

#

=====

=====

=====

=====

=====

1) enumerate

The enumerate() function takes a collection (e.g. a tuple, a list) and returns it as an enumerate object.

and enumerate() function adds a counter as the key of the enumerate

object.

Syntax

enumerate(iterable, start)

Example

Convert a list into an enumerate object:

t = ('apple', 'banana', 'cherry')

y = enumerate(t)

`print(list(y))`

#

=====

=====

=====

=====

=====

#

=====

=====

=====

=====

=====

2) Reduce()

The `reduce(fun, seq)` function is used to apply a particular function passed in its argument to all of the list elements

mentioned in the sequence passed along.

This function is defined in
"functools" module.

Syntax

functools.reduce(function,
iterable[initializer])

#

Example---

python code to demonstrate
working of reduce()

import functools

```
# # initializing list
```

```
lis = [1, 3, 5, 6, 2]
```

```
# using reduce to compute sum of list
```

```
print("The sum of the list elements is", end="")
```

```
print(functools.reduce(lambda a, b: a+b, lis))
```

```
# using reduce to compute maximum  
element from list
```

```
print("The maximum element of the  
list is :", end="")
```

```
print(functools.reduce(lambda a, b: a  
if a > b else b, lis))
```

```
#
```

```
=====
```

```
=====
```

```
=====
```

```
=====
```

```
=====
```

#

=====

=====

=====

=====

=====

3). map() function returns a map
object(which is an iterator) of the

results after applying the given function to each

item of a given iterable (list, tuple etc.)

Syntax:

map(fun, iter)

Parameters:

fun: It is a function to which map passes each element of given iterable.

iter: It is iterable which is to be

mapped.

`## Example`

`## Python program to demonstrate
working of map.`

`# Return double of n`

`def addition(n):`

`return n + n`

`# We double all numbers using map()`

`numbers = (1, 2, 3, 4)`

```
result = map(addition, numbers)
```

```
print(list(result))
```

```
#
```

```
=====
```

```
=====
```

```
=====
```

```
=====
```

```
=====
```

#

=====

=====

=====

=====

=====

4). filter() Function

The filter() function returns an iterator where the items are filtered through a function to test if the item

is

accepted or not.

Syntax:

filter(function, iterable)

Example:

ages = [5, 12, 17, 18, 24, 32]

def myFunc(x):

if x < 18:

return False

else:

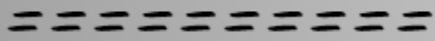
return True

adults = filter(myFunc, ages)

for x in adults:

print(x)

#



#

=====

=====

=====

=====

=====

5). zip() Function

The zip() function returns a zip object, which is an iterator of tuples where the first item in each passed iterator is paired

together, and then the second item

in each passed iterator are paired together etc.

If the passed iterables have different lengths, the iterable with the least items decides the length of the new iterator.

Syntax:

zip(iterator1, iterator2, iterator
3 ...)

Example:

```
a = ("John", "Charles", "Mike")
```

```
b = ("Jenny", "Christy", "Monica")
```

```
x = zip(a, b)
```

```
#use the tuple() function to display a  
readable version of the result:
```

```
print(tuple(x))
```

```
#
```

#

=====

=====

=====

=====

=====

6). ID() function :

id() function is a built-in function that returns the unique identifier of an object. The identifier is an integer,

which represents the memory