Lambda

-What is a Lambda Function?

Lambda functions are similar to user-defined functions but without a name. They're commonly referred to as anonymous functions.

Lambda functions are efficient whenever you want to create a function that will only contain simple expressions – that is, expressions that are usually a single line of a statement. They're also useful when you want to use the function once.

-How to Define a Lambda Function

You can define a lambda function like this:

```
lambda argument(s) : expression
```

1) `lambda` is a keyword in Python for defining the anonymous function.
2) `argument(s)` is a placeholder, that is a variable that will be used to hold the value you want to pass into the function expression. A lambda function can have multiple variables depending on what you want to achieve.
3) `expression` is the code you want to execute in the lambda function.

- Notice that the anonymous function does not have a return keyword. This is because the anonymous function will automatically return the result of the expression in the function once it is executed.

-When Should You Use a Lambda Function?

- You should use the lambda function to create simple expressions. For example, expressions that do not include complex structures such as if-else, for-loops, and so on.

- So, for example, if you want to create a function with a for-loop, you should use a user-defined function.

-How to Use a Lambda Function with Iterables

- An iterable is essentially anything that consists of a series of values, such as characters, numbers, and so on.

- In Python, iterables include strings, lists, dictionaries, ranges, tuples, and so on. When working with iterables, you can use lambda functions in conjunction with two common functions: `filter()` and `map()`.