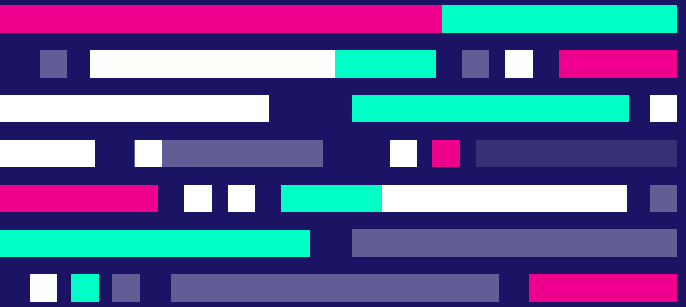
PROGRAMMING PYTHON

Lecture # 4 - Strings

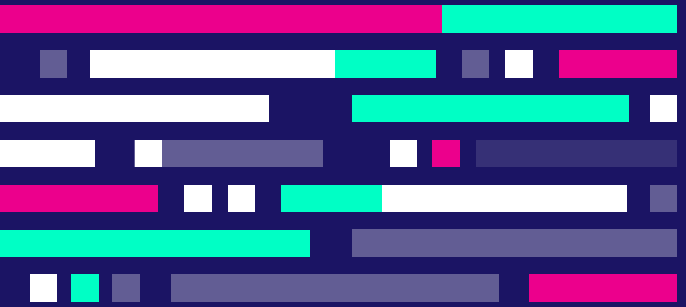
Code Camp | Alpha



Strings …

- String is a collection of alphabets, words or other characters.
- It is one of the primitive data structures and are the building blocks for data manipulation.
- Python has a built-in string class named str.
- Strings in python are surrounded by either single quotation marks, or double quotation marks.
- “hello” or ‘hell’ are the same.
- Strings are immutable, which means in order to change it’s values, we have to create new strings.
- String Manipulation
  - Indexes
  - Slicing
  - Modifying
  - Concatenation
  - Formatting
  - Escape Characters
  - String Methods

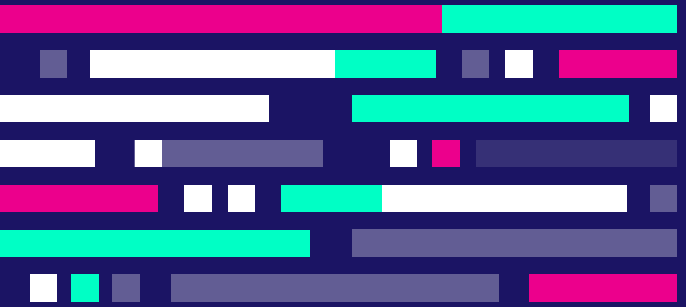
Code Camp | Alpha



Strings Creation

- You can create strings with **single quote** to incorporate **double quotes**
  - Example:
  - Sentence = 'Single quote allow you to embed "double" quotes in your string.'
- You can also create strings using **double quotes** to incorporate **single quotes**
  - Example:
  - Sentence = "Double quote allow you to embed 'single' quotes in your string."
- **Multiline strings** are created using **triple quotes**
  - Example:
  - Sentence = """Triple quotes allows to embed "double quotes" as well as 'single quotes' in your string. And can also span across multiple lines."""

Code Camp | Alpha

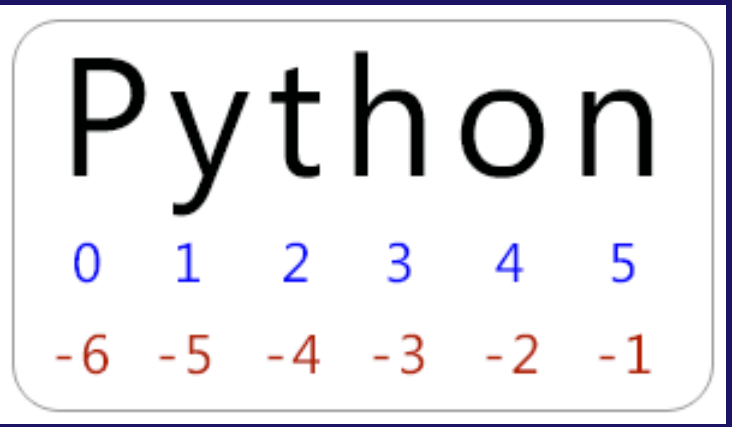


Strings Indexing

- Strings are arrays! Which means elements of the strings can be accessed.
- Strings are indexed with respect to each character in the string and the indexing begins at 0.
- Square brackets can be used to access elements of the string.
  - Example:
  - message = “Hello World”
  - print(Message[0])
  - The output of this program will be “H”
  - Explanation:
  - The length of the string “Hello World” is 11 (5+1+5) so total number of indexes will be 10, because in arrays the counting begins from 0. So the first position is 0, which means it is “H”, the second position is 1, which means “e” and so on.
- To get a desired portion of the string, is called slicing!

Code Camp | Alpha

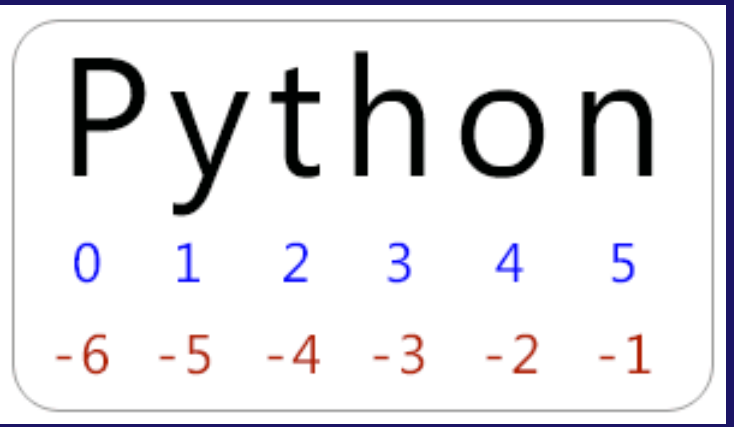
Slicing & Indexing



- You can either slice from the start of the string with index of 0 or slice from the end of the string which begins at -1. The later technique is called negative indexing.
- You can also select a range from the given string.
  - **Examples**:
  - message = "Hello World"
  - print(message[2]) # Output: l
  - print(message[-3] # Output: r
  - print(message[2:7]) # Output: llo W
  - Notice: At the 7th index it is "o", which is not printed because while selecting a range, the lower bound is included whereas the upper bound is excluded.
  - Remember, range in indexing always go from left to right, so no matter if you use negative indexing, it will just proceed in the forward direction.
  - print(message[-5:2]) # No output
  - print(message[-5:-2]) # Wor

Code Camp | Alpha

Slicing & Indexing

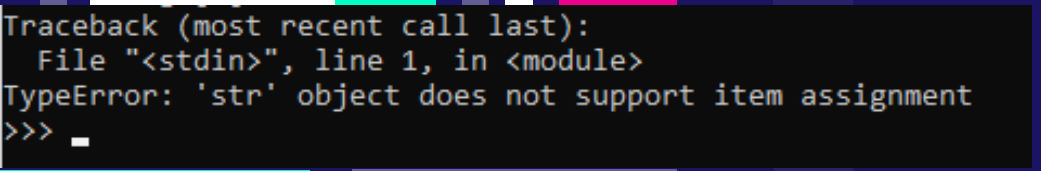


- Slicing from the start: By leaving out the start index, the range will start at the first character till the index mentioned.
  - `print(message[:5]) # Output: Hello`
- Slice to the end: By leaving out the end index, the range will go to the end, starting from the start index.
  - `print(message[6:]) # Output: World`
- Do some practice!
- *Sentence = "The quick brown fox jumps over the lazy dog"*
  - Slice "Dog" using negative indexing
  - Find the length of the string
  - Print the remaining sentence starting from "jumps over ..."
  - Slice till fox
  - Slice brown fox.

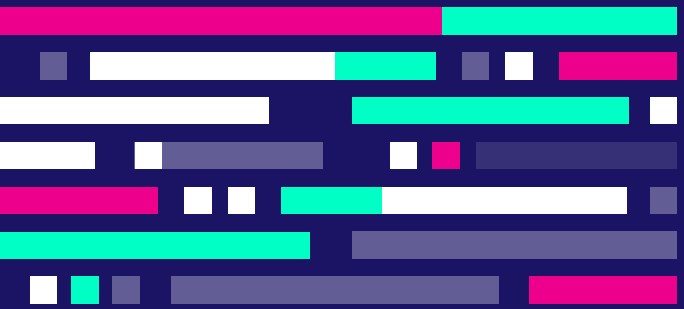
Strings Modification

- As mentioned earlier, strings are immutable, which means they cannot be modified. However, we can always create new modified stances of the string.
  - Example:
  - The following line of code trying to modify our string by changing small “e” in hello to capital “E”
  - message[1] = “E”
  - But in doing to, we receive an error

    Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
    TypeError: 'str' object does not support item assignment
  - It is because strings are not supposed to change its values.
  - However there are some methods available to perform different types of modification / formatting tasks.

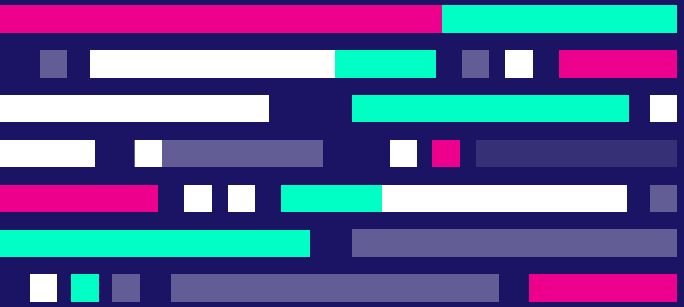


Code Camp | Alpha



Stride

- String slicing can also accept a third parameter, the stride, which refers to how many characters you want to move forward after the first character is retrieved from the string. The value of stride is set to 1 by default.
- Let's see stride in action
  - to understand it better:
  - number_string = "1020304050"
  - print(number_string[0:-1:2]) # Output: 123454
  - Its like we are telling the program to go from 0 to -1 will a jump of 2.
- Tip: Something, very cool that you can do with striding is reverse a string:
  - print(number_string[::-1]) # Output 0504030201



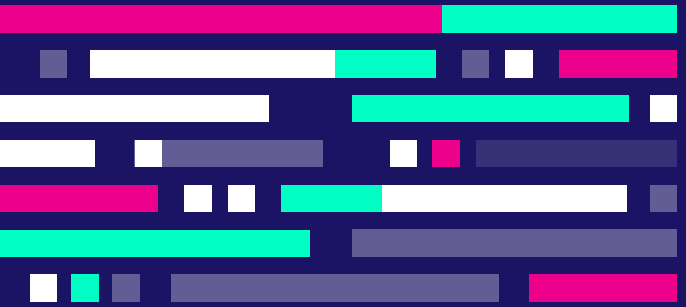
String Functions

- The upper() method returns the string in upper case:
  - a = "Hello, World!"
  - print(a.upper())
- The lower() method returns the string in lower case:
  - a = "Hello, World!"
  - print(a.lower())
- Whitespace is the space before and/or after the actual text, and very often you want to remove this space.
- The strip() method removes any whitespace from the beginning or the end:
  - a = " Hello, World! "
  - print(a.strip()) # returns "Hello, World!"
- The replace() method replaces a string with another string:
  - a = "Hello, World!"
  - print(a.replace("H", "J"))
  - Note, if you save it, it will be a new stance.
- The split() method returns a list where the text between the specified separator becomes the list items.
- The split() method splits the string into substrings if it finds instances of the separator:
  - a = "Hello, World!"
  - print(a.split(",")) # returns ['Hello', ' World!']

Code Camp | Alpha

String Concatenation

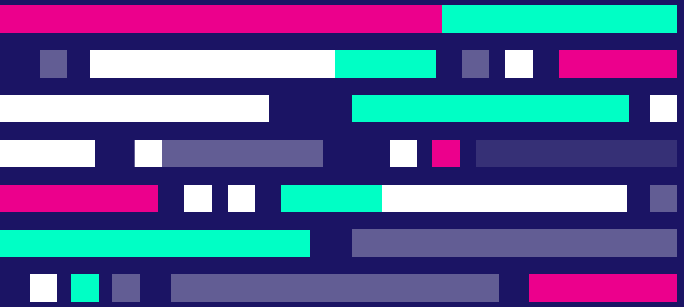
- To concatenate, or combine, two strings you can use the + operator.
- Example:
  - a = "Hello"
  - b = "World"
  - c = a + b
  - print(c)
- Concatenation of strings will not work with other data types. So, we use other techniques. Like converting something into strings, or using a comma.



Code Camp | Alpha

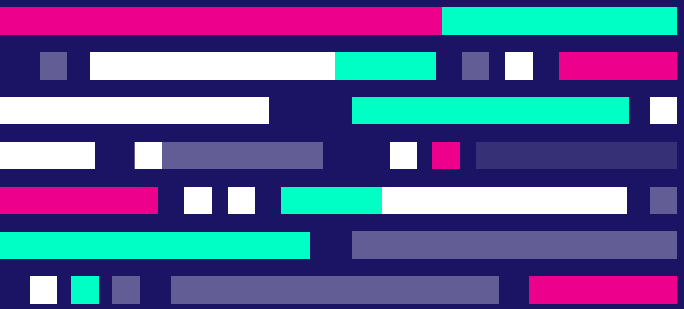
String Formatting

- To combine strings with any other data type, we need string formatting techniques.
- **Format() Method:**
  - The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:
  - **Example:**
  - bill = 100
  - output = "Your bill is ${}"
  - print(output.format(bill)) # Output: Your bill is $100
  - It can take as many arguments, but the order in which they get printed is to be maintained.
  - **Example:**
  - quantity = 3
  - itemno = 567
  - price = 49.95
  - myorder = "I want {} pieces of item {} for {} dollars."
  - print(myorder.format(quantity, itemno, price))



String Formatting

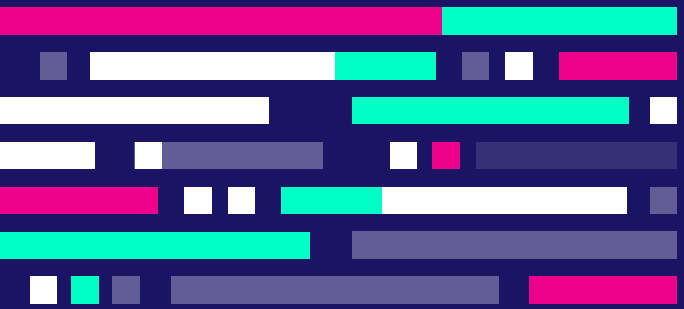
- You can use index numbers {0} to be sure the arguments are placed in the correct placeholders:
  - Example:
  - quantity = 3
  - itemno = 567
  - price = 49.95
  - myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
  - print(myorder.format(quantity, itemno, price))
  - Example:
  - "Hello, {name}! You're {age} years old.".format(name="Jane", age=25) # Output "Hello, Jane! You're 25 years old."
- format() method is not very much in use these days.
- f-strings:
  - Python f-strings provide a quick way to interpolate and format strings. They're readable, concise, and less prone to error than traditional string interpolation and formatting tools, such as the .format() method
  - An f-string is also a bit faster than those tools!
  - F-strings make the string interpolation process intuitive, quick, and concise. The syntax is similar to what you used with .format(), but it's less verbose.



String Formatting

- Example:
  - name = "Jane"
  - age = 25
  - Print(f"Hello, {name}! You're {age} years old.“) # Output: 'Hello, Jane! You're 25 years old.‘
- f-strings can also embed expressions
  - print(f"{2 * 21}“)
- They can even embed other string methods
  - print(f"Hello, {name.upper()}! You're {age} years old.“) # Output: "Hello, JANE! You're 25 years old.“
- f-strings also support the string formatting mini-language. So, you can use format specifiers in your f-strings too.
  - balance = 5425.9292
  - print(f"Balance: ${balance:.2f}") # Output: 'Balance: $5425.93'

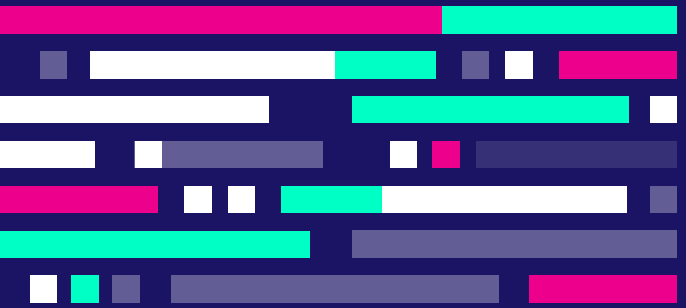
Code Camp | Alpha



Format Specifiers

- You can create a wide variety of format specifiers. Some common formats include currencies, dates, and the representation of numeric values. Consider the following examples of string formatting:
  - integer = -1234567
  - print(f"Comma as thousand separators: {integer:,}") # Output: 'Comma as thousand separators: -1,234,567'
  - sep = "_"
  - print(f"User's thousand separators: {integer:{sep}}") # Output: 'User's thousand separators: -1_234_567'
  - floating_point = 1234567.9876
  - print(f"Comma as thousand separators and two decimals: {floating_point:,.2f}")
  - Output: 'Comma as thousand separators and two decimals: 1,234,567.99'
  - date = (9, 6, 2023)
  - print(f"Date: {date[0]:02}-{date[1]:02}-{date[2]}") # Output: 'Date: 09-06-2023'
  - from datetime import datetime
  - date = datetime(2023, 9, 26)
  - print(f"Date: {date:%m/%d/%Y}") # Output: 'Date: 09/26/2023'
- These examples show how flexible the format specifiers can be. You can use them to create almost any string format.

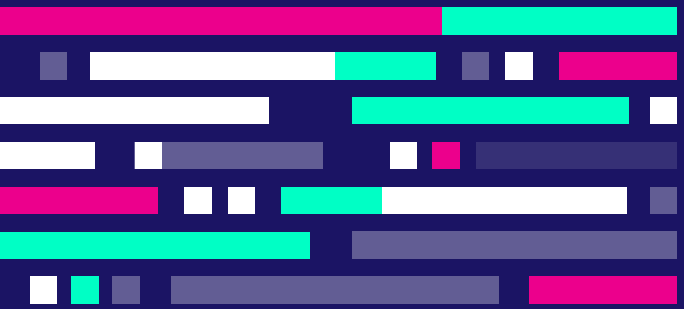
Code Camp | Alpha



More String functions

- To repeat a string, use the * operation.
  - single_word = 'hip '
  - line1 = single_word * 2 + 'hurray! '
  - print(line1 * 3) # Output: hip hip hurray!
- You can also check for membership property in a string using in and not in:
  - sub_string1 = 'ice'
  - sub_string2 = 'glue'
  - string1 = 'ice cream'
  - if sub_string1 in string1:
  - print("There is " + sub_string + " in " + string1)
  - if sub_string2 not in string1:
  - print("Phew! No " + sub_string2 + " in " + string1)
  - PRACTICE: Can you validate an email address?
- str.capitalize(): returns a copy of the string with its first character capitalized.
  - str.capitalize('capitalize first letter")
- str.islower(): returns true if all characters in the string are lowercase, false otherwise.
  - snack = 'cookie'
  - snack.islower()

Code Camp | Alpha



More String functions

- **str.find(substring):** returns the lowest index in the string where the substring is found. You can also specify the start and end index within the string where you want the substring to be searched for. Returns -1 if the substring is not found.
  - str1 = 'I got you a cookie'
  - str2 = 'cook'
  - str1.find(str2)
- **str.count(substring):** counts how many times a substring occurs in the string. You can also specify the start and the stop index for the string.
  - str1 = 'I got you a cookie, do you like cookies?'
  - str2 = 'cookie'
  - str1.count(str2)
- **str.lstrip():** removes all leading whitespace in string. This is another function that can be handy when you're working with real-life datasets.
  - str1 = " I can't hear you. Are you alright? "
  - str2 = " Yes, all is good."
  - str3 = str1.lstrip() + str2.lstrip()
  - print(str3)
- str.replace(substring, new): replaces all occurrences of the substring in string with new. You can also define a third argument max, which replaces at most max occurrences of substring in the string. Remember that that is not an inplace replacement, which means the immutable property still holds and a new string is actually formed.
  - string1 = 'hip hip hurray! hip hip hurray! hip hip hurray!'
  - string2 = string1.replace('hip', 'Hip')
  - print(string1)
  - print(string2)

Code Camp | Alpha

More String functions

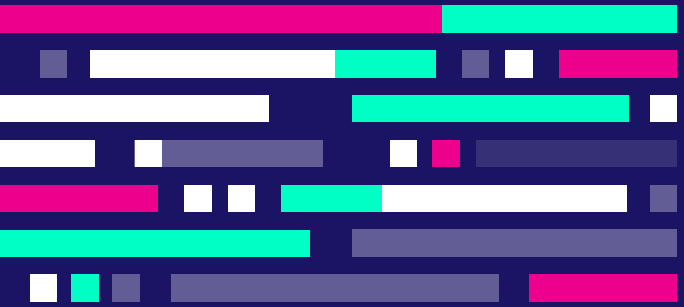
- str.split(delimiter=""): splits the string according to the delimiter (space if not provided) and returns a list of substrings.
  - dessert = 'Cake, Cookie, Icecream'
  - list_dessert = string1.split(',')
- There are more functions available, you can check in the reference provided.
- Also give Template String a go, its similar to string formatting with slight differences.

Code Camp | Alpha

Escape Characters

- To insert characters that are illegal in a string, use an escape character.
- An escape character is a backslash \ followed by the character you want to insert.
- An example of an illegal character is a double quote inside a string that is surrounded by double quotes:
  - txt = "We are the so-called "Vikings" from the north."
- To fix this problem, use the escape character \":
  - txt = "We are the so-called \"Vikings\" from the north."

Code	Result
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value



Some References

- https://www.w3schools.com/python/python_ref_string.asp
- https://www.datacamp.com/tutorial/python-string-tutorial
- https://realpython.com/python-f-strings/

Code Camp | Alpha

Practice

- First and most important of all: Check all the code snippets and see the output. Try manipulating different things for better understanding.
- Problems:
  1. Write a basic program that validates Email addresses. The program should check for the existence of “@” and “.” at desired locations.
  2. Suppose we have a user database as
     users = “user1,user2,user3,user4,user5,user6,user7”.
     Write a python program to convert it into a list of users.
  3. Write a Python program to get a string made of the first 2 and last 2 characters of a given string. If the string length is less than 2, return the empty string instead.
  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.
  5. Write a Python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing', add 'ly' instead. If the string length of the given string is less than 3, leave it unchanged.

Code Camp | Alpha

Practice

- More practice:
  1. Write a Python program to find the first appearance of the substrings 'not' and 'poor' in a given string. If 'not' follows 'poor', replace the whole 'not'...'poor' substring with 'good'. Return the resulting string.
  2. Write a function that checks if a given string is a palindrome (reads the same forwards and backward).
  3. Write a program to count the number of words in a given text.
  4. Write a program to check phone number format and write it in (+92) - xxx - xxxxxxx format.