access substrings, use the square brackets for slicing along with the index or indices to obtain your substring. For example **st[0]** or **st[1:5]**

String methods

Case
- capitalize()
- casefold()
- lower()
- swapcase()
- title()
- upper()

Check
- isalnum()
- isalpha()
- isdecimal()
- isdigit()
- isidentifier()
- islower()
- isnumeric()
- isprintable()
- isspace()
- istitle()
- isupper()

Format
- center()
- format()
- format_map()

Pattern
- endswith()
- startswith()

Search
- find()
- index()
- rfind()
- rindex()

Partition
- partition()
- rpartition()
- split()
- splitlines()

Trim
- lstrip()
- rstrip()
- strip()

Miscellaneous
- zfill()
- expandtabs()
- encode()
- join()
- count()
- replace()

Python String Methods

String Method	Description
str.capitalize()	Returns the copy of the string with its first character capitalized and the rest of the letters are in lowercased.
str.casefold()	Returns a lowered case string. It is similar to the lower() method, but the casefold() method converts more characters into lower case.
str.center()	Returns a new centered string of the specified length, which is padded with the specified character. The deafult character is space.
str.count()	Searches (case-sensitive) the specified substring in the given string and returns an integer indicating occurrences of the substring.
str.endswith()	Returns True if a string ends with the specified suffix (case-sensitive), otherwise returns False.
str.find()	Returns the index of the first occurence of a substring in the given string (case-sensitive). If the substring is not found it returns -1.
str.index()	Returns the index of the first occurence of a substring in the given string.

Python String Methods

String Method	Description
str.isalnum()	Returns True if all characters in the string are alphanumeric (either alphabets or numbers). If not, it returns False.
str.isalpha()	Returns True if all characters in a string are alphabetic (both lowercase and uppercase) and returns False if at least one character is not an alphabet.
str.isascii()	Returns True if the string is empty or all characters in the string are ASCII.
str.isdecimal()	Returns True if all characters in a string are decimal characters. If not, it returns False.
str.isdigit()	Returns True if all characters in a string are digits or Unicode char of a digit. If not, it returns False.
str.isidentifier()	Checks whether a string is valid identifier string or not. It returns True if the string is a valid identifier otherwise returns False.
str.islower()	Checks whether all the characters of a given string are lowercased or not. It returns True if all characters are lowercased and False even if one character is uppercase.

Python String Methods

String Method	Description
str.isnumeric()	Checks whether all the characters of the string are numeric characters or not. It will return True if all characters are numeric and will return False even if one character is non-numeric.
str.isprintable()	Returns True if all the characters of the given string are Printable. It returns False even if one character is Non-Printable.
str.isspace()	Returns True if all the characters of the given string are whitespaces. It returns False even if one character is not whitespace.
str.istitle()	Checks whether each word's first character is upper case and the rest are in lower case or not. It returns True if a string is titlecased; otherwise, it returns False. The symbols and numbers are ignored.
str.isupper()	Returns True if all characters are uppercase and False even if one character is not in uppercase.
str.join()	Returns a string, which is the concatenation of the string (on which it is called) with the string elements of the specified iterable as an argument.

Python String Methods

List Method	Description
str.ljust()	Returns the left justified string with the specified width. If the specified width is more than the string length, then the string's remaining part is filled with the specified fillchar.
str.lower()	Returns the copy of the original string wherein all the characters are converted to lowercase.
str.lstrip()	Returns a copy of the string by removing leading characters specified as an argument.
str.partition()	Splits the string at the first occurrence of the specified string separator sep argument and returns a tuple containing three elements, the part before the separator, the separator itself, and the part after the separator.
str.replace()	Returns a copy of the string where all occurrences of a substring are replaced with another substring.
str.rfind()	Returns the highest index of the specified substring (the last occurrence of the substring) in the given string.
str.rindex()	Returns the index of the last occurrence of a substring in the given string.

Python String Methods

List Method	Description
str.rjust()	Returns the right justified string with the specified width. If the specified width is more than the string length, then the string's remaining part is filled with the specified fill char.
str.rpartition()	Splits the string at the last occurrence of the specified string separator sep argument and returns a tuple containing three elements, the part before the separator, the separator itself, and the part after the separator.
str.rsplit()	Splits a string from the specified separator and returns a list object with string elements.
str.rstrip()	Returns a copy of the string by removing the trailing characters specified as argument.
str.split()	Splits the string from the specified separator and returns a list object with string elements.

Python String Methods

List Method	Description
str.startswith()	Returns True if a string starts with the specified prefix. If not, it returns False.
str.strip()	Returns a copy of the string by removing both the leading and the trailing characters.
str.swapcase()	Returns a copy of the string with uppercase characters converted to lowercase and vice versa. Symbols and letters are ignored.
str.title()	Returns a string where each word starts with an uppercase character, and the remaining characters are lowercase.
str.upper()	Returns a string in the upper case. Symbols and numbers remain unaffected.

Python – slice()

- A substring of a string is called a slice.
- Selecting a slice is same as selecting a character.
- Python slice() function returns a slice object.
- A sequence of objects of any type(string, bytes, tuple, list or range) can be sliced using slice() method.
- It is possible to return a range of characters from a string using the slice operation.
- The slice operator (:) returns the part of the string from the start (including) to end (excluding).

Python – slice()

- **Syntax:**

  slice(stop)

  slice(*start, stop, step*)

  where *start* - starting index where the slicing of object starts.

  ***stop* - ending index where the slicing of object stops.**

  ***step* - It is an optional argument that determines the increment between each index for slicing.**

- **Return Type : Returns a sliced object containing elements in the given range only.**

```
>>> s1="GDCST"
>>> s11=slice ( 0,2)
>>> s12=slice ( 4,2,-1)
>>> s1[s11]
'GD'
>>> s1[s12]
'TS'
>>>
```

Python – slice()

- **Example :**

(Using slice() method)

```
>>> s1="GDCST"
>>> s11=slice ( 0,2)
>>> s12=slice ( 4,2,-1)
>>> s1[s11]
'GD'
>>> s1[s12]
'TS'
>>>
```

(Using : Operator)

```
>>> s1="GDCST"
>>> s1[0:2]
'GD'
>>> s1[4:2:-1]
'TS'
>>> s1[::-1]
'TSCDG'
```

This expression reverse the entire string (easiest way to reverse a string in Python)