Python's Bisect Method

In this lesson, you will learn about the bisect module in Python.

WE'LL COVER THE FOLLOWING

In this lesson, we will learn about a function that takes an array of sorted integers and a key and returns the index of the first occurrence of that key from the array.

For example, for the array:

```
[-14, -10, 2, 108, 108, 243, 285, 285, 285, 401]
```

with

```
target = 108
```

the function would return `3`, as the first occurrence of `108` in the above array is located at index `3`.

We introduce to you: the `bisect` module in Python! Bisect is a built-in binary search method in Python. It can be used to search for an element in a sorted list. Let's see how we make use of different methods provided by the `bisect` module.

`bisect_left()`

The `bisect_left` function finds the index of the target element. In the event

where duplicate entries are satisfying the target element, the `bisect_left`

function returns the left-most occurrence. The input parameters to the method are the sorted list and the target element to be searched.

Have a look at the examples below:

```
# Import allows us to make use of the bisect module.
import bisect

# This sorted list will be used throughout this lesson
# to showcase the functionality of the "bisect" method.
A = [-14, -10, 2, 108, 108, 243, 285, 285, 285, 401]

# -10 is at index 1
print(bisect.bisect_left(A, -10))

# First occurrence of 285 is at index 6
print(bisect.bisect_left(A, 285))
```

`bisect_right()`

The `bisect_right` function returns the insertion point which comes after, or to the right of, any existing entries of the target element in the list. It takes in a sorted list as the first parameter and the target element to be searched as the second parameter.

```
# Import allows us to make use of the bisect module.
import bisect

# This sorted list will be used throughout this lesson
# to showcase the functionality of the "bisect" method.
A = [-14, -10, 2, 108, 108, 243, 285, 285, 285, 401]

# Index position to right of -10 is 2.
print(bisect.bisect_right(A, -10))

# Index position after last occurrence of 285 is 9.
print(bisect.bisect_right(A, 285))
```

`bisect()`

There is also just a regular default `bisect` function. This function is equivalent to `bisect_right` and takes a sorted list and the target element to be searched as input parameters:

```
# Import allows us to make use of the bisect module.
import bisect

# This sorted list will be used throughout this lesson
# to showcase the functionality of the "bisect" method.
A = [-14, -10, 2, 108, 108, 243, 285, 285, 285, 401]

# Index position to right of -10 is 2. (Same as bisect_right)
print(bisect.bisect(A, -10))

# Index position after last occurrence of 285 is 9. (Same as bisect_right).
print(bisect.bisect(A, 285))
```

`insort_left()` and `insort_right`

Given that the list `A` is sorted, it is possible to insert elements into `A` so that the list remains sorted. Functions `insort_left` and `insort_right` behave in a similar way to `bisect_left` and `bisect_right`, only the *insort* functions insert at the index positions. The input parameters to the method are the sorted list and the element to be inserted at a position so that the list remains sorted.

```
# Import allows us to make use of the bisect module.
import bisect

# This sorted list will be used throughout this lesson
# to showcase the functionality of the "bisect" method.
A = [-14, -10, 2, 108, 108, 243, 285, 285, 285, 401]


print(A)
bisect.insort_left(A, 108)
print(A)

bisect.insort_right(A, 108)
print(A)
```

I encourage you to know how to write your own binary search because that's going to be very useful, especially in the context of an interview. However, if

you want to apply the binary search in Python, then you can utilize this module.

In the context of an interview, if you mention that you know how to use the `bisect` module, that could be perhaps a feather in your cap because it shows a bit of maturity with the language and it shows that you know some of lesser-known features of Python.

By now, you have had good practice with solving problems using binary search, so let's get ready for some challenges in the next few lessons!