1. enumerate ()

Explanation: enumerate () is used to iterate over a sequence (like a list, tuple, or string) while keeping track of the index of the current item. It returns pairs of index and item.

Syntax: enumerate (iterable, start=0)

Iterable: The sequence you want to iterate over.

Start: The index from which the counting should begin. The default is 0.

Example:

```
fruits = ['apple', 'banana', 'orange']
for index, fruit in enumerate (fruits, start=1):
    print (f "Index {index}: {fruit}")
```

2. Reduce ()

Explanation: reduce () applies the specified function cumulatively to the items of an iterable, from left to right, reducing the iterable to a single accumulated result.

Syntax:

```
from functools import reduce
reduce (function, iterable, initializer=None)
```

Example:

```
from functools import reduce

numbers = [1, 2, 3, 4, 5]

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

print(product)
```

3. map ()

Explanation: map () applies the specified function to all items in an input iterable (or iterables) and returns an iterator that produces the results.

Syntax: map (function, iterable, ...)

Function: The function to apply to each item in the iterable.

Iterable: The iterable (e.g., list, tuple) whose elements will be processed by the function.

Example:

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

4. filter ()

Explanation: filter () constructs an iterator from elements of the iterable for which the specified function returns true.

Syntax:

```
filter (function, iterable)
```

Example:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
even_numbers = filter (lambda x: x % 2 == 0,
numbers)
print(list(even_numbers))
```

5. zip ()

Explanation: zip () aggregates elements from two or more iterables and returns an iterator of tuples where the i-th tuple contains the i-th elements from each of the input iterables.

Syntax:

```
zip (iterable1, iterable2, ...)
```

Example:

```
names = ['Alice', 'Bob', 'Charlie']
ages = [25, 30, 22]

combined = zip (names, ages)
print(list(combined))
```

6. id ()

Explanation: id () returns the identity (unique integer) of an object. This identity is unique and constant for the object during its lifetime.

Syntax:

```
id(object)
```

Example:

```
x = 42
y = x
z = 42

print(id(x)) # Identity of x
print(id(y)) # Identity of y (same as x)
print(id(z)) # Identity of z (may or may not be the same as x and y)
```