Reverse String

This lesson teaches us how to reverse a string using a stack in Python.

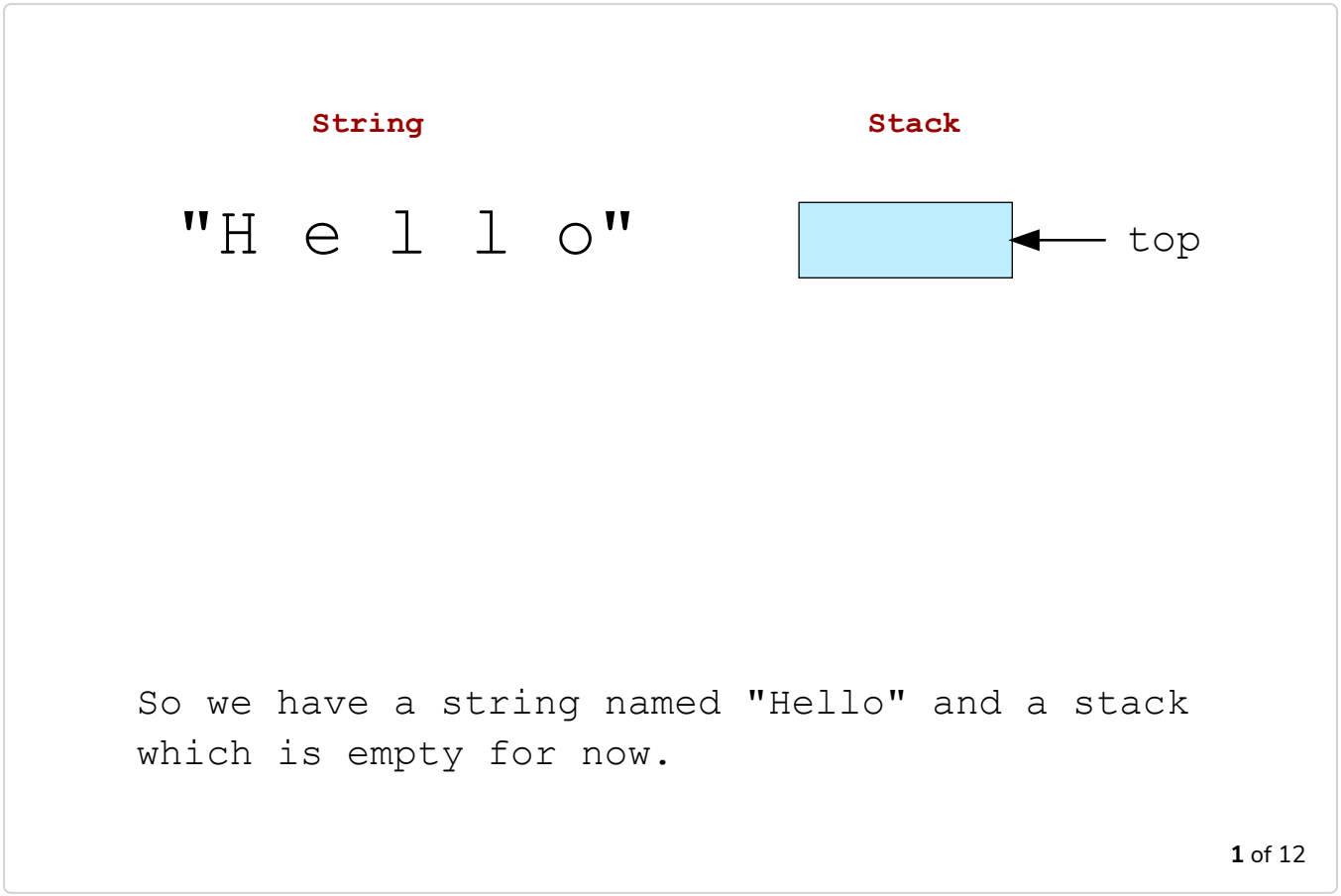
We'll cover the following

In Python, you can reverse a string very easily. For example,

```
input_str = "Educative"
print(input_str[::-1])
```

However, if you are required to reverse a string using a stack, you can make use of the following algorithm illustrated in the slides below:

Algorithm



Isn't the algorithm pretty straightforward? We push all the characters of the string onto the stack, and due to the *First-In, Last-Out* property of stack, we get all the characters in reverse order when we pop them off the stack.

Now all that is left for us is to code the algorithm shown above. Let's do it!

The stack implementation from the first lesson of this chapter has been provided to you in the file `stack.py`. Check out the code below and feel free to play around with it:

Implementation

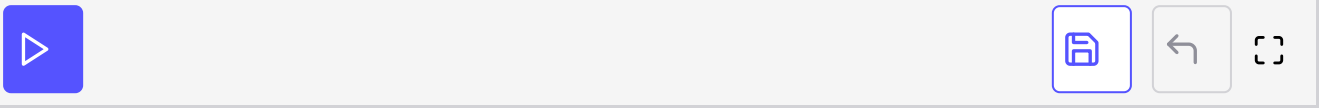
main.py

stack.py

```
from stack import Stack
def reverse_string(stack, input_str):
  for i in range(len(input_str)):
    stack.push(input_str[i])
  rev_str = ""
  while not stack.is_empty():
    rev_str += stack.pop()

  return rev_str
```

```
stack = Stack()
input_str = "!evitacudE ot emocleW"
print(reverse_string(stack, input_str))
```



Explanation

Let's discuss the `reverse_string` function in the code above. The `for` loop on **line 3** iterates over `input_str` and pushes each character on to `stack`. Then we initialize `rev_str` as an empty string. We pop off all the elements from the stack and append them to `rev_str` one by one on **line** 7 until the stack becomes empty and terminates the `while` loop on **line 6**. We return the `rev_str` on **line 9**.

Easy, right? You can test your understanding and skills in the exercise in the next lesson.