Main String Operations

Table of Contents

Immutability

```
best ='Clarusway'
```

```
best.upper()
```

```
best.title()
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

The pre-class material is clear enough.

Students choose an option

Immutability

```
var_string = 'ClarusWay'
print(var_string.lower())
print(var_string)
var_string = 'ClarusWay'.lower()
print(var_string)
```

We can't change the string

To change string, we should reassign the new (changed) string value to a variable

```
clarusway
ClarusWay
clarusway
```


Immutability

```
var_str = 'In God we Trust'
var_str.lower()
print(var_str)
```

What is the output? Try to figure out in your mind...

Immutability

```
var_str = 'In God we Trust'
var_str.lower()
print(var_str)
```

We couldn't change the string

```
In God we Trust
```


Searching a String

Searching a String

- `string.startswith()`
  Starts searching from the **beginning** to the end.

- `string.endswith()`
  Starts searching from the **very end** to the beginning.

Searching a String(review pre-class)

- Let's take a look these pre-class examples :

```
1  text = 'www.clarusway.com'
2  print(text.endswith('.com'))
3  print(text.startswith('http:'))
4
```

What is the output? Try to figure out in your mind...

Searching a String(review pre-class)

- Let's take a look these pre-class examples :

```
1  text = 'www.clarusway.com'
2  print(text.endswith('.com'))
3  print(text.startswith('http:'))
4
```

```
1  True
2  False
3
```


Searching a String(review pre-class)

- Let's take a look these pre-class examples :

```
1  text = 'www.clarusway.com'
2  print(text.endswith('.com'))
3  print(text.startswith('http:'))
4
```

```
1  True
2  False
3
```

```
1  text = 'www.clarusway.com'
2  print(text.endswith('om'))
3  print(text.startswith('w'))
4
```

What is the output? Try to figure out in your mind...

Searching a String(review pre-class)

- Let's take a look these pre-class examples :

```
1  text = 'www.clarusway.com'
2  print(text.endswith('.com'))
3  print(text.startswith('http:'))
4
```

```
1  True
2  False
3
```

```
1  text = 'www.clarusway.com'
2  print(text.endswith('om'))
3  print(text.startswith('w'))
4
```

```
1  True
2  True
3
```


Searching a String(review pre-class)

The formula syntaxes are :

- `.startswith(prefix[, start[, end]])`
- `.endswith(suffix[, start[, end]])`

Searching a String(review pre-class)

The formula syntaxes are :

- `.startswith(prefix[, start[, end]])`
- `.endswith(suffix[, start[, end]])`

ends not included

Searching a String(review pre-class)

- Consider this pre-class example :

```
1  email = "clarusway@clarusway.com is my e-mail address"
2  print(email.startswith("@", 9))
3  print(email.endswith("-", 10, 32))
4
```

What is the output? Try to figure out in your mind...

Searching a String(review pre-class)

- Consider this example :

```
1  email = "clarusway@clarusway.com is my e-mail address"
2  print(email.startswith("@", 9))
3  print(email.endswith("-", 10, 32))
4
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
c	l	a	r	u	s	w	a	y	@	c	l	a	r	u	s	w	a	y	.	c	o	m
	i	s		m	y		e	-	m	a	i	l		a	d	d	r	e	s	s		
23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42			

Searching a String

- Consider this example :

```
1  email = "clarusway@clarusway.com is my e-mail address"
2  print(email.startswith("@", 9))
3  print(email.endswith("-", 10, 32))
4
```

```
1  True
2  True
3
```

"clarusway@clarusway.com is my e-mail address"

@ is the 9th and m is the 32nd letter starting from zero

Searching a String

- Try to figure out the output of this code :

```
text = 'www.clarusway.com'
print(text.endswith('.co'))
print(text.startswith('w.'))
```

Students, write your response!

Searching a String

- The output :

```
text = 'www.clarusway.com'
print(text.endswith('.co'))
print(text.startswith('w.'))
```

Output

```
False
False
```

Changing a String

Changing a String

- The formula syntax 👇

```
string.method(arguments)
```

Changing a String

- The summary of some common and the most important methods are as follows 👇

- `str.replace(old, new[, count])` replaces all occurrences of old with the new.

The count argument is optional, and if the optional argument count is given, only the first count occurrences are replaced. `count`: Maximum number of occurrences to replace. **-1** (the default value) means replace all occurrences.

- `str.swapcase()` converts upper case to lower case and vice versa.
- `str.capitalize()` changes the first character of the string to the upper case and the rest to the lower case.
- `str.upper()` converts all characters of the string to the upper case.
- `str.lower()` converts all characters of the string to the lower case.
- `str.title()` converts the first character of each word to upper case.

Changing a String

- Let's grasp these methods through the examples 👇

```
1  sentence = "I live and work in Virginia"
2
3  print(sentence.upper())
4
5  print(sentence.lower())
6
7  print(sentence.swapcase())
8
9  print(sentence)  # note that, source text is unchanged
10
```

What is the output? Try to figure out in your mind...

Changing a String

- Let's grasp these methods through the examples 👇

```
1  sentence = "I live and work in Virginia"
2
3  print(sentence.upper())
4
5  print(sentence.lower())
6
7  print(sentence.swapcase())
8
9  print(sentence)  # note that, source text is unchanged
10
```

```
1  I LIVE AND WORK IN VIRGINIA
2  i live and work in virginia
3  i LIVE AND WORK IN vIRGINIA
4  I live and work in Virginia
5
```


Changing a String

💡 Note that,

All these methods return `str` type.

So we can use the following syntax.

```
string.method1().method2().method3()...
```

Changing a String

- Let's grasp these methods through the examples 👇

```
1  sentence = "I live and work in Virginia"
2  title_sentence = sentence.title()
3  print(title_sentence)
4
5  changed_sentence = sentence.replace("i", "+")
6  print(changed_sentence)
7
8  print(sentence)  # note that, again source text is unchanged
9
```


Changing a String

- Let's grasp these methods through the examples 👇

```
1  sentence = "I live and work in Virginia"
2  title_sentence = sentence.title()
3  print(title_sentence)
4
5  changed_sentence = sentence.replace("i", "+")
6  print(changed_sentence)
7
8  print(sentence)  # note that, again source text is unchanged
9
```

```
1  I Live And Work In Virginia
2  I l+ve and work +n V+rg+n+a
3  I live and work in Virginia
4
```

Changing a String

- **Task** 👇
  - Let's change the **first letters** of **each words** to **uppercase** of the following text 👇

```
text = 'the better the family, the better the society'
```


Changing a String

- **The code may look like** 👇

```
text = text.title()
print(text)
```

```
The Better The Family, The Better The Society
```

Changing a String

- Let's review the pre-class examples 👇

```
sentence = "I live and work in Virginia"
swap_case = sentence.swapcase()
print(swap_case)
print(swap_case.capitalize())  # changes 'i' to uppercase and
# the rest to lowercase

```


Changing a String

- Let's review the pre-class examples 👇

```
sentence = "I live and work in Virginia"
swap_case = sentence.swapcase()
print(swap_case)
print(swap_case.capitalize())  # changes 'i' to uppercase and
# the rest to lowercase

```

```
i LIVE AND WORK IN vIRGINIA
I live and work in virginia

```

Changing a String

- **Task** 👇
  - In the `text` below, accidentally the number 0(zero) is used instead of the letter 'o' (oh). Fix them using `.replace()` method and **change** the **value of the variable** considering the new text and print the result.

```
text = 'S0d0me and G0m0re'
```


Changing a String

- **The code may look like** 👇

```
text = 'S0d0me and G0m0re'
text = text.replace('0', 'o')
print(text)
```

```
Sodome and Gomore
```

Editing a String

Editing a String

- The formula syntax 👇

```
string.method(arguments)
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Editing a String

- The summary of some common and the most important methods are as follows 👇

- `string.strip()` : removes all spaces (or specified characters) from both sides.
- `string.rstrip()` : removes spaces (or specified characters) from the right side.
- `string.lstrip()` : removes spaces (or specified characters) from the left side.

Editing a String

`string.strip(arg)`

- Let's review the pre-class examples 👇

```
1  space_string = "     listen first      "
2  print(space_string.strip())  # removes all spaces from both sides
3
```

Editing a String

`string.strip(arg)`

- Let's review the pre-class examples 👇

```
space_string = "     listen first      "
print(space_string.strip())  # removes all spaces from both sides

```

```
listen first

```


Editing a String

`string.strip(arg)`

- Let's review the pre-class examples 👇

```
space_string = "     listen first      "
print(space_string.strip())  # removes all spaces from both sides

```

```
listen first

```

```
source_string = "interoperability"
print(source_string.strip("yi"))
# removes trailing "y" or "i" or "yi" or "iy" from both sides

```

Editing a String

```
string.strip(arg)
```

- Let's review the pre-class examples 👇

```
space_string = "     listen first      "
print(space_string.strip())  # removes all spaces from both sides

```

```
listen first

```

```
source_string = "interoperability"
print(source_string.strip("yi"))
# removes trailing "y" or "i" or "yi" or "iy" from both sides

```

```
nteroperabilit

```


Editing a String

```
string.lstrip(arg)
string.rstrip(arg)
```

- Let's review the pre-class examples 👇

```
source_string = "interoperability"
print(source_string.lstrip("in"))
# removes "i" or "n" or "in" or "ni" from the left side

```

What is the output? Try to figure out in your mind...

Editing a String

- Let's review the pre-class examples 👇

```
string.lstrip(arg)
string.rstrip(arg)
```

```
source_string = "interoperability"
print(source_string.lstrip("in"))
# removes "i" or "n" or "in" or "ni" from the left side

```

```
teroperability

```


Editing a String

- Let's review the pre-class examples 👇

```
string.lstrip(arg)
string.rstrip(arg)
```

```
source_string = "interoperability"
print(source_string.lstrip("in"))
# removes "i" or "n" or "in" or "ni" from the left side

```

```
teroperability

```

```
space_string = "     listen first      "
print(space_string.rstrip())  # removes spaces from the right side

```

What is the output? Try to figure out in your mind...

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

Editing a String

- Let's review the pre-class examples 👇

```
string.lstrip(arg)
string.rstrip(arg)
```

```
source_string = "interoperability"
print(source_string.lstrip("in"))
# removes "i" or "n" or "in" or "ni" from the left side

```

```
teroperability

```

```
space_string = "     listen first      "
print(space_string.rstrip())  # removes spaces from the right side

```

```
    listen first

```


Editing a String

- Let's review the pre-class examples 👇

```
string.lstrip(arg)
string.rstrip(arg)
```

```
source_string = "interoperability"
print(source_string.rstrip("yt"))
# removes "y" or "t" or "yt" or "ty" from the right side

```

What is the output? Try to figure out in your mind...

Editing a String

```
string.lstrip(arg)
string.rstrip(arg)
```

- Let's review the pre-class examples 👇

```
1  source_string = "interoperability"
2  print(source_string.rstrip("yt"))
3  # removes "y" or "t" or "yt" or "ty" from the right side
4
```

```
1  interoperabili
2
```


Editing a String

```
string.lstrip(arg)
string.rstrip(arg)
```

- Let's review the pre-class examples 👇

```
1  source_string = "interoperability"
2  print(source_string.rstrip("yt"))
3  # removes "y" or "t" or "yt" or "ty" from the right side
4
```

```
1  interoperabili
2
```

```
1  source_string = "interoperability"
2  print(source_string.rstrip("ty"))
3
```

What is the output? Try to figure out in your mind...

Editing a String

```
string.lstrip(arg)
string.rstrip(arg)
```

- Let's review the pre-class examples 👇

```
1  source_string = "interoperability"
2  print(source_string.rstrip("yt"))
3  # removes "y" or "t" or "yt" or "ty" from the right side
4
```

```
1  interoperabili
2
```

Either ty or yt works

```
1  source_string = "interoperability"
2  print(source_string.rstrip("ty"))
3
```

```
1  interoperabili
2
```


Editing a String

- **Task** 👇
  - In the `text` below, accidentally there are additional letters. Remove the additional letters and make the all words uppercase.
  - Except for the `print()` line, your code should consist of a single line.

```
text = 'tyou can learn almost everything in pre-classz'
```

desired output

```
YOU CAN LEARN ALMOST EVERYTHING IN PRE-CLASS
```

Editing a String

- **The code may look like 👇**

```
text = text.rstrip('z').lstrip('t').upper()
print(text)
```

```
YOU CAN LEARN ALMOST EVERYTHING IN PRE-CLASS
```


Editing a String

- **Task 👇**
  - In the `text` below, accidentally the number the word "**we**" is wrong written. Remove the additional "**e**" letter and print the correct text.
  - You can also use *indexing/slicing/methods* of string.

```
text = 'In God wee Trust'
```

Editing a String

- **The code options might be like : 👇**

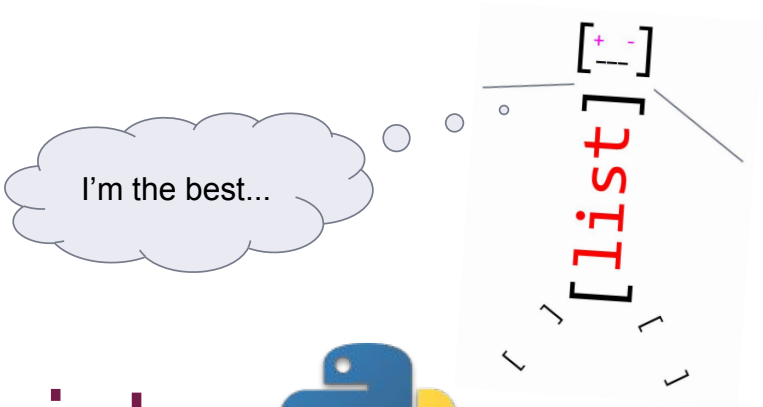
```
1  text = 'In God wee Trust'
2
3  print(text.replace("ee", "e"))
4
5  text1 = text[:9]
6  text2 = text[10:]
7  print(text1 + text2)
8
9
10
11
```

Output

```
In God we Trust
In God we Trust
```


Collection Types

- List
- Tuple
- Dictionary
- Set



Lists

Table of Contents

```
fruit = ['Apple', 'Orange', 'Banana']
best = ['Clarusway']
list()
```

Creating a List

CLARUSWAY©
WAY TO REINVENT YOURSELF

Creating a `list`

- We have two basic ways to create a list.

- `[]`
- `list()`

Creating a `list`

- How we do ?

- `[]`
- `list()`

```
list_1 = ['h', 'a', 'p', 'p', 'y']
word = 'happy'
list_2 = list(word)
print(list_1)
print(list_2)
```

What is the output? Try to figure out in your mind...

Creating a `list`

- The output :

- `[]`
- `list()`

```
list_1 = ['h', 'a', 'p', 'p', 'y']
word = 'happy'
list_2 = list(word)
print(list_1)
print(list_2)
```

```
['h', 'a', 'p', 'p', 'y']
['h', 'a', 'p', 'p', 'y']
```


Creating a `list` (review of pre-class)

- Here is another example of creating a `list` :

```
1  country = ['USA', 'Brasil', 'UK', 'Germany', 'Turkey', 'New Zealand']
2
3  print(country)
4
```

Creating a `list` (review of pre-class)

Here is another example of creating a `list` :

```
country = ['USA', 'Brasil', 'UK', 'Germany', 'Turkey', 'New Zealand']

print(country)

```

```
['USA', 'Brasil', 'UK', 'Germany', 'Turkey', 'New Zealand']

```


Creating a `list` (review of pre-class)

Here is another example of creating a `list` :

```
country = ['USA', 'Brasil', 'UK', 'Germany', 'Turkey', 'New Zealand']

print(country)

```

```
['USA', 'Brasil', 'UK', 'Germany', 'Turkey', 'New Zealand']

```

Tips:

- All the country names are printed in the same order as they were stored in the list because lists are **ordered**.

Creating a `list` (review of pre-class)

- Here is another example of creating a `list` using `list()`.

```
1  string_1 = 'I quit smoking'
2
3  new_list_1 = list(string_1)  # we created multi element list
4  print(new_list_1)
5
6  new_list_2 = [string_1]  # this is a single element list
7  print(new_list_2)
8
```

What is the output? Try to figure out in your mind...

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

Creating a `list` (review of pre-class)

- Here is the output ?

```
1  string_1 = 'I quit smoking'
2
3  new_list_1 = list(string_1)  # we created multi element list
4  print(new_list_1)
5
6  new_list_2 = [string_1]  # this is a single element list
7  print(new_list_2)
8
```

```
1  ['I', ' ', 'q', 'u', 'i', 't', ' ', 's', 'm', 'o', 'k', 'i', 'n', 'g']
2  ['I quit smoking']
3
```

Creating a `list`

Tips:

- Note that, using `list()` function, all characters of string_1 including spaces was moved into a new_list_1.
- If you noticed, **lists** can contain **more than one** of the **same** value.

```
1  string_1 = 'I quit smoking'
2
3  new_list_1 = list(string_1)  # we created multi element list
4  print(new_list_1)
5
6  new_list_2 = [string_1]  #
7  print(new_list_2)
8
```

⚠ It has only one element.

⚠ It has 14 elements.

```
1  ['I', ' ', 'q', 'u', 'i', 't', ' ', 's', 'm', 'o', 'k', 'i', 'n', 'g']
2  ['I quit smoking']
3
```


Creating a `list`

- The components of a `list` are not limited to a single data type.
- Here is an example :

```
mixed_list = [11, 'Joseph', False, 3.14, None, [1, 2, 3]]
```

⚠ int ⚠ str ⚠ bool ⚠ float ⚠ NoneType ⚠ list

Creating a `list`

- Task:

```
my_list = ['Joseph', 'Clarusway', 2020]
new_list1 = list(my_list)
new_list2 = [my_list]

print(new_list1)
print(len(new_list1))
print(new_list2)
print(len(new_list2))
```

What is the output? Try to figure out in your mind...

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

Creating a `list`

- The output:.

```
['Joseph', 'Clarusway', 2020]
3
[['Joseph', 'Clarusway', 2020]]
1
```

Creating a `list`

- Task:
  - Create a list from **string** value of `"2020's hard"`.
  - Use both `list()` function and square brackets `[]`.

Creating a `list`

- The code :

```
my_list1 = ["2020's hard"]
my_list2 = list("2020's hard")

print(my_list1)
print(my_list2)
```

Each item/char including spaces came into the list

```
["2020's hard"]
['2', '0', '2', '0', "'", 's', ' ', 'h', 'a', 'r', 'd']
```