The Strength of the String

CLARUSWAY©
WAY TO REINVENT YOURSELF

Table of Contents

1

Indexing String

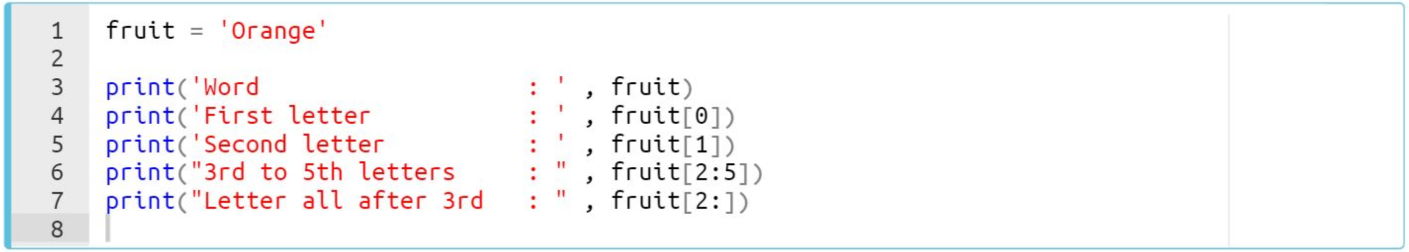
```
best ='Clarusway'
```

```
best[2]
```

```
best[2:]
```

Indexing String

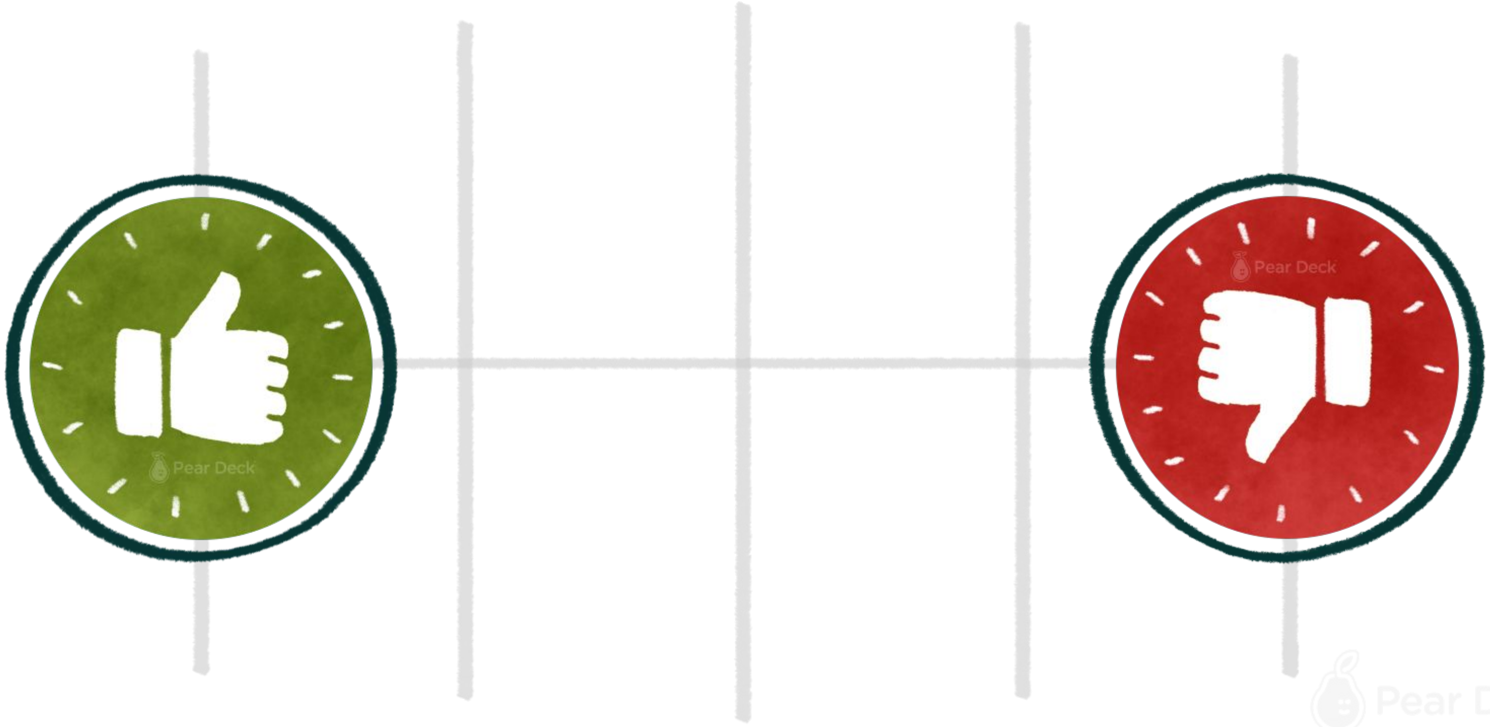
- You can access all elements of a string type data very easily. Accordance with the sequence of string letters, you can specify them from left to right in brackets. Here's an eg. :



```
fruit = 'Orange'

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

CLARUSWAY©
WAY TO REINVENT YOURSELF

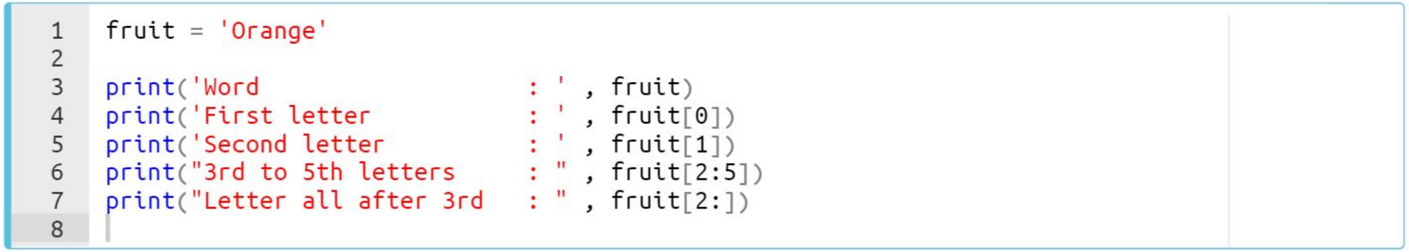
How was the pre-class content?



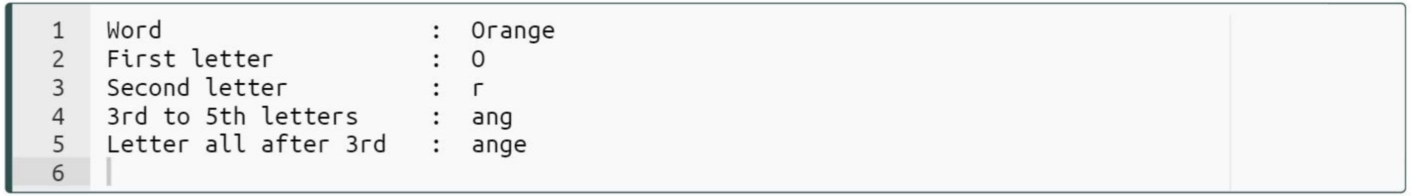
Students, drag the icon!

Indexing String

- You can access all elements of a string type data very easily. Accordance with the sequence of string letters, you can specify them from left to right in brackets. Here’s an eg. :



```python
fruit = 'Orange'

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

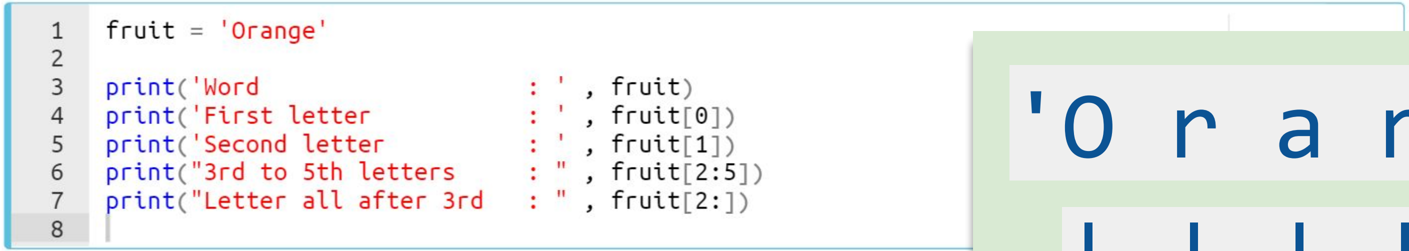


```
Word                    :  Orange
First letter            :  O
Second letter           :  r
3rd to 5th letters      :  ang
Letter all after 3rd    :  ange
```

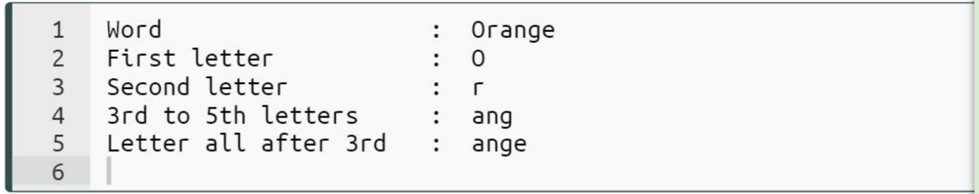
CLARUSWAY©
WAY TO REINVENT YOURSELF

Indexing String

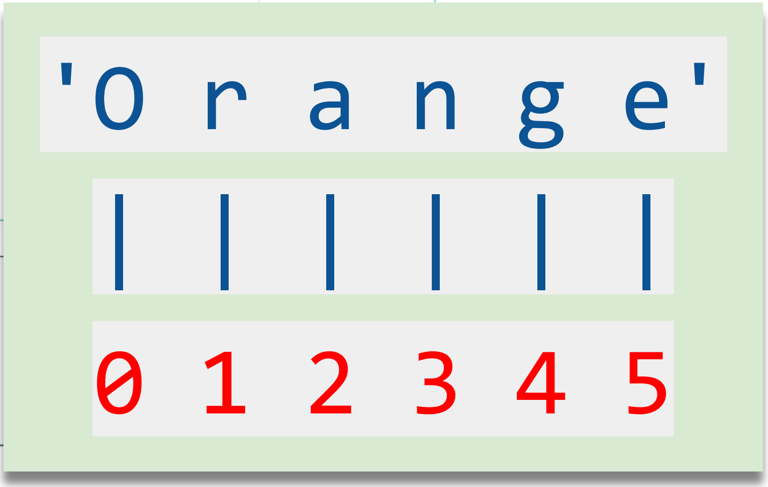
- You can access all elements of a string type data very easily. Accordance with the sequence of string letters, you can specify them from left to right in brackets. Here's an eg. :



```python
fruit = 'Orange'

print('Word                : ' , fruit)
print('First letter        : ' , fruit[0])
print('Second letter       : ' , fruit[1])
print("3rd to 5th letters  : " , fruit[2:5])
print("Letter all after 3rd  : " , fruit[2:])

```



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

```



CLARUSWAY©
WAY TO REINVENT YOURSELF

Indexing String

- **Task**
  - First, Login to your LMS,
  - Then, click **here** to complete and submit the task.

Remove a *specific* char at a *specific/given* index from a string.

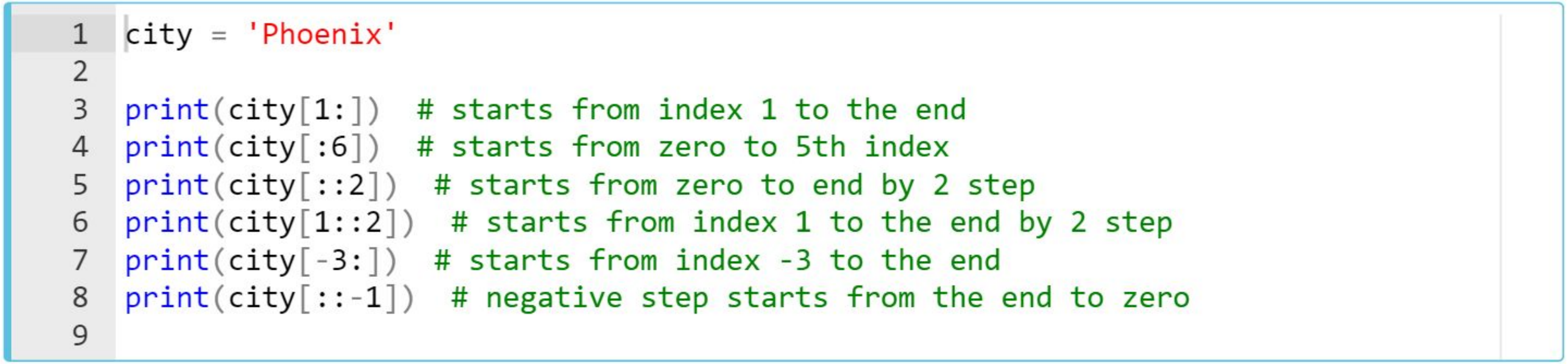
Given a string (cla**r**usway) and an index number int n (n=3), return a new string where the character at **index n** has been removed.

For example: if n=3 then, result: **clausway**

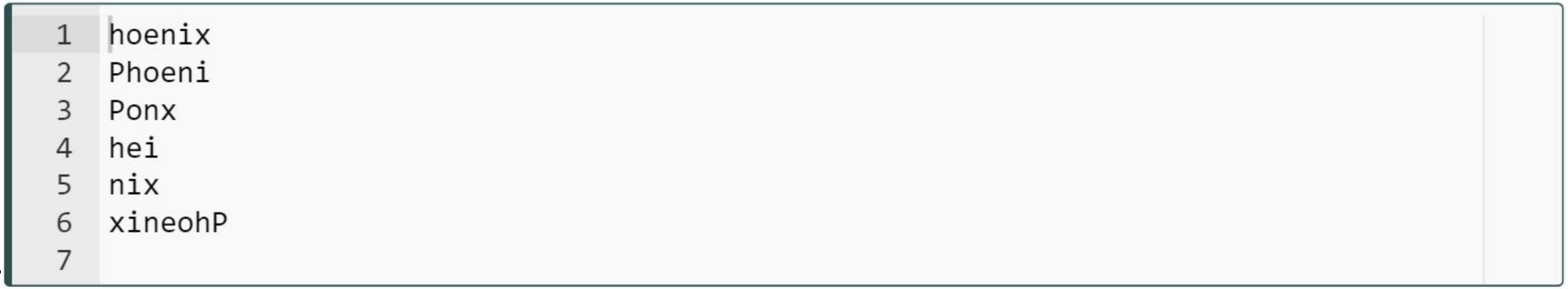
```
word = 'clarusway'; n = 3;

........

.......
```

Indexing String

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

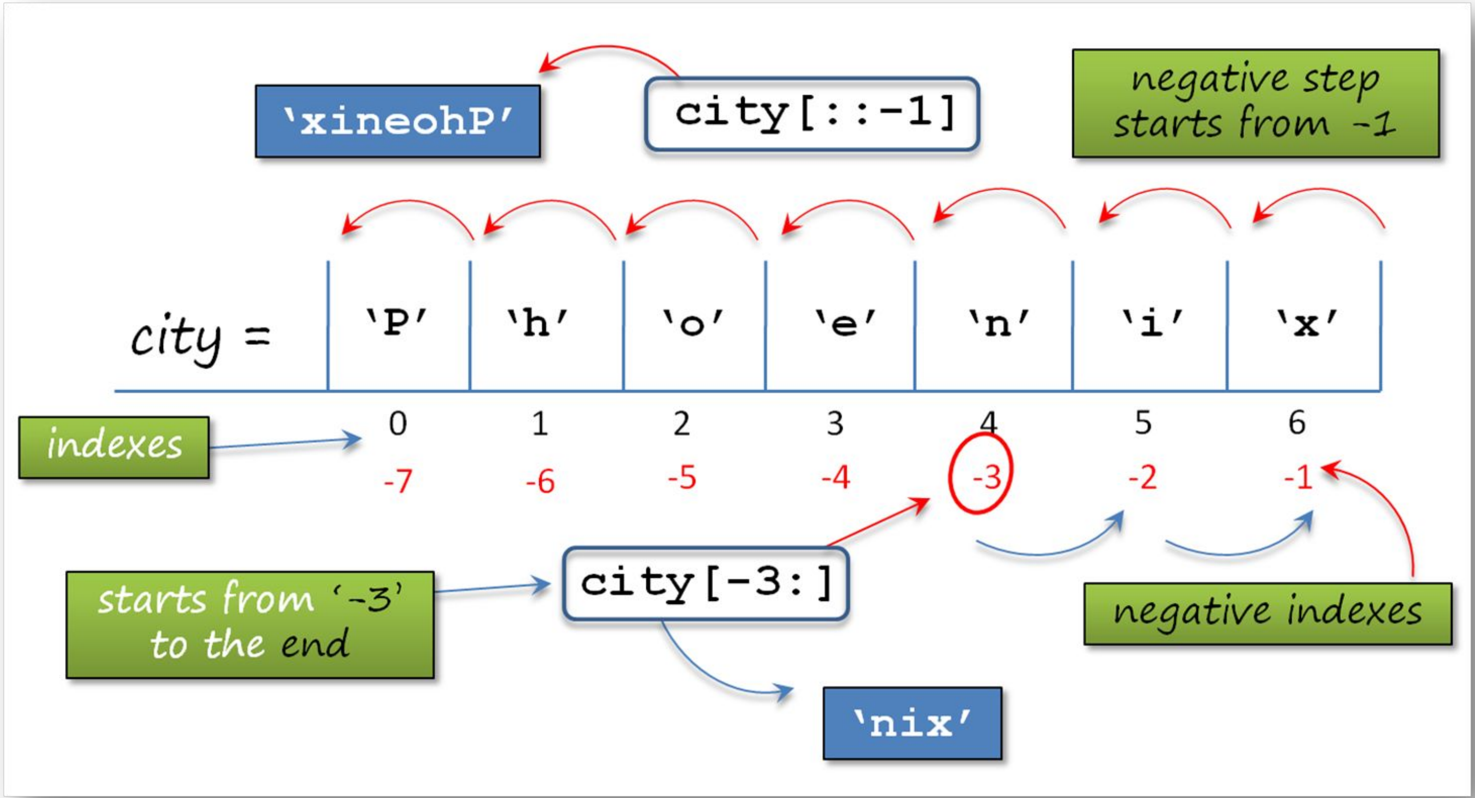


```python
city = 'Phoenix'

print(city[1:])  # starts from index 1 to the end
print(city[:6])  # starts from zero to 5th index
print(city[::2])  # starts from zero to end by 2 step
print(city[1::2])  # starts from index 1 to the end by 2 step
print(city[-3:])  # starts from index -3 to the end
print(city[::-1])  # negative step starts from the end to zero
```



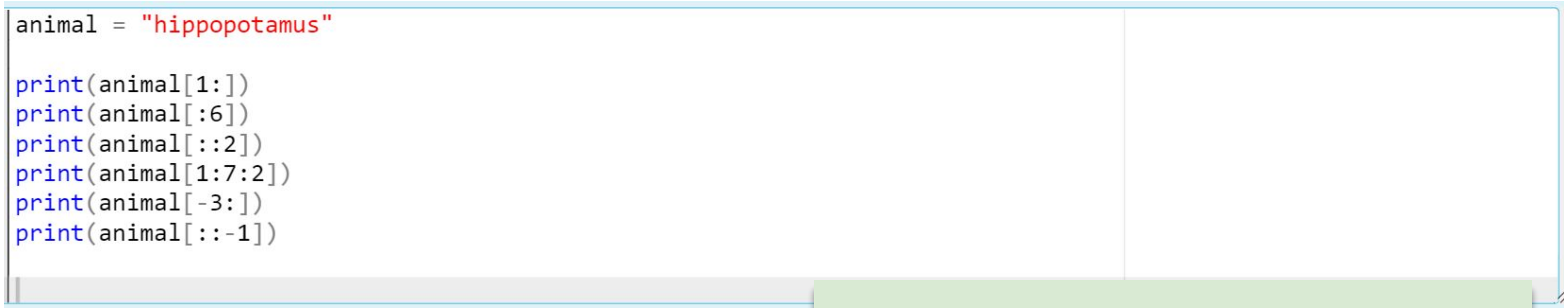
```
hoenix
Phoeni
Ponx
hei
nix
xineohP
```

Indexing String



Indexing String

Here is another example :

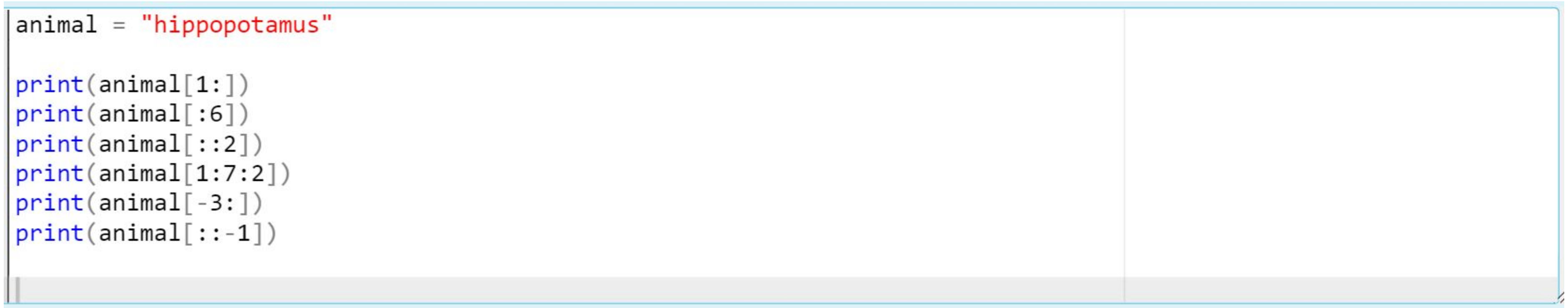


```
animal = "hippopotamus"

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

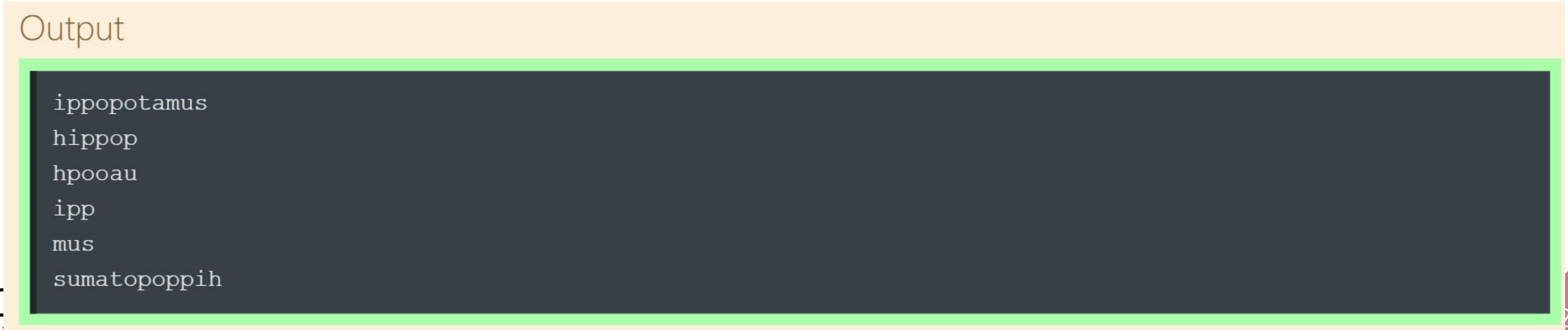
What is the output? Try to guess in your mind...

Indexing String

Here is another example :



```python
animal = "hippopotamus"

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

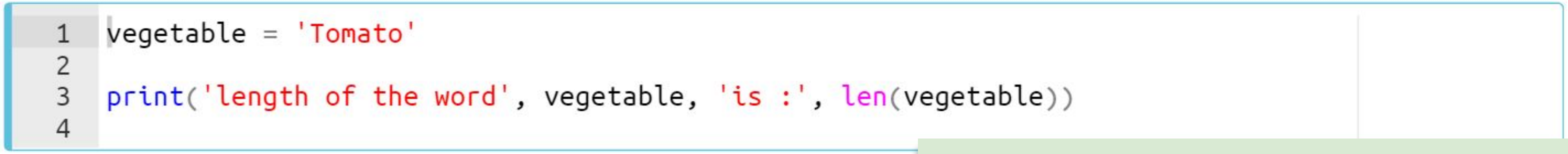


Output

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

Indexing String

- You can use the `len()` function to find out the length (number of characters) of a text or a variable of any type. It returns an `int` type.



```python
vegetable = 'Tomato'

print('length of the word', vegetable, 'is :', len(vegetable))

```

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

Indexing String

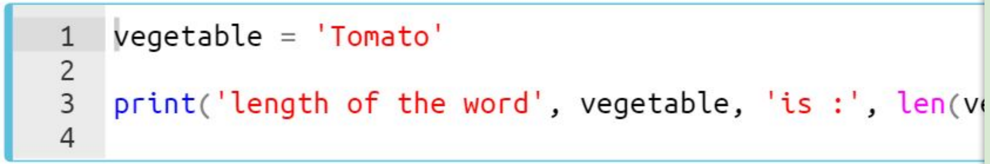
- You can use the `len()` function to find out the length (number of characters) of a text or a variable of any type. It returns an `int` type.

```python
vegetable = 'Tomato'

print('length of the word', vegetable, 'is :', len(vegetable))

```

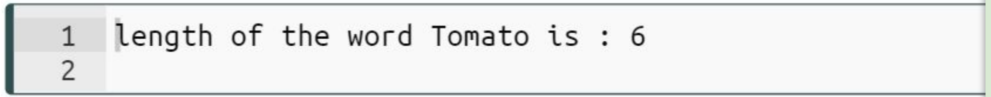
```
length of the word Tomato is : 6

```

Indexing String

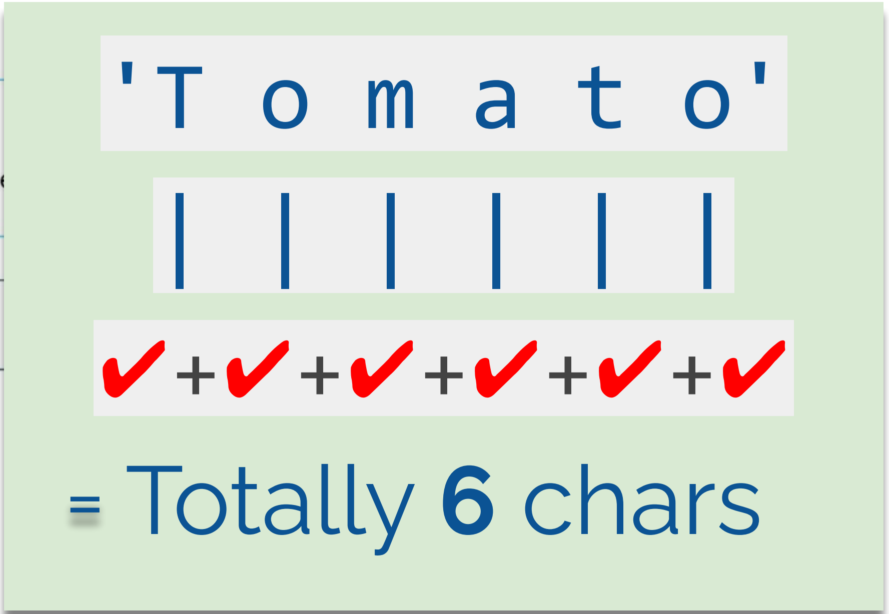
- You can use the `len()` function to find out the length (number of characters) of a text or a variable of any type. It returns an `int` type.



```
vegetable = 'Tomato'

print('length of the word', vegetable, 'is :', len(ve
```



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



String Formatting

2

String Formatting with Arithmetic Syntax

String Formatting with Arithmetic Syntax

- We can use arithmetic operator syntaxes in string formatting operations
- 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 = 3 * str_one
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 = 3 * str_one
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 = 3 * str_one
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

- 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
```

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...'

string_2 = '1453'

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

How many *space* chars here?

```
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
```

Students, write your response!

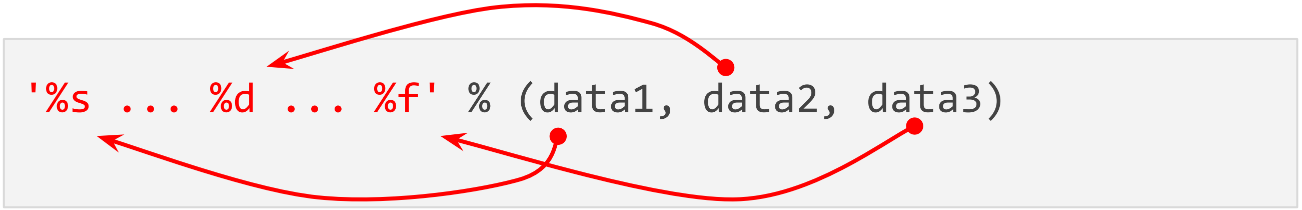
String Formatting

3 String Formatting with `%` Operator

String Formatting with % Operator

- In this way, % Operator gets the values in order and prints them in order using several characters accordingly. The basic chars we use are :

- s string
- d digit/numeric
- f float



```
'%s ... %d ... %f' % (data1, data2, data3)
```

```
data1 ... data2 ... data3
```

String Formatting with % Operator

- Here are the examples :

- s
- d
- f

```
phrase1 = 'There are 3 %s in the game room'
phrase2 = 'There is only a %.5s here'
print(phrase1 % 'children')
print(phrase2 % 'children')
print('There is only a %.5s here' % 'children')
print('1 %s 2 %s 6 %s' % ('one', 'two', 'six'))
```

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

String Formatting with % Operator

- Here are the examples :

- s
- d
- f

```
phrase1 = 'There are 3 %s in the game room'
phrase2 = 'There is only a %.5s here'
print(phrase1 % 'children')
print(phrase2 % 'children')
print('There is only a %.5s here' % 'children')
print('1 %s 2 %s 6 %s' %.('one', 'two', 'six'))
```

```
File "code.py", line 6
    print('1 %s 2 %s 6 %s' %.('one', 'two', 'six'))
                            ^
SyntaxError: invalid syntax
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

String Formatting with % Operator

- Here are the examples :

- s
- d
- f

```
phrase1 = 'There are 3 %s in the game room'
phrase2 = 'There is only a %.5s here'
print(phrase1 % 'children')
print(phrase2 % 'children')
print('There is only a %.5s here' % 'children')
print('1 %s 2 %s 6 %s' % ('one', 'two', 'six'))
```

```
There are 3 children in the game room
There is only a child here
There is only a child here
1 one 2 two 6 six
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

String Formatting with % Operator

- Here are the examples :

- s
- d
- f

```python
phrase1 = 'There are %d child and %d cats here'
phrase2 = (1, 3)
print(phrase1 % phrase2)
print(phrase1 % (1, 3))
print('There are %d child and %d cats here' % (1, 3))
```

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

String Formatting with % Operator

- Here are the examples :

- s
- d
- f

```
phrase1 = 'There are %d child and %d cats here'
phrase2 = (1, 3)
print(phrase1 % phrase2)
print(phrase1 % (1, 3))
print('There are %d child and %d cats here' % (1, 3))
```

```
There are 1 child and 3 cats here
There are 1 child and 3 cats here
There are 1 child and 3 cats here
```

String Formatting with % Operator

- Task:
  - Print out the following text using % Operator
  - **Output** : I have 22 $ and I bought milk for my cat.

```
my_text = 'I have .. $ and I bought .. for my ...'
```

String Formatting with % Operator

- The code is like:

```python
my_text = 'I have %d $ and I bought %s for my %s.' % (22, 'milk', 'cat')
print(my_text)
```

String Formatting with % Operator

- Here are the examples :

- s
- d
- f

```
phrase1 = 'The pi constant is %.2f'
phrase2 = 'The pi constant is %.4f'
number = 3.14159
print(phrase1 % number)
print(phrase2 % number)
print('More accurate value of pi is %.5f' % 3.14159)
```

String Formatting with % Operator

- Here are the examples :

- s
- d
- f

```
phrase1 = 'The pi constant is %.2f'
phrase2 = 'The pi constant is %.4f'
number = 3.14159
print(phrase1 % number)
print(phrase2 % number)
print('More accurate value of pi is %.5f' % number)
```

```
The pi constant is 3.14
The pi constant is 3.1416
More accurate value of pi is 3.14159
```

String Formatting with % Operator

- Here are the examples :

- s
- d
- f

Rounds the number up.

```
phrase1 = 'The pi constant is %.2f'
phrase2 = 'The pi constant is %.4f'
number = 3.14159
print(phrase1 % number)
print(phrase2 % number)
print('More accurate value of pi is %.5f' % 3.14159)
```

```
The pi constant is 3.14
The pi constant is 3.1416
More accurate value of pi is 3.14159
```

String Formatting with % Operator

- Here is the combined example :

- s
- d
- f

```
phrase = 'I have %d %s and it weigh %.2f kg each' % (2, 'cats', 1.5)
print(phrase)
```

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

String Formatting with % Operator

- Here is the combined example :

- s
- d
- f

```python
phrase = 'I have %d %s and it weigh %.2f kg each' % (2, 'cats', 1.5)
print(phrase)
```

```
I have 2 cats and it weigh 1.50 kg each
```

String Formatting

4

String Formatting with `string.format()` Method

String Formatting with `string.format()` Method

- `string.format()` method is the improved form of `%` Operator formatting :
- The value of expression comes from `.format()` method in order. Curly braces 👉 `{}` receives values from `.format()`.
- The formula syntax 👇

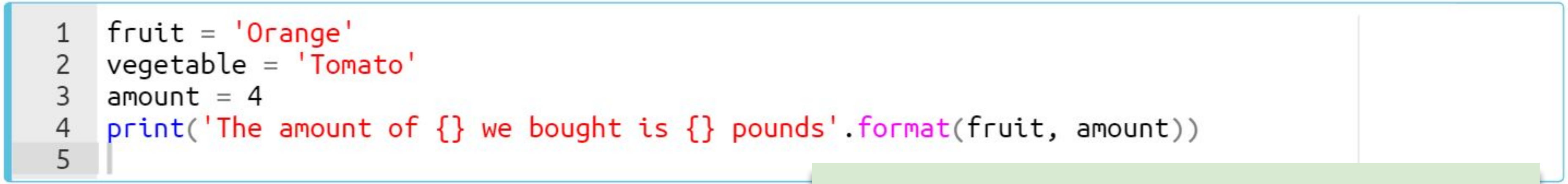
String Formatting with `string.format()` Method

- `string.format()` method is the improved form of `%` Operator formatting :
- The value of expression comes from `.format()` method in order. Curly braces 👉🏻 `{}` receives values from `.format()`.
- The formula syntax 👇

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

String Formatting with `string.format()` Method

- Take a look at the example 👇

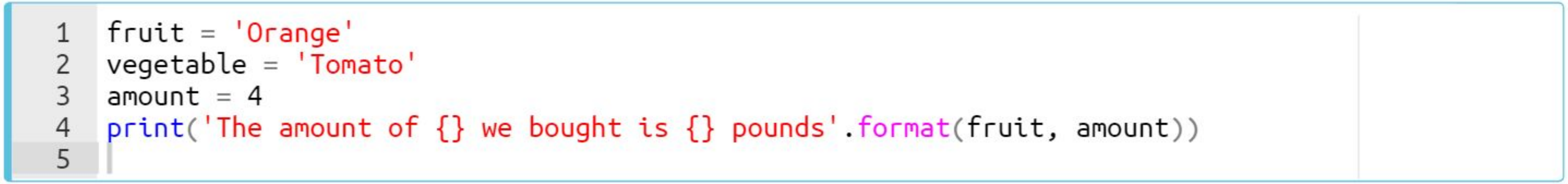


```
fruit = 'Orange'
vegetable = 'Tomato'
amount = 4
print('The amount of {} we bought is {} pounds'.format(fruit, amount))
```

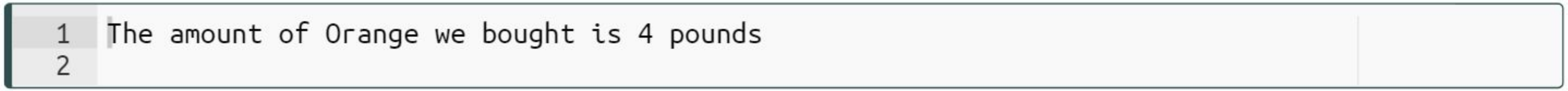
What is the output? Try to guess in your mind...

String Formatting with `string.format()` Method

- Take a look at the example 👇



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

```

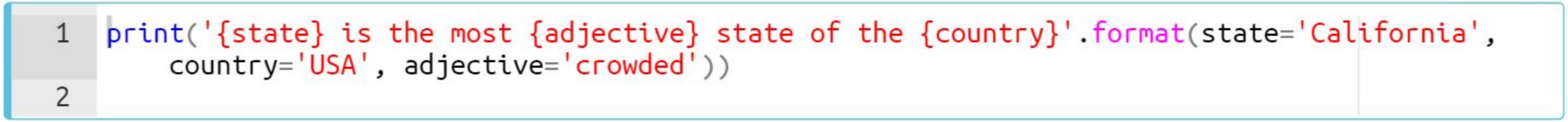


```
The amount of Orange we bought is 4 pounds

```

CLARUSWAY©
WAY TO REINVENT YOURSELF

String Formatting with `string.format()` Method

- If you've written more variables than you need in the `.format()` method, the extra ones just will be ignored. Using keywords in 👉 `{}` makes string more readable. 👇



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

CLARUSWAY©
WAY TO REINVENT YOURSELF

String Formatting with `string.format()` Method

- If you've written more variables than you need in the `.format()` method, the extra ones just will be ignored. Using keywords in 👉 `{}` makes string more readable. 👇

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

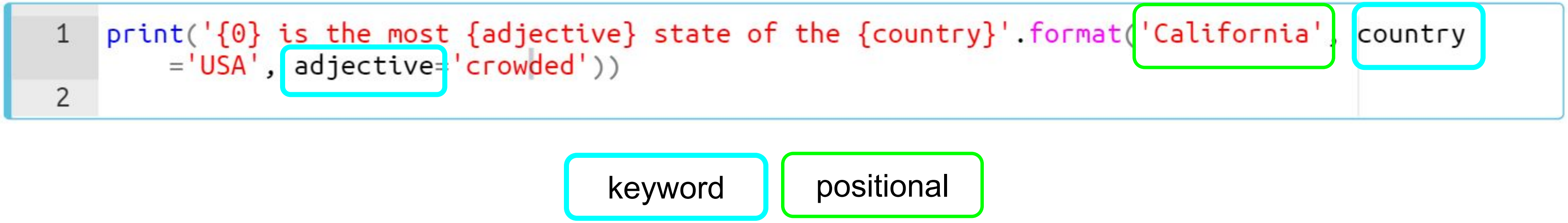
```
California is the most crowded state of the USA
```

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.



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

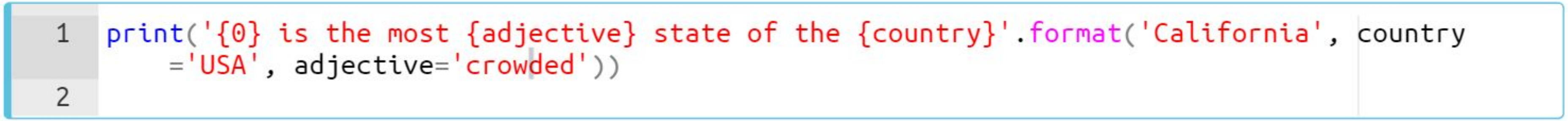
CLARUSWAY©
WAY TO REINVENT YOURSELF

String Formatting with `string.format()` Method

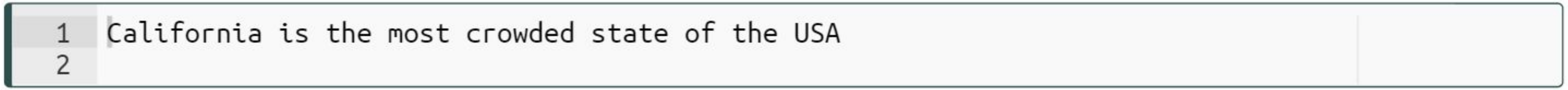
💡**Tips:**

- 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.



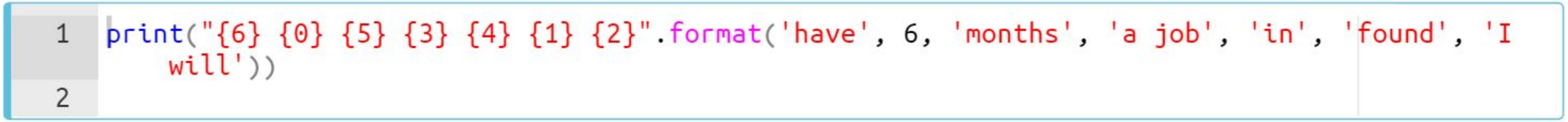
```python
print('{0} is the most {adjective} state of the {country}'.format('California', country
    ='USA', adjective='crowded'))
```



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

String Formatting with `string.format()` Method

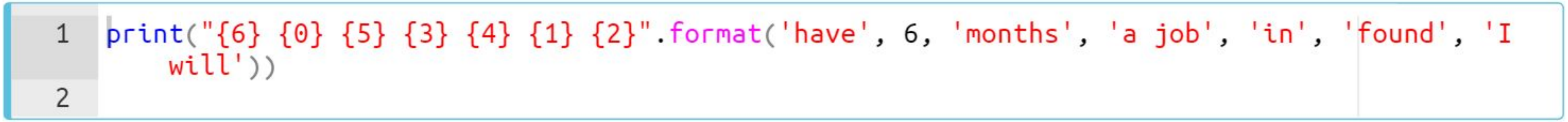
- You can use the same variable in a string more than once if you need it. Also, you can select the objects by referring to their positions in brackets.



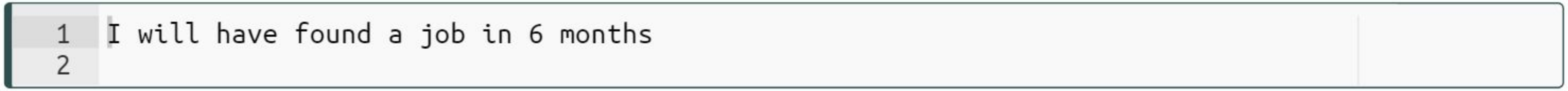
```python
print("{6} {0} {5} {3} {4} {1} {2}".format('have', 6, 'months', 'a job', 'in', 'found', 'I will'))
```

String Formatting with `string.format()` Method

- You can use the same variable in a string more than once if you need it. Also, you can select the objects by referring to their positions in brackets.



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



```
I will have found a 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)
```

Try it on Playground...

String Formatting with `string.format()` Method

- **Task :**
  - To print the statement of "**generosity wins in all circumstances**", arrange the following code using both positional and keyword arguments.

```python
condition = 'circumstances'
morality = 'generosity'

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

String Formatting with `string.format()` Method

- The code should be like these:

```
phrase = '{morality} {1} {0} {condition}'.format('in all', 'wins', condition = 
'circumstances', morality = 'generosity')

print(phrase)
```

or

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

print(phrase)
```

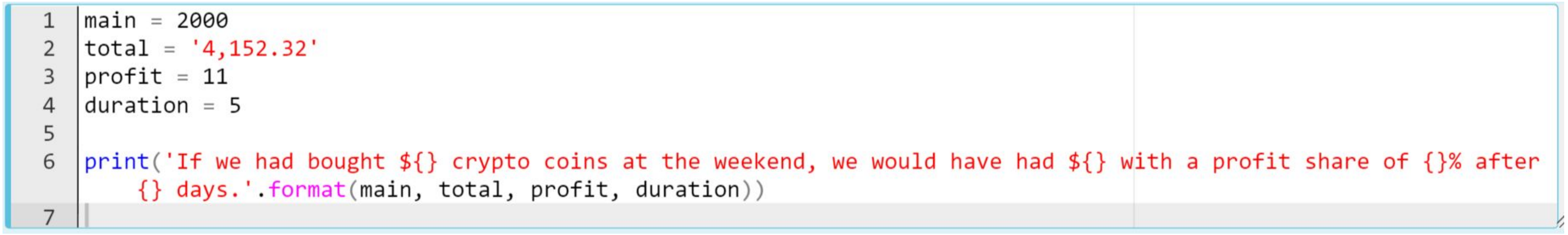
Try it on Playground...

String Formatting with `string.format()` Method

- **Task :**
  - Let’s print the text below using `.format()` method **only for numerical** text. Create variables for **numerical values** each. Take the numerical values from variables.
  - Text : “If we had bought $**2000** crypto coins at the weekend, we would have had $**4,152.32** with a profit share of **11**% after **5** days.”

String Formatting with `string.format()` Method

- One of the solutions of the code might be like this:



```python
main = 2000
total = '4,152.32'
profit = 11
duration = 5

print('If we had bought ${} crypto coins at the weekend, we would have had ${} with a profit share of {}% after
    {} days.'.format(main, total, profit, duration))

```

String Formatting with `string.format()` Method

- **Task**
  - First, Login to your LMS,
  - Then, click **here** to complete and submit the task.

String Formatting

5

String Formatting with `f-string`

String Formatting with `f-string`

- It is the easiest and useful formatting method of the strings.
- `f-string` is the string syntax that is enclosed in quotes with a letter `f` at the beginning. Curly braces 👉 `{}` that contain variable names or expressions are used to replace with their values.
- The formula syntax 👇

CLARUSWAY©
WAY TO REINVENT YOURSELF

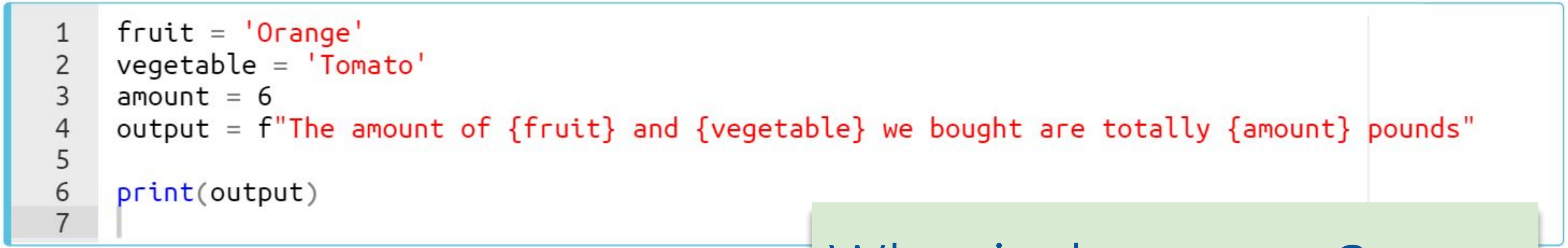
String Formatting with `f-string`

- It is the easiest and useful formatting method of the strings.
- `f-string` is the string syntax that is enclosed in quotes with a letter `f` at the beginning. Curly braces 👉 `{}` that contain variable names or expressions are used to replace with their values.
- The formula syntax 👇

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

String Formatting with `f-string`

- Take a look at the example 👇

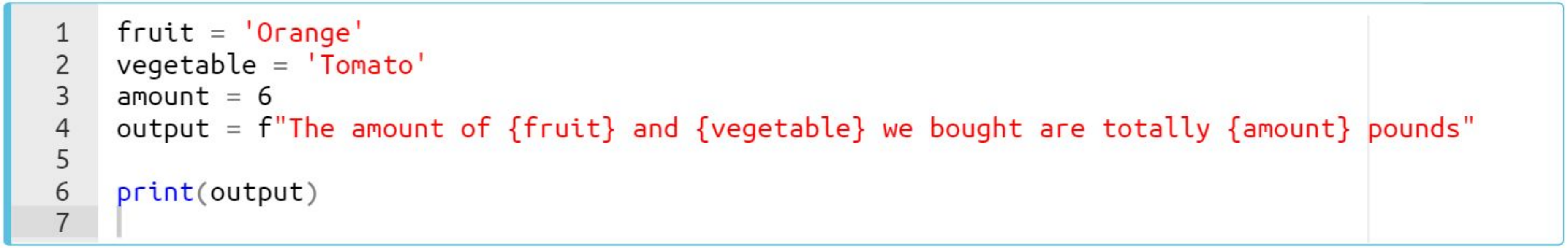


```python
fruit = 'Orange'
vegetable = 'Tomato'
amount = 6
output = f"The amount of {fruit} and {vegetable} we bought are totally {amount} pounds"

print(output)
```

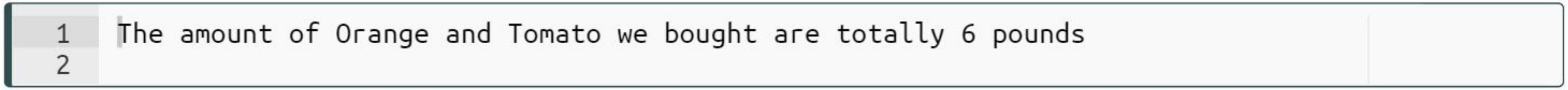
What is the output? Try to guess in your mind...

String Formatting with `f-string`

- Take a look at the example 👇



```python
fruit = 'Orange'
vegetable = 'Tomato'
amount = 6
output = f"The amount of {fruit} and {vegetable} we bought are totally {amount} pounds"

print(output)
```

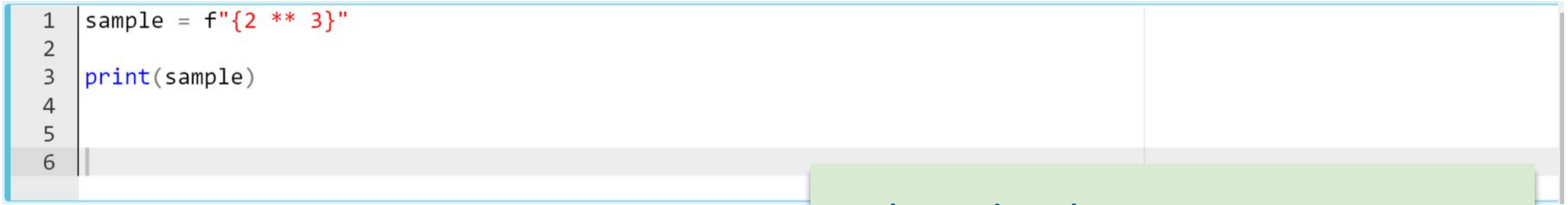


```
The amount of Orange and Tomato we bought are totally 6 pounds
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

String Formatting with `f-string`

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



```python
sample = f"{2 ** 3}"

print(sample)


```

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

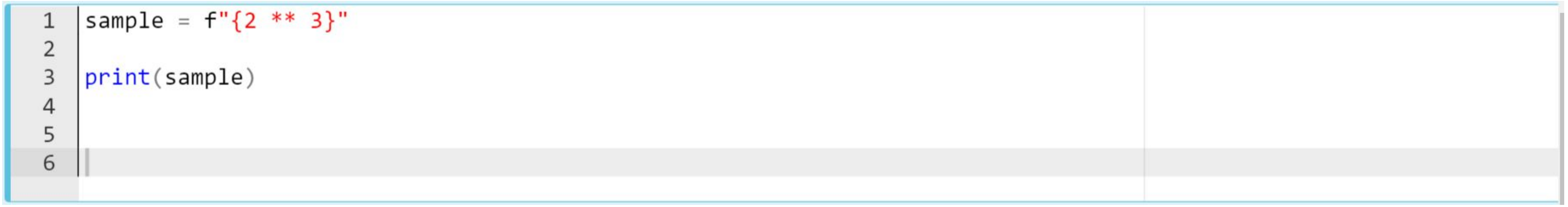
Students, write your response!

REINVENT YOURSELF

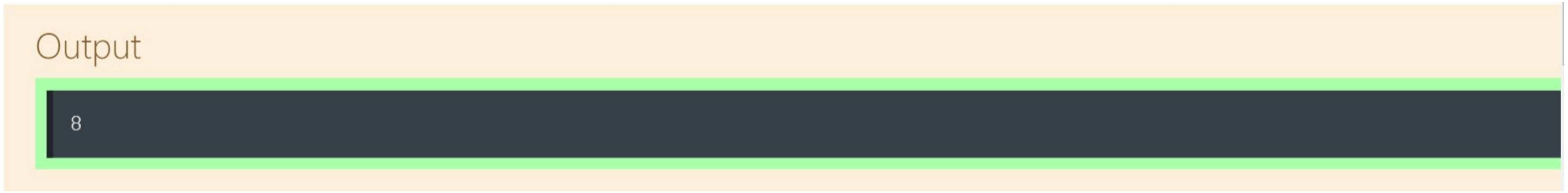
Pear Deck Interactive Slide
Do not remove this bar

String Formatting with `f-string`

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



```python
sample = f"{2 ** 3}"

print(sample)
```



Output

```
8
```

String Formatting with `f-string`

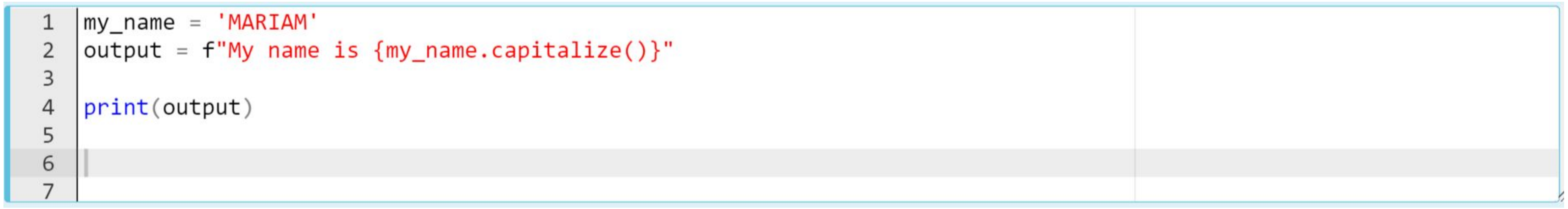
- Task :
  - Type a Python code to get the output of "**My name is Mariam**", using `.capitalize()` and `f-string` methods with the `name` variable below.

```
name = "MARIAM"
```

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

String Formatting with `f-string`

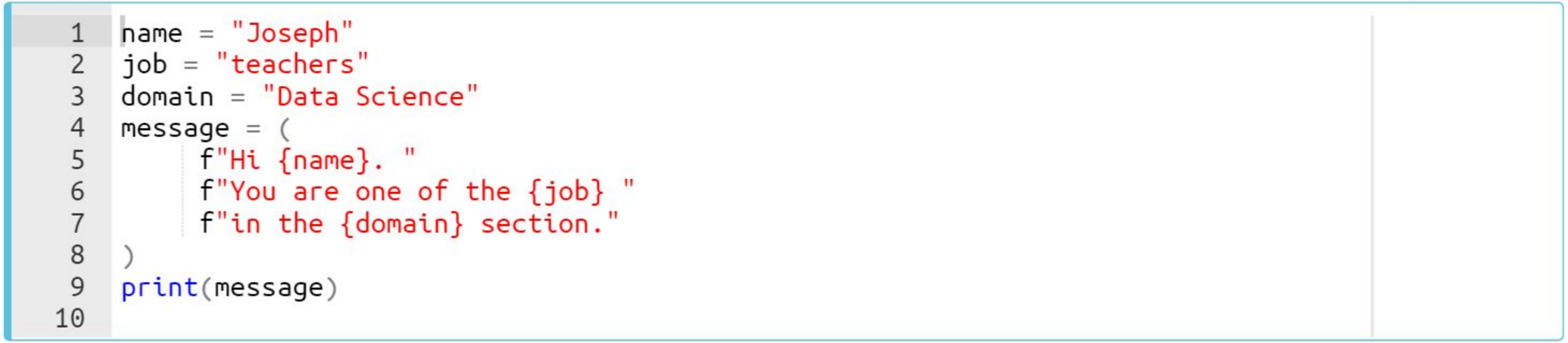
- The code should be like :



```python
my_name = 'MARIAM'
output = f"My name is {my_name.capitalize()}"

print(output)


```

String Formatting with `f-string`

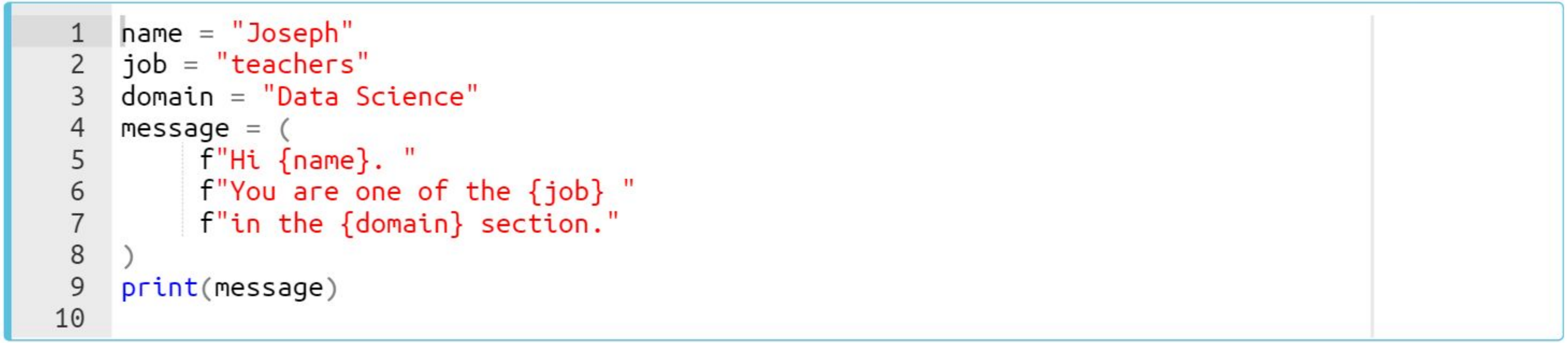
- There is also a multiline `f-string` formatting style. 👇



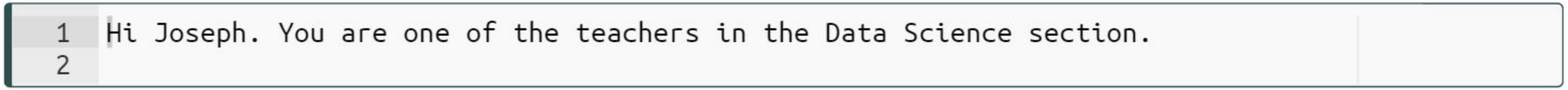
```python
name = "Joseph"
job = "teachers"
domain = "Data Science"
message = (
     f"Hi {name}. "
     f"You are one of the {job} "
     f"in the {domain} section."
)
print(message)

```

String Formatting with `f-string`

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



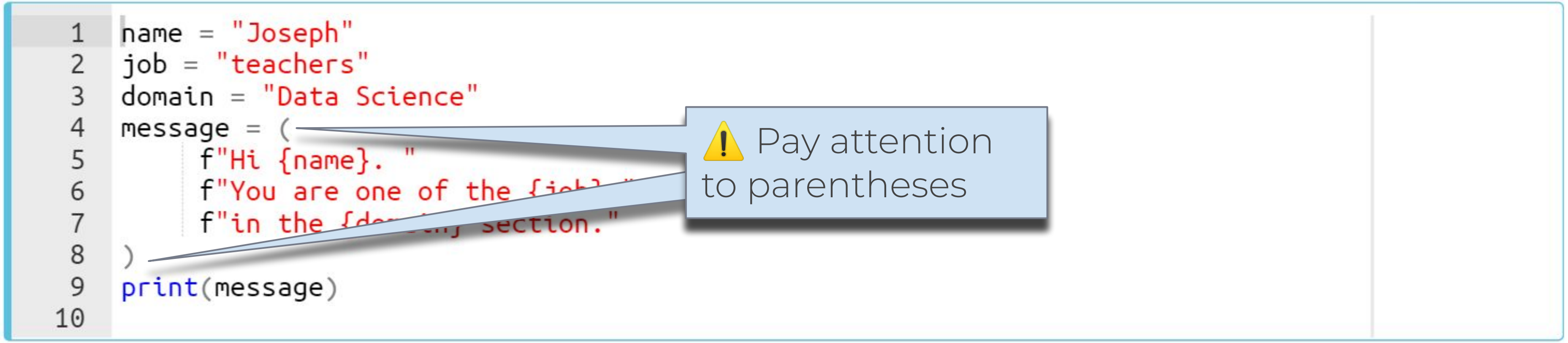
```python
name = "Joseph"
job = "teachers"
domain = "Data Science"
message = (
        f"Hi {name}. "
        f"You are one of the {job} "
        f"in the {domain} section."
)
print(message)

```



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

```

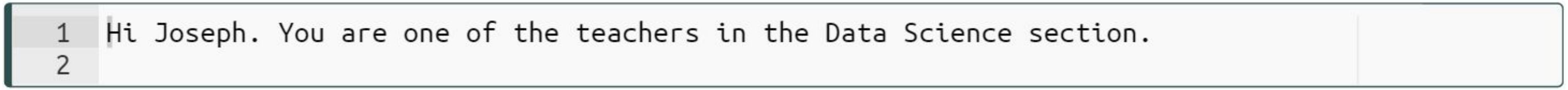
String Formatting with `f-string`

- There is also a multiline `f-string` formatting style. 👇



```python
name = "Joseph"
job = "teachers"
domain = "Data Science"
message = (
        f"Hi {name}. "
        f"You are one of the {job} "
        f"in the {domain} section."
)
print(message)
```

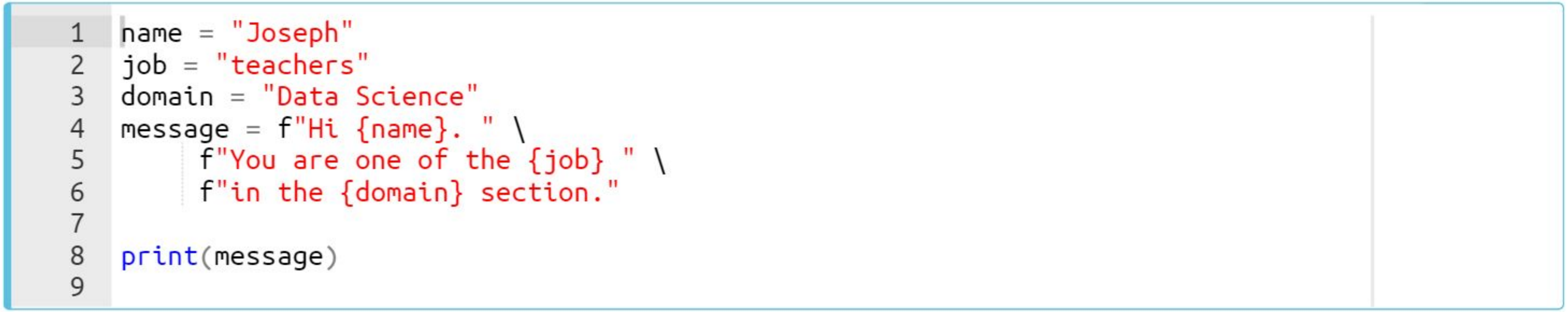
⚠️ Pay attention to parentheses



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

String Formatting with `f-string`

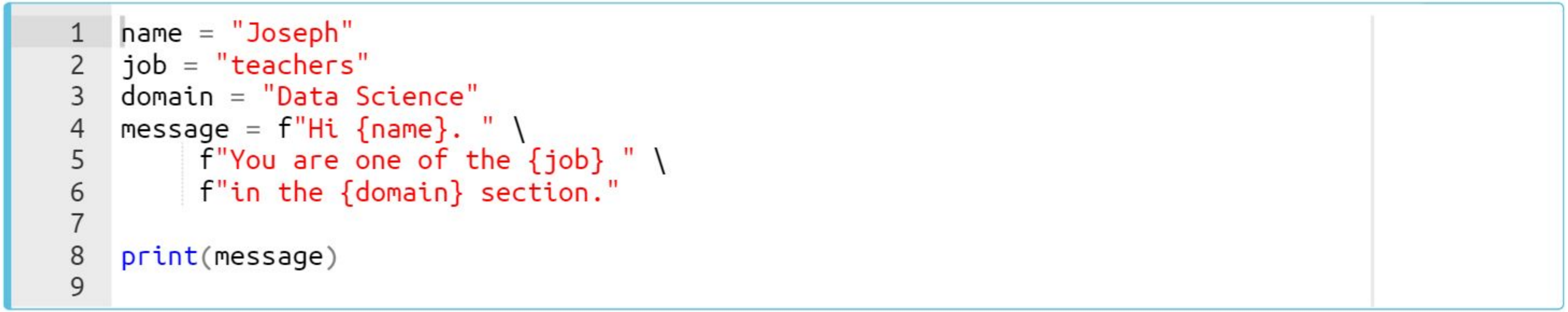
- If you want to use multiple `f-string` formatting lines *without parentheses*, you will have the other option that you can use backslash 👉 `\` between lines. 👇



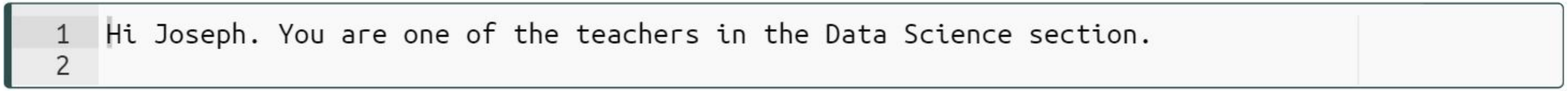
```python
name = "Joseph"
job = "teachers"
domain = "Data Science"
message = f"Hi {name}. " \
     f"You are one of the {job} " \
     f"in the {domain} section."

print(message)

```

String Formatting with `f-string`

- If you want to use multiple `f-string` formatting lines without parentheses, you will have the other option that you can use backslash 👉 `\` between lines. 👇



```python
name = "Joseph"
job = "teachers"
domain = "Data Science"
message = f"Hi {name}. " \
     f"You are one of the {job} " \
     f"in the {domain} section."

print(message)
```



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

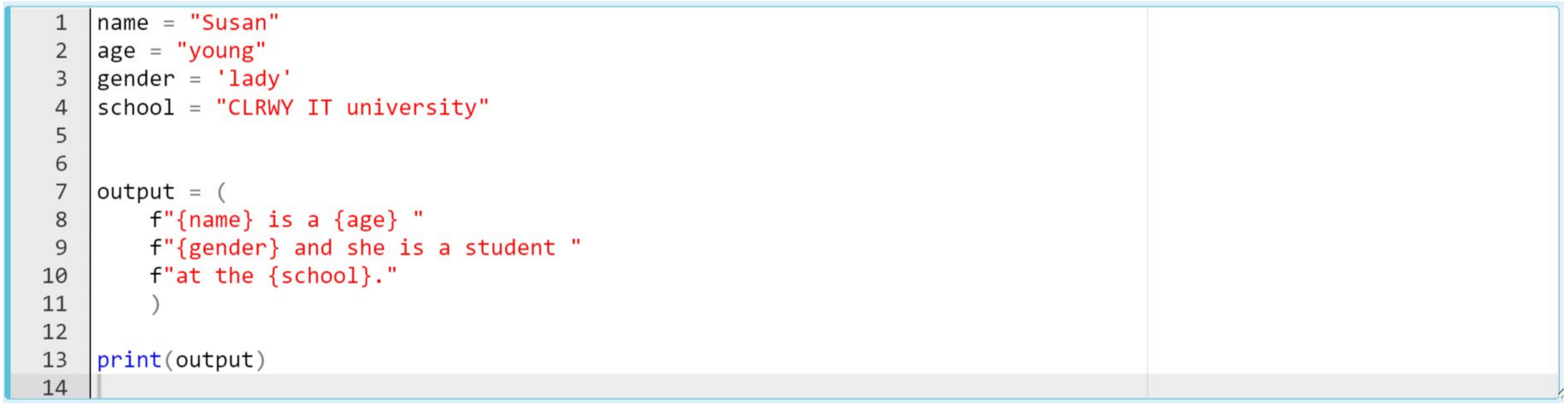
String Formatting with `f-string`

- **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` with the `variables` below.

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

String Formatting with `f-string`

- The code should be like :



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


output = (
    f"{name} is a {age} "
    f"{gender} and she is a student "
    f"at the {school}."
    )

print(output)

```

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.

Draw or type 2 things you already know about today’s topic:

1

2

JSWAY©
REINVENT YOURSELF

Students, draw anywhere on this slide!

Pear Deck Interactive Slide
Do not remove this bar

THANKS!

End of the Lesson

(Strength of the String)

next Lesson

[Main String Operations]

click above