Python Theory Assignment - Answers

1. What is the difference between a function and a method in Python?

A function is a block of code that performs a specific task and can be defined using the `def` keyword. It can be called independently. A method, on the other hand, is associated with an object and is called on that object. Methods are functions that belong to objects/classes. Example: `len([1,2,3])` is a function, while `'hello'.upper()` is a method.

2. Explain the concept of function arguments and parameters in Python.

Parameters are variables defined in the function definition, whereas arguments are the actual values passed when calling the function. Example: `def greet(name):` → 'name' is a parameter. Calling `greet('Dev')` → 'Dev' is an argument.

3. What are the different ways to define and call a function in Python?

Functions in Python can be defined using the `def` keyword or lambda expressions. They can be called directly by name with parentheses. Example: `def add(a,b): return a+b` → `add(2,3)`. Example (lambda): `sum = lambda a,b: a+b` → `sum(2,3)`.

4. What is the purpose of the `return` statement in a Python function?

The `return` statement is used to send the result of a function back to the caller. Without `return`, a function returns `None` by default. Example: `def square(x): return x*x` → `square(5)` gives `25`.

5. What are iterators in Python and how do they differ from iterables?

An iterable is an object that can return an iterator, like lists, tuples, and strings. An iterator is an object with `__iter__()` and `__next__()` methods used to fetch elements one at a time. Example: `iter([1,2,3])` gives an iterator.

6. Explain the concept of generators in Python and how they are defined.

Generators are special functions that use `yield` to produce a sequence of values lazily (one at a time). They are memory efficient. Example: `def gen(): yield 1; yield 2`.

7. What are the advantages of using generators over regular functions?

- Memory efficiency (they generate values on demand). - Useful for large datasets and infinite sequences. - Easier to implement iterators. Example: Reading large files line by line using a generator.

8. What is a lambda function in Python and when is it typically used?

A lambda function is an anonymous, one-line function defined with the `lambda` keyword. It is typically used for short, temporary operations. Example: `lambda x: x*2` → used in `map()` or `filter()`.

9. Explain the purpose and usage of the `map()` function in Python.

The `map()` function applies a given function to each element of an iterable and returns an iterator. It is often used with lambda. Example: `map(lambda x: x*2, [1,2,3])` → `[2,4,6]`.

10. What is the difference between `map()`, `reduce()`, and `filter()` functions in Python?

- `map()`: Applies a function to all elements of an iterable (e.g., squaring numbers). - `filter()`: Selects elements based on a condition (e.g., even numbers). - `reduce()`: Applies a rolling computation to pairs of elements (e.g., summing a list). Example: `reduce(lambda a,b: a+b, [1,2,3,4])` → `10`.