**Name:**  Anushka Dashrath Adhav

**Roll No: SE2204**

**Class:** SY BCA Science - E2

**String Methods**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.No.** | **Methods** | **Description** | **Code & Output** |
| 1. | **Capitalize()** | Capitalizes first letter of string | **Code:** text = "anushka adhav"  capitalized\_text = text.capitalize()  print(capitalized\_text)  **Output:**  Anushka adhav |
| 2. | **count(str, beg= 0, end=len(string))** | Counts how many times str occurs in  string or in a substring of string if starting  index beg and ending index end are  given. | **Code:**text = "Anushka adhav,adhav anushka"  substring = "anushka"  count\_result = text.count(substring, 0, 20)  print(count\_result)  count\_result2 = text.count(substring, 7, len(text))  print(count\_result2)    **Output:**  0  1 |
| 3. | **endswith(suffix, beg=0,**  **end=len(string))** | Determines if string or a substring of  string (if starting index beg and ending  index end are given) ends with suffix;  returns true if so and false otherwise. | **Code:**  text = "Anushka ,Adhav!"  result1 = text.endswith('Adhav!')  print(result1)  result2=text.endswith('Adhav!', 0, 10)  print(result2)  **Output:**  True  False |
| 4. | **isalnum()** | Returns true if string has at least 1  character and all characters are  alphanumeric and false otherwise. | **Code:**  text1 = "Anushka2204"  result1 = text1.isalnum()  print(result1)  text2 = "Anushka 2004"  result2 = text2.isalnum()  print(result2)  **Output:**  True  False |
| 5. | **isalpha()** | Returns true if string has at least 1  character and all characters are alphabetic  and false otherwise. | **Code:**  text1 = "Anushka"  result1 = text1.isalpha()  print(result1)  text2 = "Anushka 2004"  result2 = text2.isalpha()  print(result2)  **Output:**  True  False |
| 6. | **isdigit()** | Returns true if string contains only digits  and false otherwise. | **Code:**  text1 = "2004"  result1 = text1.isdigit()  print(result1)  text2 = "Anushka2004"  result2 = text2.isdigit()  print(result2)  **Output:**  True  False |
| 7. | **islower()** | Returns true if string has at least 1 cased  character and all cased characters are in  lowercase and false otherwise. | **Code:**  text1 = "anushka"  result1 = text1.islower()  print(result1)  text2 = "Anushka"  result2 = text2.islower()  print(result2)  **Output:**  True  False |
| 8. | **isnumeric()** | Returns true if a unicode string contains  only numeric characters and false  otherwise. | **Code:**  text1 = "2004"  result1 = text1.isnumeric()  print(result1)  text2 = "Anushka2004"  result2 = text2.isnumeric()  print(result2)    **Output:**  True  False |
| 9. | **isspace()** | Returns true if string contains only  whitespace characters and false  otherwise. | **Code:**  text1 = " "  result1 = text1.isspace()  print(result1)  text2 = "Anushka2004"  result2 = text2.isspace()  print(result2)  **Output:**  True  False |
| 10. | **istitle()** | Returns true if string is properly  "titlecased" and false otherwise. | **Code:**  text1 = "Anushka Adhav"  result1 = text1.istitle()  print(result1)  text2 = "anushka adhav"  result2 = text2.istitle()  print(result2)  **Output:**  True  False |
| 11. | **isupper()** | Returns true if string has at least one  cased character and all cased characters  are in uppercase and false otherwise. | **Code:**  text1 = "ANUSHKA"  result1 = text1.isupper()  print(result1)  text2 = "anushka"  result2 = text2.isupper()  print(result2)  **Output:**  True  False |
| 12. | **join(seq)** | Merges (concatenates) the string  representations of elements in sequence  seq into a string, with separator string. | **Code:**  words = ["Anushka", "Dashrath", "Adhav"]  result1 = " ".join(words)  print(result1)  **Output:**  Anushka Dashrath Adhav |
| 13. | **len(string)** | Returns the length of the string. | **Code:**  string1= ["Anushka","Dashrath","Adhav"]  result1 = len(string1)  print(result1)  **Output:**  3 |
| 14. | **ljust(width[, fillchar])** | Returns a space-padded string with the  original string left-justified to a total of  width columns. | **Code:**  text1 = "Anushkase2204"  result1 = text1.ljust(20,'\*')  print(result1)  **Output:**  Anushkase2204\*\*\*\*\*\*\* |
| 15. | **lower()** | Converts all uppercase letters in string to  lowercase. | **Code:**  string1= AnushkaDashrathAdhavSE2004  result1 = len(string1)  print(result1)  **Output:**  anushkadashrathadhavse2004 |
| 16. | **lstrip()** | Removes all leading whitespace in string. | **Code:**  text1 = " Anushka Adhav"  result1 = text1.lstrip()  print(result1)  **Output:**  Anushka Adhav |
| 17. | **maketrans()** | Returns a translation table to be used in  translate function. | **Code:**  original\_string = "anushka adhav"  translation\_table = str.maketrans("aa", "AA")  translated\_string = original\_string.translate(translation\_table)  print(translated\_string)  **Output:**  AnushkA AdhAv |
| 18. | **max(str)** | Returns the max alphabetical character  from the string str. | **Code:**  string1="AnushkaDashrathAdhav"  result1 =max(string1)  print(result1)  **Output:**  v |
| 19. | **min(str)** | Returns the min alphabetical character  from the string str. | **Code:**  string1="AnushkaDashrathAdhav"  result1 =min(string1)  print(result1)  **Output:**  A |
| 20. | **replace(old, new [, max])** | Replaces all occurrences of old in string  with new or at most max occurrences if  max given. | **Code:**  text1 = "Anushka Dashrath Adhav"  result1 = text1.replace("Adhav", "Raut")  print(result1)  **Output:**  Anushka Dashrath Raut |
| 21. | **rfind(str, beg=0,end=len(string))** | Same as find(), but search backwards in  string. | **Code:**  text1 ="Anushka Dashrath adhav"  result3 = text1.rfind("adh")  print(result3)  **Output:**  17 |
| 22. | **rjust(width,[, fillchar])** | Returns a space-padded string with the  original string right-justified to a total of  width columns. | **Code:**  text1 = "Anushkase2204"  result1 = text1.rjust(20,'\*')  print(result1)  **Output:**  \*\*\*\*\*\*\*Anushkase2204 |
| 23. | **rstrip()** | Removes all trailing whitespace of string. | **Code:**  text1 = "Anushka Adhav "  result1 = text1.rstrip()  print(result1)  **Output:**  Anushka Adhav |
| 24. | **split(str="", num=string.count(str))** | Splits string according to delimiter str  (space if not provided) and returns list of  substrings; split into at most num  substrings if given. | **Code:**  text2 = "Anushka Dashrath Adhav se2204"  result2 = text2.split(" ", 1)  print(result2)  **Output:**  ['Anushka', 'Dashrath Adhav se2204'] |
| 25. | **splitlines( num=string.count('\n'))** | Splits string at all (or num) NEWLINEs  and returns a list of each line with  NEWLINEs removed. | **Code:**  text2 = "AnushkaDashrath\nAdhav\nse2204"  result2 = text2.splitlines()  print(result2)  **Output:**  ['AnushkaDashrath', 'Adhav', 'se2204'] |
| 26. | **swapcase()** | Inverts case for all letters in string. | **Code:**  text1 = "Anushka Adhav"  result1 = text1.swapcase()  print(result1)  **Output:**  aNUSHKA aDHAV |
| 27. | **title()** | Returns "titlecased" version of string, that  is, all words begin with uppercase and the  rest are lowercase. | **Code:**  text1 = "anushka adhav"  result1 = text1.title()  print(result1)  **Output:**  Anushka Adhav |
| 28. | **translate(table, deletechars="")** | Translates string according to translation  table str(256 chars), removing those in  the del string. | **Code:**  text3 = "anushka dashrath adhav"  translation\_table = str.maketrans("aeiou", "12345")  result3 = text3.translate(translation\_table)  print(result3)  **Output:**  1n5shk1 d1shr1th 1dh1v |
| 29. | **upper()** | Converts lowercase letters in string to  uppercase. | **Code:**  text1 = "anushka adhav"  result1 = text1.upper()  print(result1)  **Output:**  ANUSHKA ADHAV |
| 30. | **zfill (width)** | Returns original string left padded with zeros to a total of width characters;  intended for numbers, zfill() retains any  sign given (less one zero). | **Code:**  text1 = "anushka"  result1 = text1.zfill(10)  print(result1)  **Output:**  000anushka |
| 31. | **isdecimal()** | Returns true if a unicode string contains  only decimal characters and false  otherwise. | **Code:**  text1 = "2004"  result1 = text1.isdecimal()  print(result1)  text2 = "anushka"  result2 = text2.isdecimal()  print(result2)  **Output:**  True  False |

**Tuple Functions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Serial No.** | **Function** | **Description** | **Code & Output** |
| 1. | **all()** | Return True if all elements of the tuple are true (or if the tuple is empty). | **Code:**  tuple1 = (1, 2, 3, 4, 5)  tuple2 = (0, 1, 2, 3)  print(all(tuple1))  print(all(tuple2))  **Output:**  True  False |
| 2. | **any()** | Return True if any element of the tuple is true. If the tuple is empty, return False. | **Code:**  tuple1 = (1, 2, 3, 4, 5)  tuple2 = ()  print(any(tuple1))  print(any(tuple2))  **Output:**  True  False |
| 3. | **enumerate()** | Return an enumerate object. It contains the index and value of all the items of tuple as pairs. | **Code:**  my\_tuple = ('apple', 'banana', 'cherry')  for index, value in enumerate(my\_tuple, start = 1):  print(index, value)  **Output:**  1 apple  2 banana  3 cherry |
| 4. | **len()** | Return the length (the number of items) in the tuple. | **Code:**  my\_tuple = ('apple', 'banana', 'cherry')  print(len(my\_tuple))  **Output:**  3 |
| 5. | **max()** | Return the largest item in the tuple. | **Code:**  my\_tuple = ('apple', 'banana', 'cherry')  print(max(my\_tuple))  **Output:**  cherry |
| 6. | **min()** | Return the smallest item in the tuple. | **Code:**  my\_tuple = ('apple', 'banana', 'cherry')  print(max(my\_tuple))  **Output:**  apple |
| 7. | **sorted()** | Take elements in the tuple and return a new sorted list (does not sort the tuple itself). | **Code:**  my\_tuple = (3, 1, 2, 5, 4)  print(sorted(my\_tuple))  **Output:**  [1, 2, 3, 4, 5] |
| 8. | **sum()** | Return the sum of all elements in the tuple. | **Code:**  my\_tuple = (3, 1, 2, 5, 4)  print(sum(my\_tuple))  **Output:**  15 |
| 9. | **tuple()** | Convert an iterable (list, string, set, dictionary) to a tuple. | **Code:**  string = "Anushka"  print(tuple(string))  **Output:**  ('A’, 'n', 'u’, 's’, 'h’,’k’,’a’) |