Python Functions: lambda, map, and filter

AN INTRODUCTION OF FUNCTIONS WITH EXAMPLES

Overview

- **Lambda Functions**: Anonymous functions defined with the lambda keyword.
- **Map Function**: Applies a function to all items in an input list (or other iterable).
- **Filter Function**: Filters items in an iterable based on a function that returns True or False.

Iterable

- An **iterable** is any Python object that can return an iterator. It represents a collection of items that can be iterated (looped) over. Common examples of iterables include lists, tuples, dictionaries, sets, and strings.

Iterator

- An **iterator** is an object that represents a stream of data. It provides a way to access elements of an iterable one at a time. An iterator implements two methods:

- __iter__(): Returns the iterator object itself.
- __next__(): Returns the next item from the iterator. When there are no more items, it raises the StopIteration exception.

```python
# Create an iterator from a list
my_list = [1, 2, 3, 4, 5]
my_iter = iter(my_list)

# Using the iterator to fetch items
print(next(my_iter))  # Output: 1
print(next(my_iter))  # Output: 2
print(next(my_iter))  # Output: 3
```

Lambda Functions

- **Definition:** Lambda functions are small anonymous functions defined with the lambda keyword.
- **Syntax:** lambda arguments: expression

```
# Traditional function
def square(x):
    return x * x

# Lambda function
square_lambda = lambda x: x * x

print(square_lambda(5))  # Output: 25
```

Map Function

- The map function applies a given function to all items in an input list (or other iterable).
- **Syntax:** map(function, iterable)

```
def double(x):
    return x * 2

# Using map
numbers = [1, 2, 3, 4, 5]
doubled = map(double, numbers)

print(list(doubled))  # Output: [2, 4, 6, 8, 10]
```

```
numbers = [1, 2, 3, 4, 5]
doubled = map(lambda x: x * 2, numbers)

print(list(doubled))  # Output: [2, 4, 6, 8, 10]
```

Filter Function

- The filter function filters items in an iterable based on a function that returns True or False.
- **Syntax:** filter(function, iterable)

```python
# Function to filter even numbers
def is_even(x):
    return x % 2 == 0

# Using filter
numbers = [1, 2, 3, 4, 5]
evens = filter(is_even, numbers)

print(list(evens))  # Output: [2, 4]
```

```python
numbers = [1, 2, 3, 4, 5]
evens = filter(lambda x: x % 2 == 0, numbers)

print(list(evens))  # Output: [2, 4]
```

Use Cases

- **Lambda Functions**: Quick, small functions where defining a full function is unnecessary.
- **Map Function**: Transform data efficiently, apply operations to each item in a collection.
- **Filter Function**: Select items that meet certain criteria, useful for cleaning or processing data.

THANK YOU

HAPPY LEARNING!