Day 13:

**Lambda Functions (Anonymous Functions)**:

* + A **lambda function** is a concise way to create small, throwaway functions without a name.
  + They are mainly used in combination with other functions like map(), filter(), and **reduce().**
  + The syntax for a lambda function is: **lambda arguments: expression.**
  + For example, a lambda that adds two numbers: **lambda x, y: x + y**.

**The map() Function**:

* The **map()** function applies a given function to each element in an iterable (e.g., a list) and returns an iterator with the results.
* Syntax: **map(function, iterable(s))**
* Ex: fruit = ["Apple", "Banana", "Pear", "Apricot", "Orange"]

starts\_with\_A = lambda s: s[0] == "A"

result = list(map(starts\_with\_A, fruit))

# Result: [True, False, False, True, False]

**The filter() Function**:

* The filter() function filters elements from an iterable based on a condition specified by a function.
* It returns an iterator containing only the elements that satisfy the condition.

Ex: data = [1, 2, 3, 4, 5]

even\_numbers = list(filter(lambda x: x % 2 == 0, data))

# Result: [2, 4]

**The reduce() Function**:

* The **reduce()** function (from the **functools** module) applies a binary function cumulatively to the items of an iterable.
* It reduces the iterable to a single value.

Ex: from functools import reduce

numbers = [1, 2, 3, 4]

product = reduce(lambda x, y: x \* y, numbers)

# Result: 24