python methods

python is a versatile and strong programming language with a wide range of applications in several sectors. It is also a object oriented programming language which uses the classes and objects.

-> python methods are functions that are associated with an object or a class in python programming.

-> they are used to perform specific tasks or operations on the data stored within objects or classes.

** methods in python are used to define the behaviours of the python objects.

** methods are used to improve the readability and maintainability of code.

** they help in breaking down complex tasks into smaller, more manageable tasks.

types of methods

- Instance methods
- class methods
- static methods.

*enumerate()

The enumerate function in python converts a data collection object into an enumerate object. it returns an object that contains a counter as a key for each value within an object, making items within the collection easier to access.

syntax: enumerate(iterable, start=0)

parameters:

iterable: any object that supports iteration

start: the index value from which the counter is to be started, by default it is 0.

program:

```
l1 = ["eat",
l1 = ["love", "your", "self"]
s1 = "python"

obj1 = enumerate(l1)
obj2 = enumerate (s1)

print ("Return type:", type(obj1))
print (list(enumerate(l1)))
print (list(enumerate(s1,2)))
```

op - Return type: <class 'enumerate'>

```
[(0, 'eat'), (1, 'sleep'), (2, 'repeat')]
[(3, 'p'), (4, 'y'), (5, 't'), (6, 'h'), (7, 'o'), (8, 'n')]
```

* Reduce()

-> reduce() is a built-in function that applies a given function to the elements of an iterable, reducing them to a single value. the sy

-> It is a powerful tool in python that operates on a list (or any iterable), applies a function to its elements, and 'reduces' them to a single output.

-> It's part of the functools module, which needs to be imported before you can use xeduce().

-> It stores the intexmediate xesult and only xeturns the final summation value.

syntax - functools.xeduce(function, itexable [initializex])

Eg: from functools impost xeduce

```
def add(a,b):
    xeturn a+b.

num_list = [1,2,3,4,5,6,7,8,9,10]
sum = xeduce(add, num_list)
psint(f"sum of the integexss of num_list :{sum}")
sum = xeduce(add, num_list, 10)
psint(f"sum of the integexss of num_list with
        initial value 10: {sum}")
```

O/P - sum of the integexss of num_list : 55.
sum of the integexss of num_list with initial value 10: 65.

* map()

Python map() applies a function on all the items of an itexatox gives as input, An itexatox, for example can be a list, a tuple, aset, a dictionaxy, a string and it xeturn an itexable map object. Python map() is a built-in-function.

Eg: calculate the length of each word in the tuple:

```
def myfunc(n):
    return len(n)
x = map(myfunc, ('apple', 'banana', 'cherry'))
print(x)
print(list(x))
```

O/P -

<map object at 0x056D4ff0>

[5, 6, 6].

-> The map() function execute a specified function for each item in an item is sent to the function as a parameter.

Syntax

map(function, iterables)

Parameters

function - the function to execute for each item.

iterable - A sequence, collection or an iterator object. You can send as many iterables as you like, just make sure the function has one parameter for each iterable

Eg:

```
def myfun(a,b):
    return a+b
x = map(myfunc, ('apple', 'banana', 'cherry'),
        ('orange', 'lemon', 'pineapple'))
print(x)
print(list(x))
```

O/P -

<map object at 0x034244f0>

['appleorange', 'bananalemon',
'cherrypineapple'].

filter()

->It filters the given sequence with the help of function that tests each element in the sequence to be true or not.

syntax: filter(function, sequence)

parameters:

- function - function that tests if each element of a sequence is true or not.
- sequence - sequence which needs to be filtered, it can be sets, lists, tuples, or containers of any iterators.

-> the filter function along with a custom function "fun()" to filter out words from the python list.

```
def fun(variable):
    letters = ['a', 'e', 'i', 'o', 'u']
    if (variable in letters):
        return True
    else:
        return False
sequence = ['g', 'e', 'e', 'j', 'k', 's', 'p', 'r']
filtered = filter(fun, sequence)
print("the filtered letters are:")
for s in filtered:
    print(s)
```

o/p - the filtered letters are:

e
e

*zip()

->It takes iterable containers and returns a single iterator object, having mapped values from all the containers.

->It is used to map the similar index of multiple containers so that they can be used just using a single entity.

syntax- zip(*iterators).

parameters - python iterables or containers (list, string etc.)

Return value: Returns a single iterator object.

->the zip() function is used to combine two or more lists into a single iterable, the resulting iterable contains tuples.

->where the first element from each list is paired together, the second element from each list is paired together, and so on.

eg,

```
name = ["Ram", "sita", "lakshman", "hanuma"]
roll-no = [4, 1, 3, 2]

mapped = zip(name, roll-no).
print(set(mapped))
```

op - {('sita', 1), ('lakshman', 3), ('hanuma', 2), ('Ram', 4)}

Id()

-> this function is a built-in function that returns the unique identifier of an object. the identifier is an integer, which represents the memory address of the object.

-> the id() function is commonly used to check if two variables or objects refer to the same memory location.

syntax id(object)

-> we can see the function accepts a single parameter and is used to return the identity of an object.

-> the identity has to be unique and constant for this object during its lifetime.

Eg:
```
x=42
y=x
z=42
print(id(x))
print(id(y)) # same as x
print(id(z)) # same as x and y.
```

o/p -
```
140642115230496
140642115230496
140642115230496.
```

-> python id() function examples

-> these are the diff ways.

- in built datatypes
- custom object
- with sets
- with Tuples.