Assignment 24

# Q1

The def statement and lambda expressions in Python are both used to create functions, but they have some notable differences. The def statement is used to define named functions with a block of code enclosed within the function body. It requires a function name, which is used to call the function later. The function body of a def statement can consist of multiple lines of code, and the return statement is used to explicitly return a value from the function.

On the other hand, lambda expressions are anonymous functions created using the lambda keyword. They don't have a function name and are often assigned to variables or used as arguments to other functions. Lambda expressions have a concise one-line syntax and are limited to a single expression in their function body. They implicitly return the result of the expression without the need for a return keyword.

# Q2

Lambda functions are a type of anonymous function in Python. They are defined using the lambda keyword, and they can be used anywhere that a regular function can be used.

Lambda functions have a few benefits over regular functions:

They are shorter and more concise.

They can be used as arguments to other functions.

They can be used as inline functions.

# Q3

Map:

The map function applies a given function to each element of an iterable and returns a new iterable containing the results.

map(function, iterable)

Produces a new iterable of the same length as the input iterable.

Transforms each element in the input iterable using the provided function.

Retains the original order of elements.

The returned iterable is typically a map object, which can be converted to other iterable types if needed.

Useful for performing element-wise transformations or calculations on iterables.

Filter:

The filter function applies a given predicate function to each element of an iterable and returns a new iterable containing only the elements that satisfy the condition.

filter(predicate, iterable)

Produces a new iterable containing elements that pass the condition specified by the predicate function.

Retains the original order of elements.

The returned iterable is typically a filter object, which can be converted to other iterable types if needed.

Useful for selecting or filtering elements based on a given condition.

Reduce:

The reduce function applies a given binary function to the elements of an iterable in a cumulative way and returns a single value.

reduce(function, iterable)

Applies the given function to the first two elements of the iterable, then to the result and the next element, and so on, until a single value is obtained.

Reduces the iterable to a single value by performing a cumulative computation.

Does not retain the original order of elements.

Useful for aggregating or combining the elements of an iterable into a single result.

# Q4

Function annotations in Python are a way to associate metadata or additional information with function parameters and return values. They provide a way to specify the expected types, clarify the purpose, or document the behaviour of function arguments and return values. Function annotations are optional and do not affect the runtime behaviour of the function.

Function annotations are defined using the syntax parameter: annotation for specifying annotations for function parameters and -> annotation for specifying the annotation for the return value.

# Q5

Recursive functions are functions that call themselves within their own body. They are a powerful programming technique used when a problem can be divided into smaller subproblems of the same nature. In recursive functions the problem is divided into smaller parts, each of which is solved recursively until a base case is reached.

The basic structure of a recursive function consists of two parts: the base case(s) and the recursive call(s). The base case(s) define the termination condition for the recursion and provide the result for the smallest subproblem. The recursive call(s) represent the self-referencing part of the function, where it calls itself with a smaller or simpler version of the original problem.

# Q6

Single responsibility principle: A function should only do one thing. This makes the function easier to understand, test, and maintain.

Don't repeat yourself (DRY): Avoid duplicating code in different functions. This makes the code more concise and easier to maintain.

Keep functions small: Functions should be short and to the point. This makes the code easier to read and understand.

Using descriptive names: The names of functions should be descriptive of what the function does. This makes the code easier to read and understand.

Using consistent formatting: The formatting of functions should be consistent throughout the code. This makes the code easier to read and understand.

Using comments: Comments should be used to explain the purpose of the function and the steps that the function takes. This makes the code easier to understand and maintain.

Test functions: Functions should be tested to ensure that they work correctly. This helps to prevent errors and makes the code more reliable.

Following these guidelines will help us to write functions that are well-designed, easy to understand, and maintainable.

# Q7

Return value: The most common way for a function to communicate results is to return a value. The return value can be any type of object, such as a number, a string, a list, or a dictionary.

Printing to the console: Functions can also communicate results by printing them to the console. This is a good way to debug functions and to see the results of their calculations.

Changing global variables: Functions can also communicate results by changing global variables. This is a good way to share data between functions.

Using exceptions: Functions can also communicate results by raising exceptions. Exceptions are used to signal errors or unusual conditions.

Writing to a file: Functions can write their results to a file. This is a good way to save the results of a calculation or to store data for later use.

Calling another function: Functions can call other functions to communicate results. This is a good way to break down a problem into smaller, more manageable tasks.