```
(lambda x,y,z: x+y-z)(10,20,30)
```

```
0
```

```
a=list(range(10,45,3))
print(a)
```

```
[10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43]
```

```
# perform square on a
map(lambda x: x**2,a)
```

```
<map at 0x7ae6bac1ac80>
```

```
list(map(lambda x: x**2,a))
```

```
[100, 169, 256, 361, 484, 625, 784, 961, 1156, 1369, 1600, 1849]
```

```
a='abcdefghiomnop'
list(filter(lambda x:x in ['a','e','i','o','u'],a))
```

```
['a', 'e', 'i', 'o', 'o']
```

```
#filter out even no. from list of numbers
numbers=[1,2,3,4,5,6]
even_num=list(filter(lambda x:x%2==0,numbers))
print(even_num)
```

```
[2, 4, 6]
```

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

```
[1, 4, 9, 16, 25, 36]
```

```
name=['vishnu singh', 'komalsingh','anuj']
length=lambda x: len(x)
print(length(name))
```

```
3
```

```
list(map(lambda x:len(x),name))
```

```
[12, 10, 4]
```

```
sorted(name,key=lambda x:len(x))
```

```
['anuj', 'komalsingh', 'vishnu singh']
```

```
people=[
    {'name':'vishnu','age':31,'occupation':'sme'},
    {'name':'anuj','age':27,'occupation':'teacher'},
    {'namne':'vandana','age':32,"occupation":'student'}
]
```

```
people
```

```
[{'name': 'vishnu', 'age': 31, 'occupation': 'sme'},
 {'name': 'anuj', 'age': 27, 'occupation': 'teacher'},
 {'namne': 'vandana', 'age': 32, 'occupation': 'student'}]
```

```
sorted(people,key=lambda x:x['age'])
```

```
[{'name': 'anuj', 'age': 27, 'occupation': 'teacher'},
 {'name': 'vishnu', 'age': 31, 'occupation': 'sme'},
 {'namne': 'vandana', 'age': 32, 'occupation': 'student'}]
```

```
sorted(people,key=lambda x:x['occupation'])
```

```
[{'name': 'vishnu', 'age': 31, 'occupation': 'sme'},
 {'namne': 'vandana', 'age': 32, 'occupation': 'student'},
 {'name': 'anuj', 'age': 27, 'occupation': 'teacher'}]
```

```
sorted(people,key=lambda x: x['age'])
```

```
[{'name': 'anuj', 'age': 27, 'occupation': 'teacher'},
 {'name': 'vishnu', 'age': 31, 'occupation': 'sme'},
 {'namne': 'vandana', 'age': 32, 'occupation': 'student'}]
```

```
data={'a':10,'b':20,'c':14,'d':40}
max(data,key=lambda x:data[x])
```

```
'd'
```

```
Start coding or generate with AI.
```