



The Strength of the String



Table of Contents



- ▶ Indexing&Slicing Strings
- ▶ String Formatting with Arithmetic Syntax
- ▶ String Formatting with % Operator
- ▶ String Formatting with `string.format()` Method
- ▶ String Formatting with '**f-string**'

How was the pre-class content? Did you get it?



Students, drag the icon!





```
best = 'clarusway'
```

```
best[2]
```

Indexing & Slicing Strings

```
best[2:]
```



Indexing&Slicing Strings

- ▶ Let's elaborate on this example :

```
1 fruit = 'Orange'
2
3 print('Word          : ' , fruit)
4 print('First letter   : ' , fruit[0])
5 print('Second letter  : ' , fruit[1])
6 print("3rd to 5th letters : " , fruit[2:5])
7 print("Letter all after 3rd : " , fruit[2:])
8
```



Indexing&Slicing Strings

- ▶ Let's elaborate on this example :

```
1 fruit = 'Orange'
2
3 print('Word           : ' , fruit)
4 print('First letter   : ' , fruit[0])
5 print('Second letter  : ' , fruit[1])
6 print("3rd to 5th letters : " , fruit[2:5])
7 print("Letter all after 3rd : " , fruit[2:])
8
```

```
1 Word           : Orange
2 First letter   : 0
3 Second letter  : r
4 3rd to 5th letters : ang
5 Letter all after 3rd : ange
6
```



Indexing&Slicing Strings

- Let's elaborate on this example :

```
1 fruit = 'Orange'
2
3 print('Word          : ', fruit)
4 print('First letter   : ', fruit[0])
5 print('Second letter  : ', fruit[1])
6 print('3rd to 5th letters : ', fruit[2:5])
7 print('Letter all after 3rd : ', fruit[2:])
8
```

```
1 Word          : Orange
2 First letter   : O
3 Second letter  : r
4 3rd to 5th letters : ang
5 Letter all after 3rd : ange
6
```

'O r a n g e'

| | | | |

0 1 2 3 4 5

[start:stop:step]



Indexing&Slicing Strings

Here is an example of *Pre-Class* content:

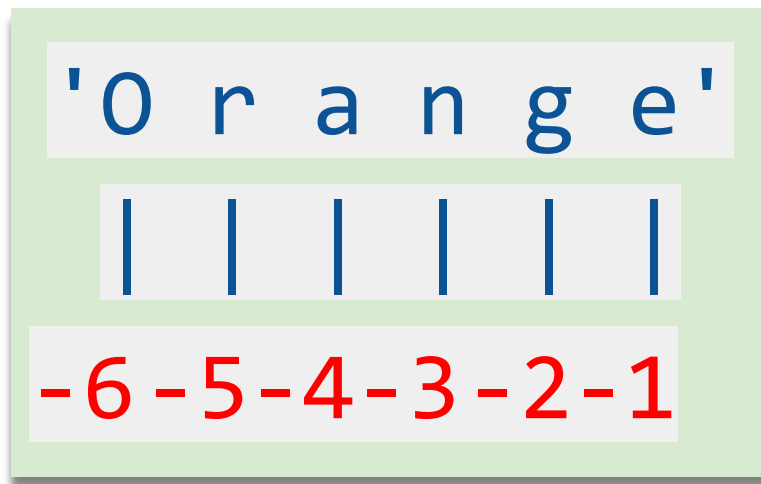
```
1 city = 'Phoenix'
2
3 print(city[1:]) # starts from index 1 to the end
4 print(city[:6]) # starts from zero to 5th index
5 print(city[::2]) # starts from zero to end by 2 step
6 print(city[1::2]) # starts from index 1 to the end by 2 step
7 print(city[-3:]) # starts from index -3 to the end
8 print(city[::-1]) # negative step starts from the end to zero
9
```

```
1 hoenix
2 Phoeni
3 Ponx
4 hei
5 nix
6 xineohP
7
```



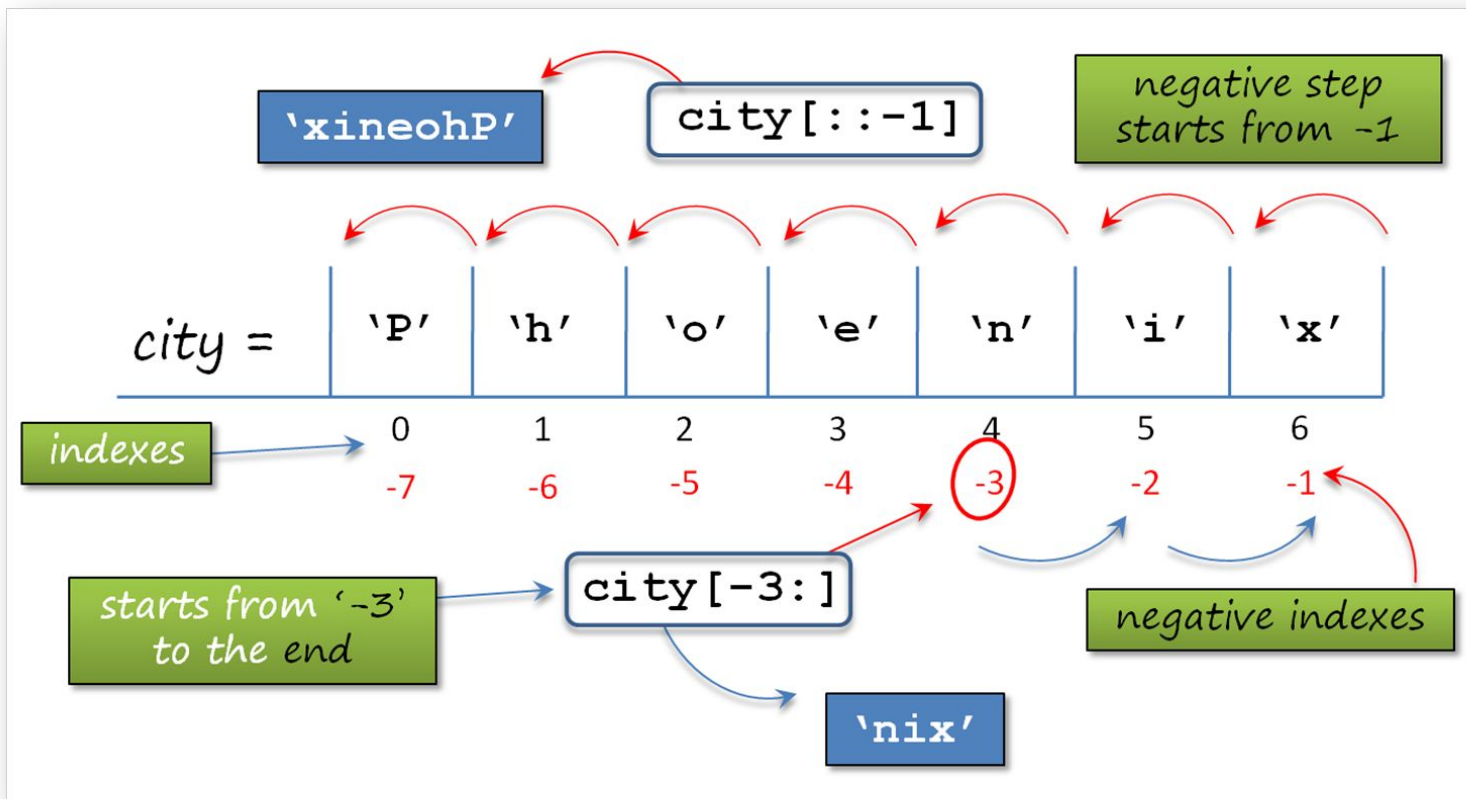

Negative Indexing Strings

- ▶ Negative indexing works as the same :





Indexing&Slicing Strings





Indexing&Slicing Strings

Here is another example :

```
animal = "hippopotamus"

print(animal[1:])
print(animal[:6])
print(animal[::2])
print(animal[1:7:2])
print(animal[-3:])
print(animal[::-1])
```

What is the output? Try to
guess in your mind...



```
animal = "hippopotamus"
```

```
print(animal[1:])  
print(animal[:6])  
print(animal[::2])  
print(animal[1:7:2])  
print(animal[-3:])  
print(animal[::-1])
```

Output

```
ippopotamus  
hippop  
hpooau  
ipp  
mus  
sumatopoppih
```



Indexing&Slicing Strings

- ▶ `len()` function measure the length of any iterable :

```
1 vegetable = 'Tomato'
2
3 print('length of the word', vegetable, 'is :', len(vegetable))
4
```

What is the output? Try to guess in your mind...



Indexing&Slicing Strings

- ▶ The output :

```
1 vegetable = 'Tomato'
2
3 print('length of the word', vegetable, 'is :', len(vegetable))
4
```

```
1 length of the word Tomato is : 6
2
```

'T o m a t o'

| | | | |

✓+✓+✓+✓+✓+✓

= Totally 6 chars



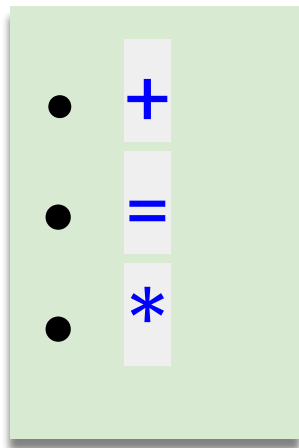
String Formatting



String Formatting with Arithmetic Syntax

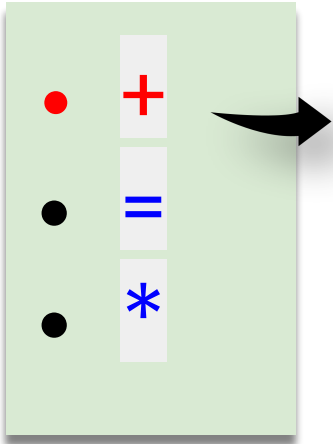
String Formatting with Arithmetic Syntax

- ▶ Here are basic operators :



String Formatting with Arithmetic Syntax

- ▶ We can use arithmetic operator syntaxes in string formatting operations
- ▶ Here are basic operators :



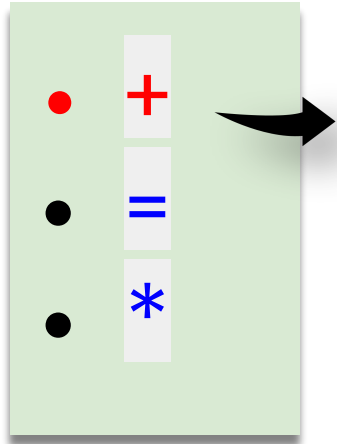
```
str_one = 'upper'  
str_two = 'case'  
str_comb = str_one + str_two  
print('upper' + 'case')  
print(str_one + str_two)  
print(str_comb)
```

What is the output? Try to guess in your mind...



String Formatting with Arithmetic Syntax

- ▶ We can use arithmetic operator syntaxes in string formatting operations
- ▶ Here are basic operators :



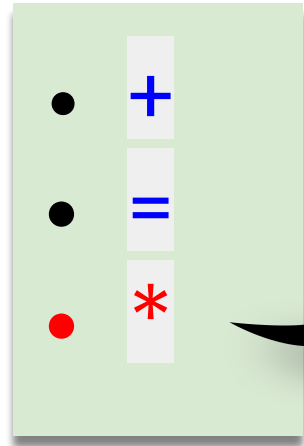
```
str_one = 'upper'  
str_two = 'case'  
str_comb = str_one + str_two  
print('upper' + 'case')  
print(str_one + str_two)  
print(str_comb)
```

```
uppercase  
uppercase  
uppercase
```



String Formatting with Arithmetic Syntax

- ▶ Another example :



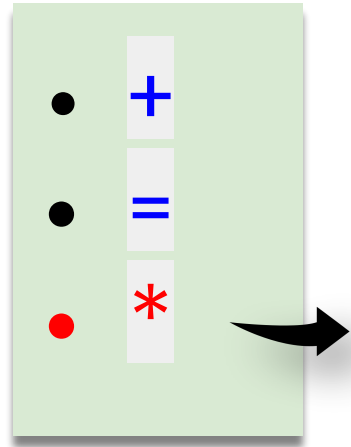
```
str_one = 'upper'  
str_two = 3 * 'upper'  
str_comb = str_one * 3  
print(str_two)  
print(str_comb)  
print(* str_one)
```

What is the output? Try to guess in your mind...



String Formatting with Arithmetic Syntax

- ▶ Another example :

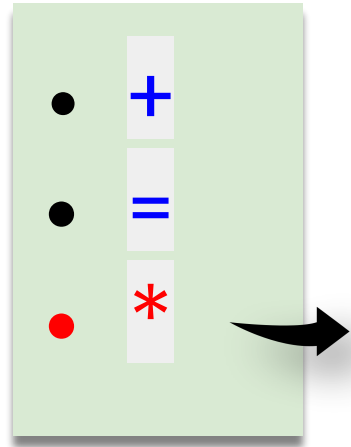


```
str_one = 'upper'  
str_two = 3 * 'upper'  
str_comb = str_one * 3  
print(str_two)  
print(str_comb)  
print(* str_one)
```

```
upperupperupper  
upperupperupper  
u p p e r
```

String Formatting with Arithmetic Syntax

- ▶ Another example :



```
str_one = 'upper'  
str_two = 3 * 'upper'  
str_comb = str_one * 3  
print(str_two)  
print(str_comb)  
print(*str_one)
```

Separates the string into its elements

```
upperupperupper  
upperupperupper  
u p p e r
```

String Formatting with Arithmetic Syntax

- ▶ Separate these strings into its characters using  `*` :

```
string_1 = 'I am angry...'
```

```
string_2 = '1453'
```

```
'joseph@clarusway.com' # Do not use variable
```

String Formatting with Arithmetic Syntax

- The output :

```
string_1 = 'I am angry...'
print(* string_1)
string_2 = '1453'
print(* string_2)
'joseph@clarusway.com' # Do not use variable
print(* 'joseph@clarusway.com')
```

```
I   a m   a n g r y . . .
1 4 5 3
j o s e p h @ c l a r u s w a y . c o m
```

String Formatting with Arithmetic Syntax

- The output :

```
string_1 = 'I am angry...'
```

**How many *space*
chars here?**

```
len(string_1) = 1453
```

```
'joseph@clarusway.com' # Do not use variable
```

```
I   a m   a n g r y . . .
```

```
1 4 5 3
```

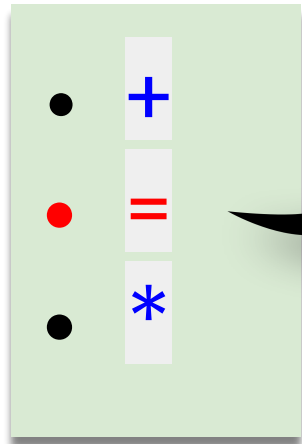
```
j o s e p h @ c l a r u s w a y . c o m
```





String Formatting with Arithmetic Syntax

- ▶ Another example :



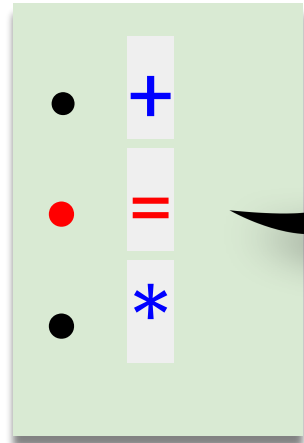
```
str_one = 'upper'  
str_one += 'case'  
print(str_one)  
str_one += 'letter'  
print(str_one)  
str_one += 'end'  
print(str_one)
```

What is the output? Try to guess in your mind...



String Formatting with Arithmetic Syntax

- ▶ Another example :



```
str_one = 'upper'  
str_one += 'case'  
print(str_one)  
str_one += 'letter'  
print(str_one)  
str_one += 'end'  
print(str_one)
```

```
uppercase  
uppercaseletter  
uppercaseletterend
```

```
str1 = str1 + str
```

```
str1 += str
```

```
str1 = str1 * 2
```

```
str1 *= 2
```



String Formatting



String Formatting with
`string.format()` Method

String Formatting with `string.format()` Method

- ▶ The formula syntax 

```
'string {} string {} string'.format(data1, data2)
```

String Formatting with `string.format()` Method

- Take a look at the example 

```
1 fruit = 'Orange'
2 vegetable = 'Tomato'
3 amount = 4
4 print('The amount of {} we bought is {} pounds'.format(fruit, amount))
5
```

What is the output? Try to guess in your mind...

String Formatting with `string.format()` Method

- Take a look at the example 

```
1 fruit = 'Orange'  
2 vegetable = 'Tomato'  
3 amount = 4  
4 print('The amount of {} we bought is {} pounds'.format(fruit, amount))  
5
```

```
1 The amount of Orange we bought is 4 pounds  
2
```

String Formatting with `string.format()` Method

- ▶ Consider this example. 

```
1 print('{state} is the most {adjective} state of the {country}'.format(state='California',  
2                               country='USA', adjective='crowded'))
```

String Formatting with `string.format()` Method

- ▶ Using keywords in  `{}` makes string more readable.



```
1 print('{state} is the most {adjective} state of the {country}'.format(state='California',  
2                               country='USA', adjective='crowded'))
```

```
1 California is the most crowded state of the USA  
2
```


String Formatting with `string.format()` Method

💡 Tips:

- If you have noticed, we do not have to write the keywords in `.format()` method in order.

► Mix Use of String Arguments

```
1 print('{0} is the most {adjective} state of the {country}'.format('California',  
2     = 'USA', adjective='crowded'))
```

keyword

positional

String Formatting with `string.format()` Method



💡 Tips:

- If you have noticed, we do not have to write the keywords in `.format()` method in order.

- ▶ You can combine both the positional and the keyword arguments in the same `.format()` method.

```
1 print('{0} is the most {adjective} state of the {country}'.format('California', country  
2      ='USA', adjective='crowded'))
```

```
1 California is the most crowded state of the USA  
2
```

String Formatting with `string.format()` Method

- Usage of `string.format` method.

```
1 print("{}-{}-{}".format("12", "Feb", "Feb"))  
2 print("{no}-{month}-{month}".format(no="12", month="Feb"))  
3
```

```
1 print("{6} {5} {0} {1} {3} {4} {2}".format("a new", "job", "months", "in", 6, "have started", "I  
will"))
```

String Formatting with `string.format()` Method



- Usage of `string.format` method.

```
1 print("{}-{}-{}".format("12", "Feb", "Feb"))  
2 print("{no}-{month}-{month}".format(no="12", month="Feb"))  
3
```

Output

```
12-Feb-Feb  
12-Feb-Feb
```

```
1 print("{6} {5} {0} {1} {3} {4} {2}".format("a new", "job", "months", "in", 6, "have started", "I  
will"))
```

Output

```
I will have started a new job in 6 months
```

String Formatting with `string.format()` Method

► Task :

- To print the statement of “**generosity wins in all circumstances**”, arrange the following code.

```
phrase = '{2} {} {} {}'.format('circumstances', 'in all', 'generosity', 'wins')  
print(phrase)
```

String Formatting with `string.format()` Method



- ▶ The code should be like that :

```
phrase = '{2} {3} {1} {0}'.format('circumstances', 'in all', 'generosity', 'wins')  
print(phrase)
```



String Formatting



String Formatting with `f-string`



String Formatting with **f-string**

- ▶ The formula syntax 

```
f'string {variable1} string {variable2} string'
```




String Formatting with **f-string**

- Take a look at the example 

```
1 fruit = 'Orange'
2 vegetable = 'Tomato'
3 amount = 6
4 output = f"The amount of {fruit} and {vegetable} we bought are totally {amount} pounds"
5
6 print(output)
7
```

What is the output? Try to guess in your mind...



String Formatting with **f-string**


- Take a look at the example 

```
1 fruit = 'Orange'
2 vegetable = 'Tomato'
3 amount = 6
4 output = f"The amount of {fruit} and {vegetable} we bought are totally {amount} pounds"
5
6 print(output)
7
```

```
1 The amount of Orange and Tomato we bought are totally 6 pounds
2
```



String Formatting with **f-string**

- ▶ You can use all valid expressions, variables, and even methods in curly braces. 

```
1 sample = f"{2 ** 3}"  
2  
3 print(sample)  
4  
5  
6
```

What is the output? Try to guess in your mind...





String Formatting with **f-string**

- ▶ You can use all valid expressions, variables, and even methods in curly braces. 

```
1 sample = f"{2 ** 3}"  
2  
3 print(sample)  
4  
5  
6
```

Output

```
8
```



String Formatting with **f-string**

► Task :

- Type a Python code to get the output of “**My name is Mariam**”, using **.capitalize()** and **f-string** methods with the **name** variable below.

```
name = "MARIAM"
```

You're familiar with **.capitalize()** method from **pre-class** materials



String Formatting with **f-string**

- ▶ The code should be like :

```
1 my_name = 'MARIAM'
2 output = f"My name is {my_name.capitalize()}"
3
4 print(output)
5
6
7
```



String Formatting with **f-string**

- ▶ There is also a multiline **f-string** formatting style. 

```
1 name = "Joseph"
2 job = "teachers"
3 domain = "Data Science"
4 message = (
5     f"Hi {name}. "
6     f"You are one of the {job} "
7     f"in the {domain} section."
8 )
9 print(message)
10
```



String Formatting with **f-string**

- There is also a multiline **f-string** formatting style. 

```
1 name = "Joseph"
2 job = "teachers"
3 domain = "Data Science"
4 message = (
5     f"Hi {name}. "
6     f"You are one of the {job} "
7     f"in the {domain} section."
8 )
9 print(message)
10
```

 Pay attention
to parentheses

```
1 Hi Joseph. You are one of the teachers in the Data Science section.
2
```




String Formatting with **f-string**

- ▶ You can use backslash  \ between lines. 

```
1 name = "Joseph"
2 job = "teachers"
3 domain = "Data Science"
4 message = f"Hi {name}. " \
5           f"You are one of the {job} " \
6           f"in the {domain} section."
7
8 print(message)
9
```



String Formatting with **f-string**

- The output :

```
1 name = "Joseph"
2 job = "teachers"
3 domain = "Data Science"
4 message = f"Hi {name}. " \
5           f"You are one of the {job} " \
6           f"in the {domain} section."
7
8 print(message)
9
```

```
1 Hi Joseph. You are one of the teachers in the Data Science section.
2
```



String Formatting with f-string

► Task :

- Type a Python code to get the output of “Susan is a young lady and she is a student at the CLRWY IT university.”, using f-string in *multiline* with the variables below.

```
name = "Susan"  
age = "young"  
gender = "lady"  
school = "CLRWY IT university"
```



String Formatting with **f-string**

- ▶ The code should be like :

```
1 name = "Susan"
2 age = "young"
3 gender = 'lady'
4 school = "CLRWY IT university"
5
6
7 output = (
8     f"{name} is a {age} "
9     f"{gender} and she is a student "
10    f"at the {school}."
11 )
12
13 print(output)
14
```



Indexing&Slicing Strings

► Task

- First, Login to your LMS,
- Then, click [here](#) to complete and submit the task.



String Formatting with `string.format()` Method

► Task

- First, Login to your LMS,
- Then, click [here](#) to complete and submit the task.



String Formatting with `f-string()` Method



► Task

- First, Login to your LMS,
- Then, click [here](#) to complete and submit the task.

