Python

Python function enumerate()

Python Enumerate() is a buit-in function available with the Python library. It takes the given input as a collection or tuples and returns it as an enumerate object. The Python Enumerate() command adds a counter to each item of the iterable object and returns an enumerate object as an output string.

Example:

In this example, mylist is the list given to Enumerate function in Python. The list() function is used to display the Enumerate Python output.

```
mylist = ['A', 'B' ,'C', 'D']
e_list = enumerate(mylist)
print(list(e_list))
```

Output:

```
[(0, 'A'), (1, 'B'), (2, 'C'), (3, 'D')]
```

*Note: if you replace the green line with: e_list = enumerate(mylist,2)

the output will be:

```
[(2, 'A'), (3, 'B'), (4, 'C'), (5, 'D')]
```

Lambda expression:

In Python, an anonymous function is a function that is defined without a name.

While normal functions are defined using the **def** keyword in Python, anonymous functions are defined using the **lambda** keyword.

Hence, **anonymous** functions are also called lambda functions.

Syntax of Lambda Function:

```
lambda arguments: expression
```

Example:

Here is an example of lambda function that doubles the input value.

```
double = lambda x: x * 2
print(double(5))
```

output

```
10
```

In the above program, lambda x: x * 2 is the lambda function. Here x is the argument and x * 2 is the expression that gets evaluated and returned.

This function has no name. It returns a function object which is assigned to the identifier double. We can now call it as a normal function.

The statement:

```
double = lambda x: x * 2
```

Is nearly the same as:

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

- **Import operator module and explore its functions**

Python has predefined functions for many mathematical, logical, relational, bitwise etc operations under the module “operator”. Some of the basic functions are covered in this article.

1. **add(a, b)** :- This functions returns addition of the given arguments.
   Operation – a + b.

2. **sub(a, b)** :- This functions returns difference of the given arguments.
   Operation – a – b.
3. **mul(a, b)** :- This functions returns product of the given arguments.
   Operation – a * b.

Example:

```
# importing operator module
import operator

# Initializing variables
a = 4
b = 3
# using add() to add two numbers
print ("The addition of numbers is :",end="");
print (operator.add(a, b))
# using sub() to subtract two numbers
print ("The difference of numbers is :",end="");
print (operator.sub(a, b))
# using mul() to multiply two numbers
print ("The product of numbers is :",end="");
print (operator.mul(a, b))
```

Output:

```
The addition of numbers is :7

The difference of numbers is :1

The product of numbers is :12
```

4. **truediv(a,b)** :- This functions returns division of the given arguments.
   Operation – a / b.

5. **floordiv(a,b)** :- This functions also returns division of the given arguments. But the value is floored value i.e. returns greatest small integer.
   Operation – a // b.

6. **pow(a,b)** :- This functions returns exponentiation of the given arguments.
   Operation – a ** b.
7. **mod(a,b)** :- This functions returns modulus of the given arguments.
   Operation – a % b.
8. **lt(a, b)** :- This function is used to check if a is less than b or not. Returns true if a is less than b, else returns false.
   Operation – a < b.
9. **le(a, b)** :- This function is used to check if a is less than or equal to b or not. Returns true if a is less than or equal to b, else returns false.
   Operation – a <= b.
10. **eq(a, b)** :- This function is used to check if a is equal to b or not. Returns true if a is equal to b, else returns false.
    Operation – a == b.
11. **gt(a,b)** :- This function is used to check if a is greater than b or not. Returns true if a is greater than b, else returns false.
    Operation – a > b.
12. **ge(a,b)** :- This function is used to check if a is greater than or equal to b or not. Returns true if a is greater than or equal to b, else returns false.
    Operation – a >= b.
13. **ne(a,b)** :- This function is used to check if a is not equal to b or is equal. Returns true if a is not equal to b, else returns false.
    Operation – a != b.