



Chapter 3: String

Prepared by: Hanan Hardan

Python String

- Strings in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as "hello".

- Assign String to a Variable

```
a = "Hello"
```

```
Print(a)
```

Python String

- You can assign a multiline string to a variable by using three quotes:
- You can use three double quotes:

Example:

```
a = """Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua. """
```

- Or three single quotes:

Example

```
a = "Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."  
print(a)
```

Note: in the result, the line breaks are inserted at the same position as in the code.

Str Class

- `str()` creates an empty string.

```
s=str()
```

Can also assign using:

```
strValue = " "
```

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.

Note: remember that the first character has the position 0

```
a = "Hello, World!"  
print(a[1])
```

Looping Through a String

- Since strings are arrays, we can loop through the characters in a string, with a for loop.

Example

Loop through the letters in the word "banana":

```
for x in "banana":  
    print(x)
```

String Length

- To get the length of a string, use the `len()` function.

Example:

```
a = "Hello, World!"
```

```
print(len(a))
```

Check String

- To check if a certain phrase or character is present in a string, we can use the keyword `in`.

Example: Check if "free" is present in the following text:

```
txt = "The best things in life are free! "
```

Solution:

```
txt = "The best things in life are free!"
```

```
if "free" in txt:
```

```
    print("Yes, 'free' is present.")
```

Check if NOT

- To check if a certain phrase or character is NOT present in a string, we can use the keyword `not in`.

Example: Check if "expensive" is NOT present in the following text:

```
txt = "The best things in life are free!"
```

Solution:

```
txt = "The best things in life are free!"
```

```
if "expensive" not in txt:
```

```
    print("Yes, 'expensive' is NOT present.")
```

strings Generating a New string

Syntax	Semantics
<p>dataA + dataB</p> <p>Example:</p> <pre>x='ABC' y='DEF' z=x+y print(z)</pre>	<p>Generates a third string that is dataB items added to the end of dataA.</p>

strings Generating a New string (continued)

Syntax	Semantics
<pre>data * k Example: x='ABC' z=x*3 print(z)</pre>	<p>Generates a new list of data items repeated k times. 'ABC' * 3 becomes 'ABCABCABC'</p>

strings Generating a New string (continued)

Syntax	Semantics
<pre>dataA += dataB Example: x='ABC' y='DEF' x+=y print(x)</pre>	<p>dataA becomes dataA with dataB added to the end. This is the same as dataA = dataA + dataB</p>

strings Generating a New string (continued)

Syntax	Semantics
<pre>data *= k Example: x='ABC' x*=3 print(x)</pre>	<p>data becomes data k times. This is the same as <code>data = data * k</code></p>

Example 1

Write a Python program find the common values that appear in two given strings. Sample Output:

Original strings:

Python3

Python2.7

Intersection of two said String:

Python

Example 1

```
def intersection_of_two_string(str1, str2):
    result = ""
    for ch in str1:
        if ch in str2 and ch not in result:
            result += ch
    return result
str1 = 'Python3'
str2 = 'Python2.7'
print("Original strings:")
print(str1)
print(str2)
print("\nIntersection of two said String:")
print(intersection_of_two_string(str1, str2))
```

Python - Slicing Strings

- You can return a range of characters by using the slice syntax.
- Specify the start index and the end index, separated by a colon, to return a part of the string.
- Example: Get the characters from position 2 to position 5 (not included):

```
b = "Hello, World!"  
print(b[2:5])
```

- By leaving out the start index, the range will start at the first character:

```
b = "Hello, World!"  
print(b[:5])
```

- By leaving out the *end* index, the range will go to the end:

```
b = "Hello, World!"  
print(b[2:])
```

Python - Slicing Strings

Negative Indexing

- Use negative indexes to start the slice from the end of the string:

Example: Get the characters From: "o" in "World!" (position -5)

To, but not included: "d" in "World!" (position -2):

Solution:

```
b = "Hello, World!"  
print(b[-5:-2])
```

Example 2

Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return instead of the empty string.

Sample String : 'w3resource'

Expected Result : 'w3ce'

```
def string_both_ends(str):
    if len(str) < 2:
        return ""
    return str[0:2] + str[-2:]
print(string_both_ends('w3resource'))
print(string_both_ends('w3'))
print(string_both_ends('w'))
```

Example 3

Write a Python function to get a string made of 4 copies of the last two characters of a specified string (length must be at least 2).

Sample function and result :

`insert_end('Python') -> onononon`

`insert_end('Exercises') -> eseseses`

```
def insert_end(str):  
    sub_str = str[-2:]  
    return sub_str * 4
```

```
print(insert_end('Python'))
```

```
print(insert_end('Exercises'))
```

Example 4

Write a Python program to get a single string from two given strings, separated by a space and swap the first two characters of each string.

Sample String : 'abc', 'xyz'

Expected Result : 'xyc abz'

```
def chars_mix_up(a, b):  
    new_a = b[:2] + a[2:]  
    new_b = a[:2] + b[2:]  
  
    return new_a + ' ' + new_b  
print(chars_mix_up('abc', 'xyz'))
```

Example 5

Write a Python program to remove the characters which have odd index values of a given string.

```
def odd_values_string(str):  
    result = ""  
    for i in range(len(str)):  
        if i % 2 == 0:  
            result = result + str[i]  
    return result  
  
print(odd_values_string('abcdef'))  
print(odd_values_string('python'))
```

Python - Modify Strings

Python has a set of built-in methods that you can use on strings.

method	Description
upper()	The upper() method returns the string in upper case:
lower()	The lower() method returns the string in lower case:
strip()	The strip() method removes any whitespace from the beginning or the end:
replace()	The replace() method replaces a string with another string:
split()	The split() method returns a list where the text between the specified separator becomes the list items.
find(pattern,start)	find(pattern,start),Returns index position of pattern in s beginning at start. Start default is 0. Not found return -1.

Python - Modify Strings

Example 1:

```
a = "Hello, World!"  
print(a.upper())  
print(a.lower())
```

Example 2:

```
a = " Hello, World! "  
print(a.strip()) # returns "Hello, World!"
```

Example 3:

```
a = "Hello, World!"  
print(a.replace("H", "J"))
```

Example 4:

```
a = "Hello, World!"  
print(a.split(", ")) # returns ['Hello', ' World!']
```

Python - Modify Strings

Example 4:

```
a = "Hello, World!"  
print(a.split(",")) # returns ['Hello', ' World!']
```

Example 5:

```
request = 'eggs and milk and apples'  
print(request)  
x=request.split( ) # returns ['eggs', 'and', 'milk', 'and', 'apples']  
print(x)  
x=request.split('and') # returns ['eggs ', ' milk ', ' apples']  
print(x)  
x=request.split(' and ') # returns ['eggs', 'milk', 'apples']  
print(x)
```

Python - Modify Strings

Example 6:

```
s="computer information system"
```

```
print(s.find("information"))
```

```
print(s.find("information",3))
```

```
print(s.find("information",10))
```

```
9  
9  
-1
```

Example 8

Write a Python script that takes input from the user and displays that input back in upper and lower cases.

Sample Output:

What's your favourite language? english

My favourite language is ENGLISH

My favourite language is english

Solution:

```
user_input = input("What's your favourite language? ")
print("My favourite language is ", user_input.upper())
print("My favourite language is ", user_input.lower())
```

Example 9

Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '\$', except the first char itself.

Sample String : 'restart'

Expected Result : 'resta\$t'

```
def change_char(str1):  
    char = str1[0]  
    str1 = str1.replace(char, '$')  
    str1 = char + str1[1:]  
    return str1  
  
print(change_char('restart'))
```

Python - String Format

- As we learned in the Python Variables chapter, we cannot combine strings and numbers like this:

```
age = 36
```

```
txt = "My name is John, I am " + age # error
```

- But we can combine strings and numbers by using the format() method!
- The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:

Example

```
quantity = 3
```

```
itemno = 567
```

```
price = 49.95
```

```
myorder = "I want {} pieces of item {} for {} dollars."
```

```
print(myorder.format(quantity, itemno, price))
```

Python - String Format

- You can use index numbers {0} to be sure the arguments are placed in the correct placeholders:

Example:

```
quantity = 3
```

```
itemno = 567
```

```
price = 49.95
```

```
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
```

```
print(myorder.format(quantity, itemno, price))
```