Date-01_oct_2021

MAP

This function accepts another function and a sequence of 'iterable' as parameters

Gives output after applying the function to each iterable in the sequence .

SYNTAX:

map(function ,iterables)

function agrements can be user-defined function or lambda function

In [2]:
```
# Using Normal Function
def func1(x):
    return x+3

r1=map(func1,[2,4,6,8])
print(r1)
print(list(r1))
```

```
<map object at 0x000001DF07BDEEB0>
[5, 7, 9, 11]
```

In [3]:
```
# Using Normal Function
def func1(x):
    return x+3

print(list(map(func1,[2,4,6,8])))
#OR
r1=map(func1,[2,4,6,8])
print(r1)
print(list(r1))
```

```
[5, 7, 9, 11]
<map object at 0x000001DF07BD3760>
[5, 7, 9, 11]
```

In [5]:
```
# Using Lambda Functions

r2=map(lambda x:x+3,[2,4,6,8])
print(r2)
print(list(r2))
```

```
<map object at 0x000001DF07BDE970>
[5, 7, 9, 11]
```

In [9]:
```
r2=map(lambda x:x**2,[2,4,6,8])
print(r2)
print(list(r2))
```

```
<map object at 0x000001DF07BDE850>
[4, 16, 36, 64]
```

In [6]:
```
# Using Lambda Functions
a=[2,3,4,5,6]
r2=map(lambda x:x+3,a)
print(r2)
print(list(r2))
```

```
<map object at 0x000001DF07BDEFD0>
[5, 6, 7, 8, 9]
```

In [8]:
```
a=[2,3,4,5,6]
r2=map(lambda x:x**2,a)
print(r2)
print(list(r2))
```

```
<map object at 0x000001DF07BDEC40>
[4, 9, 16, 25, 36]
```

FILTER

This is used to generate an output list of values that return true

when the function is called

SYNTAX

fliter (funjction, iterable)

In [10]:
```
#  Using Normal Function

def func2(x):
    if x>5:
        return x

r3=filter(func2,[2,4,6,8])
print(r3)
print(list(r3))
```

```
<filter object at 0x000001DF07BDEF10>
[6, 8]
```

In [11]:
```
def func2(x):
        return x

r3=filter(func2,[2,4,6,8])
print(r3)
print(list(r3))
```

```
<filter object at 0x000001DF07BDEA60>
[2, 4, 6, 8]
```

In [12]:
```
# Using Lambda Functions

r4=filter(lambda x:x>5, [2,4,6,8])

print(r4)
print(list(r4))
```

```
<filter object at 0x000001DF07BDE910>
[6, 8]
```

REDUCE

This applies a provided function to 'iterables' and returns a single value

In [9]:
```
# Using Normal Function
from functools import reduce
def func3(x,y):
    if x>y:
        return x
    else:
        return y

print(reduce(func3,[8,14,6,2]))
```

```
14
```

In [10]:
```
def func3(x,y):
    return x+y

print(reduce(func3,[8,14,6,2]))
```

```
30
```

In [7]:
```
from functools import reduce
def func3(x,y):
    return x+y

print(reduce(func3,[8,14,6,2]))
```

```
30
```

In [11]:
```
from functools import reduce
def func3(x,y):
    if x>y:
        return x
    else:
        return y

print(reduce(func3,[8,14,6,2]))
```

```
14
```

In [12]:
```
# Using Lambda Function

from functools import reduce

print(reduce(lambda x,y: x if x>y else y,[8,4,6,2]))
```

```
8
```

Factorial of a number using Lambda

In [13]:
```
x = lambda num : 1 if num <= 1 else num*x(num-1)
x(6)
```

Out[13]:
```
720
```

In [21]:
```
from functools import reduce

n=int(input())
factorial = reduce(lambda x, y: x * y, range(1,n+1))

print('%d != %d' %(n, factorial))
```

```
4
4 != 24
```

PDF>>>>>>>>>>>>>>>>>>>>>" Raushan Raj "<<<<<<<<<<<<<<<<<<<<<<<<
function lecture no. 03
01 oct 2021