

## LAMBDA FUNCTION - EMİR

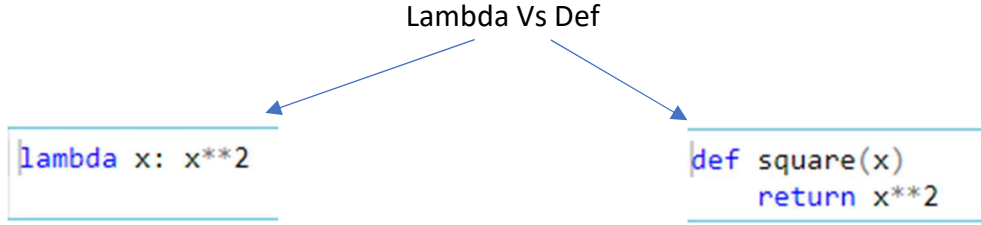
Fonksiyonları tanımlamanın başka bir yolu **lambda** işlevleridir.

Lambda'da fonksiyon isimleri verilmediği için anonim olarak adlandırılırlar.

Syntax ⇒ **lambda** parametre: ifade

### Peki neden Lambda kullanılır?

Tek seferlik bir fonksiyon olduğu için.



Bu iki işlev tamamen aynı şeyi yapar. Gelin biraz daha örneklere göz atalım:

**lambda** x, y: (x+y)/2 # İki sayıyı al ve sonucu dönder.

**lambda** x: 'tek' if x % 2 != 0 else 'çift' # Lambda içinde if koşulu da kullanılabilir.

### LMS sorusu:

If you need to use a **one-time** function, defining a **lambda** function is the best option.

You can use the **return** keyword in defining of the lambda functions.

Select one:

☐ True

☒ False ✓

### Lambda'nın avantajları:

- Syntax **Parantez** kullanarak da yazabilirsiniz
- İfadeyi bir **değişkene** de atayabilirsiniz
- Birkaç **yerleşik (built-in)** işlevde kullanabilirsiniz,
- **Def** fonksiyonu içinde kullanılabilir.

### LMS sorusu:

Fill in the blanks indicated by ... to define a **lambda** function that creates the desired number of echoes (repetition) of an entered word.

For example:

Test	Result
print(echo_word('hello', 3))	hellohellohello

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```
1 echo_word = lambda x,y: x*y
```

What is the output of the following lambda function?

```
print((lambda x: x**3)(5))
```

Answer:  ✓

## map() fonksiyonu içinde Lambda

map() belirli bir yinelenebilir nesnenin (liste, tuple, vb.) her öğesine, verilen fonksiyonu uyguladıktan sonra çıktıların bir listesini döndürür.

```
iterable = [1, 2, 3, 4, 5]
map(lambda x:x**2, iterable)
result = map(lambda x:x**2, iterable)

print(type(result))          <class 'map'>
print(type(result))          [1, 4, 9, 16, 25]
print(list(map(lambda x:x**2, iterable)))  [1, 4, 9, 16, 25]
```

### LMS sorusu:

Fill in the blanks indicated by ... to create a **lambda** function that multiplies each item in the **number\_list** by 3.  
(Note: In order to print the result, convert it into the **list** type.)

```
number_list=[1, 2, 3, 4, 5]
```

For example:

Test	Result
print(result)	[3, 6, 9, 12, 15]

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```
1 number_list = [1, 2, 3, 4, 5]
2
3 result = list(map(lambda x: x * 3, number_list))
```

## filter() fonksiyonu içinde Lambda

filter() lambda yardımıyla verilen diziyi (yinelenebilir nesneleri) filtreler.

```
first_ten = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
even = filter(lambda x:x%2==0, first_ten)
print(type(even))          <class 'filter'>
print('Even numbers are : ', list(even))  Even numbers are : [0, 2, 4, 6, 8]
```

### LMS sorusu:

Fill in the blanks indicated by ... to create a **lambda** function that filters the numbers smaller than 6 (it should give all the numbers which greater than or equal to 6) in the **number\_list**. (Note: In order to print the result, convert it into the **list** type.)

```
number_list=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

For example:

Test	Result
print(result)	[6, 7, 8, 9, 10]

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```
1 number_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2
3 result = list(filter(lambda x: x > 5, number_list))
```

## def içinde Lambda

Kullanıcı tanımlı bir işlevde bir lambda ifadesi kullanmak bize faydalı fırsatlar sağlar. Kullanabileceğimiz bir grup işlev tanımlayabiliriz.

```
def modular_function(n):  
    return lambda x: x ** n  
  
power_of_2 = modular_function(2)  
power_of_3 = modular_function(3)  
power_of_4 = modular_function(4)  
  
print(power_of_2(2)) # 2 to the power of 2      4  
print(power_of_3(2)) # 2 to the power of 3      8  
print(power_of_4(2)) # 2 to the power of 4     16
```

#### LMS sorusu

We can use the **lambda** statement in a user-defined function (**def**).

Select one:

☒ True ✓

☐ False