[4.2] Strings

June 26, 2022

1 Python Strings

Concept: Python provides several ways to access the individual characters in a string. Strings also have methods that allow you to perform operations on them.

Little more about strings:

- A string is a sequence (a positionally ordered collection) of other objects or elements.
- A string maintains a left-to-right order among the contained items.
- Items in a string are stored and fetched by their relative position.
- Stings are immutable in Python – they cannot be changed in place after they are created. In simple words, immutable objects can never be overwritten, e.g. we can't change a string by assigning to one of its positions, but we can always build a new string and assign it to the same name.

[1]:
```python
# Lets create a string
name = "GIFT University"
print(name)
```

```
GIFT University
```

2 Accessing the Individual Characters in a String

Some programming tasks require that you access the individual characters in a string. For example, you are probably familiar with websites that require you to set up a password. For security reasons, many sites require that your password have at least one uppercase letter, at least one lowercase letter, and at least one digit. When you set up your password, a program examines each character to ensure that the password meets these qualifications.

2.1 Iterating over a String with the for Loop

One of the easiest ways to access the individual characters in a string is to use the for loop.

Here is the general format:

for variable in string:

```
statement

statement
```

```
etc.
```

[2]:
```python
name = "GIFT University"
for ch in name:
    print(ch,end=" ")
```

```
G I F T   U n i v e r s i t y
```

2.2 Indexing

Another way that you can access the individual characters in a string is with an index. Each character in a string has an index that specifies its position in the string. Indexing starts at 0, so the index of the first character is 0, the index of the second character is 1, and so forth.

You can use an index to retrieve a copy of an individual character in a string.

[3]:
```python
my_string = 'Roses are red'
ch = my_string[6]
print(ch)
```

```
a
```

2.3 IndexError Exceptions

An *IndexError* exception will occur if you try to use an index that is out of range for a particular string. For example, the string 'Boston' has 6 characters, so the valid indexes are 0 through 5. **(The valid negative indexes are −1 through −6.)** The following is an example of code that causes an IndexError exception.

[4]:
```python
city = 'Boston'
print(city[6])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_15088/3374377224.py in <module>
      1 city = 'Boston'
----> 2 print(city[6])

IndexError: string index out of range
```

2.4 The len Function

The len function can also be used to get the length of a string. The following code demonstrates:

[5]:
```python
city = 'Boston'
size = len(city)
print(size)
```

```
6
```

3 String Concatenation

A common operation that performed on strings is concatenation, or appending one string to the end of another string.The + operator produces a string that is the combination of the two strings used as its operands.

[6]:
```
first_name = "Ali "
last_name = "Babar"
full_name = first_name + last_name
print(full_name)
```

```
Ali Babar
```

4 Strings Are Immutable

In Python, strings are immutable, which means that once they are created, they cannot be changed. Some operations, such as concatenation, give the impression that they modify strings, but in reality they do not.

Because strings are immutable, you cannot use an expression in the form string[index] on the left side of an assignment operator. For example, the following code will cause an error:

[7]:
```
# Assign 'Bill' to friend.
friend = 'Bill'
# Can we change the first character to 'J'?
friend[0] = 'J' # No, this will cause an error!
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_15088/882698649.py in <module>
      2 friend = 'Bill'
      3 # Can we change the first character to 'J'?
----> 4 friend[0] = 'J' # No, this will cause an error!

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

The last statement in this code will raise an exception because it attempts to change the value of the first character in the string 'Bill'.

5 String Slicing

CONCEPT: You can use slicing expressions to select a range of characters from a string

A slice is a span of items that are taken from a sequence. When you take a slice from a string, you get a span of characters from within the string. String slices are also called substrings.

To get a slice of a string, you write an expression in the following general format:

```
string[start : end]
```

```
[8]: full_name = 'Patty Lynn Smith'
     middle_name = full_name[6:10]
     print(middle_name)
```

```
Lynn
```

If you leave out the start index in a slicing expression, Python uses 0 as the starting index. Here is an example:

```
[9]: full_name = 'Patty Lynn Smith'
     first_name = full_name[:5]
     print(first_name)
```

```
Patty
```

If you leave out the end index in a slicing expression, Python uses the length of the string as the end index. Here is an example:

```
[10]: full_name = 'Patty Lynn Smith'
      last_name = full_name[11:]
      print(last_name)
```

```
Smith
```

What do you think the following code will assign to the my_string variable?

```
[11]: full_name = 'Patty Lynn Smith'
      my_string = full_name[:]
      print(my_string)
```

```
Patty Lynn Smith
```

The second statement assigns the entire string ‘Patty Lynn Smith’ to my_string. The statement is equivalent to:

```
[12]: my_string = full_name[0 : len(full_name)]
      print(my_string)
```

```
Patty Lynn Smith
```

Slicing expressions can also have step value, which can cause characters to be skipped in the string. Here is an example of code that uses a slicing expression with a step value:

```
[13]: letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
      print(letters[0:26:2])
```

```
ACEGIKMOQSUWY
```

6 Testing, Searching, and Manipulating Strings

6.1 Testing Strings with in and not in

In Python you can use the in operator to determine whether one string is contained in another string.

```
string1 in string2
```

string1 and string2 can be either string literals or variables referencing strings. The expression returns true if string1 is found in string2. For example, look at the following code:

[14]:
```python
text = 'The GIFT University Gujranwala'
if 'GIFT' in text:
    print('The string "GIFT" was found.')
else:
    print('The string "GIFT" was not found.')
```

```
The string "GIFT" was found.
```

You can use the not in operator to determine whether one string is not contained in another string. Here is an example:

[15]:
```python
names = 'Bill Joanne Susan Chris Juan Katie'
if 'Pierre' not in names:
    print('Pierre was not found.')
else:
    print('Pierre was found.')
```

```
Pierre was not found.
```

6.2 String Methods

We will discuss several string methods for performing the following types of operations: * Testing the values of strings * Performing various modifications * Searching for substrings and replacing sequences of characters

Here is the general format of a string method call:

```
stringvar.method(arguments)
```

6.3 String Testing Methods

The string methods shown in Fig:2 test a string for specific characteristics. For example, the isdigit method returns true if the string contains only numeric digits. Otherwise, it returns false.

[16]:
```python
string1 = '1200'
if string1.isdigit():
    print(string1, 'contains only digits.')
else:
    print(string1, 'contains characters other than digits.')
```

```
1200 contains only digits.
```

6.4 Modification Methods

Although strings are immutable, meaning they cannot be modified, they do have a number of methods that return modified versions of themselves.

7 The Repetition Operator

The repetition operator(*) works with strings as well. Here is the general format:

```
string_to_copy * n
```

[17]:
```
my_string = 'w' * 5
print(my_string)
```

```
wwwww
```

[18]:
```
print('Hello ' * 5)
```

```
Hello Hello Hello Hello Hello
```

8 Splitting a String

Strings in Python have a method named split that returns a list containing the words in the string.

[19]:
```
# This program demonstrates the split method.
def main():
    # Create a string with multiple words.
    my_string = 'One two three four'
    # Split the string.
    word_list = my_string.split()
    # Print the list of words.
    print(word_list)
# Call the main function
main()
```

```
['One', 'two', 'three', 'four']
```

[20]:
```
date_string = '11/26/2014'
```

If you want to break out the month, day, and year as items in a list, you can call the split method using the '/' character as a separator, as shown here:

[21]:
```
date_list = date_string.split('/')
print(date_list)
```

```
['11', '26', '2014']
```

9 Exercise

9.1 Sum of Digits in a String

Write a program that asks the user to enter a series of single-digit numbers with nothing separating them. The program should display the sum of all the single digit numbers in the string. For example, if the user enters 2514, the method should return 12, which is the sum of 2, 5, 1, and 4.

9.2 Reverse Incremental String Slicing

Sometimes, while working with Pythonn strings, we can have a problem in which we need to perform the slice and print of strings in reverse order. This can have application in day-day programming. So in this exercise write a program that reverse the string in incremental order.

For Example:

```
Orignal String : GIFT

Incremental reverse strings: ['T','TF','TFI','TFIG']
```

9.3 Alphabetic Telephone Number Translator

Many companies use telephone numbers like 555-GET-FOOD so the number is easier for their customers to remember. On a standard telephone, the alphabetic letters are mapped to numbers in the following fashion:

Letters	Number
A, B, and C	2
D, E, and F	3
G, H, and I	4
J, K, and L	5
M, N, and O	6
P, Q, R, and S	7
T, U, and V	8
W, X, Y, and Z	9

Write a program that asks the user to enter a 10-character telephone number in the format XXX-XXX-XXXX. The application should display the telephone number with any alphabetic characters that appeared in the original translated to their numeric equivalent. For example, if the user enters 555-GET-FOOD the application should display 555-438-3663.

9.4 Password Validation

Assume that you are writing a program for a local software house. It is a password checker program. Two Python built-in functions you'll need are ord(character) and chr(asciiNumber). ord(character) takes a character and returns its ascii value. In opposite fashion, chr(asciiValue) takes an ascii value and returns a character mapped on that ascii value. Following is ascii chart.

Fig:5 ASCII Code Table

That particular software house sets the criteria of password as: A password is valid if and only if it has: * At least one lower case letter * At least one upper case letter * At least one special characters from the above given image. * At least one digit

All other passwords are invalid. Use the functions described above to create this program.Summing up the program requirements:

1. takes input as password from user
2. Apply the described tests
3. And prints whether the password is valid or invalid.

9.5 Sentence Capitalizer

Write a program with a function that accepts a string as an argument and returns a copy of the string with the first character of each sentence capitalized. For instance, if the argument is "hello. my name is Joe. what is your name?" the function should return the string "Hello. My name is Joe. What is your name?" The program should let the user enter a string and then pass it to the function. The modified string should be displayed.

9.6 Pig Latin

Write a program that accepts a sentence as input and converts each word to "Pig Latin." In one version, to convert a word to Pig Latin you remove the first letter and place that letter at the end of the word. Then you append the string "ay" to the word. Here is an example:

```
English: I SLEPT MOST OF THE NIGHT
Pig Latin: IAY LEPTSAY OSTMAY FOAY HETAY IGHTNAY
```