Lambda Functions

```
def normalFunc(x):
  # squaring the number
  return x**2
```

```
normalFunc(12)
```

```
144
```

```
type(normalFunc)
```

```
function
```

```
lambda x: x**2
```

```
<function __main__.<lambda>(x)>
```

```
a = lambda x: x**2
```

```
type(a)
```

```
function
```

```
a(12)
```

```
144
```

```
a = normalFunc(12)
print(a)
print(type(a))
b = lambda x: x**2
print(b)
print(type(b))
```

```
144
<class 'int'>
<function <lambda> at 0x7e5ed62460e0>
<class 'function'>
```

```
a = normalFunc

print(a)
print(type(a))
```

```
<function normalFunc at 0x7e5ef29c3b50>
<class 'function'>
```

```
# lambda functions can take any number of arguments

summing = lambda a,b: a+b
sub = lambda a,b: a-b

four_arg_func = lambda a,b,c,d: a+b-c*d
```



```
summing(10,20)
```

```
30
```

```
four_arg_func(12,5,8,10)
```

```
-63
```

```
def add10(x):
  return x+10

def sum(a):
  return add10(a)
```

```
sum(10)
```

```
20
```

```
def f():
  print(x)

x = 10

a = f()
print(a)
print(type(a))
```

```
10
None
<class 'NoneType'>
```

```
f2 = lambda : x

type(f2)
```

```
function
```

```
b = f2()
print(b)
print(type(b))
```

```
10
<class 'int'>
```

```
f3 = lambda : print(x)

type(f3)
```

```
function
```

```
c = f3()

print(c)
print(type(c))
```

```
10
None
<class 'NoneType'>
```

```
lf1 = lambda : 10

lf2 = lambda : print(10)
```

```
def f1(): # lf1
  return 10

def f2(): # lf2
  print(10 )


print(type(lf1))
print(type(lf2))
```

```
<class 'function'>
<class 'function'>
```

```
a = lf1()

print(a)
print(type(a))
```

```
10
<class 'int'>
```

```
b = lf2()

print(b)
print(type(b))
```

```
10
None
<class 'NoneType'>
```

```
def myFunc(n):
  return lambda a: a*n

myLambda1 = myFunc(2)


myLambda2 = myFunc(3)
myLambda2(20)


myLambda2 = myFunc(3)
myLambda2([12,20,30])
```

```
[12, 20, 30, 12, 20, 30, 12, 20, 30]
```

```
type(myLambda1)
```

```
function
```

```
myLambda1(20)
```

```
40
```

```
a = lambda x: lambda y: x+y


type(a)
```

```
function
```

```
output = a(10)

output(5)
```

```
15
```

```
a(10)(5)
```

```
15
```

Filter

```
scores = [50, 45, 10, 76, 100, 12, 5]

# create a list which takes the score greater than 40
def func(x):
  if x>40:
    return True
  else:
    return False

filtered_list = filter(func, scores)

print(filtered_list)
```

```
<filter object at 0x7e5ebfc5da80>
```

```
filtered_list = list(filter(func, scores))

print(filtered_list)
```

```
[50, 45, 76, 100]
```

```
for value in filtered_list:
  print(value)
```

```
50
45
76
100
```

```
numbers = [12,42,3,75,124,29,71,1,2,47,10001]

evenNumbers = list(filter(lambda x: x%2==0, numbers))
oddNumbers = list(filter(lambda x: x%2==1, numbers))

print(evenNumbers)
print(oddNumbers)
```

```
[12, 42, 124, 2]
[3, 75, 29, 71, 1, 47, 10001]
```

```
def isPrime(n):
  if n<=1:
    return False

  for i in range(2,n):
    if n%i == 0:
      return False

  return True

numbers = [7,17,29,144,142,2001,199,176, 187]

primeNumbers = list(filter(isPrime, numbers))

print(primeNumbers)
```

```
[7, 17, 29, 199]
```

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

```
for value in numbers:
  if isPrime(value):
    primeNumbers.append(value)

print(primeNumbers)
```

```
def myFun(a):
  if a>18:
    return False

numbers = [1,2,9,12,18,20,25,40,50]

filtered_numbers = list(filter(myFun, numbers))

print(filtered_numbers)
```

```
[]
```

```
a = myFun(2)

print(a)
```

```
None
```

```
bool(None)
```

```
False
```

```
def myFun(a):
  if a>18:
    return False
  else:
    return a+12

numbers = [1,2,9,12,18,20,25,40,50]

filtered_numbers = list(filter(myFun, numbers))

print(filtered_numbers)
```

```
[1, 2, 9, 12, 18]
```

```
def myFun(a):
  if a>18:
    return False
  else:
    return a-12

numbers = [1,2,9,12,18,20,25,40,50]

filtered_numbers = list(filter(myFun, numbers))

print(filtered_numbers)
```

```
[1, 2, 9, 18]
```

```
bool(-2)
```

```
True
```

MAP Function

```
def myFun(a):
  if a>18:
    return False

numbers = [1,2,9,12,18,20,25,40,50]

map_numbers = list(map(myFun, numbers))

print(map_numbers)
```

```
[None, None, None, None, None, False, False, False, False]
```

```
scores = [55, 75, 45, 38, 58, 67, 81, 92]

map_scores = map(lambda x: x+7, scores)

print(map_scores)
```

```
<map object at 0x7e5ebfcaa290>
```

```
for value in map_scores:
  print(value)
```

```
62
82
52
45
65
74
88
99
```

```
scores = [55, 75, 45, 38, 58, 67, 81, 92]

map_scores = list(map(lambda x: x+7, scores))

print(map_scores)
```

```
[62, 82, 52, 45, 65, 74, 88, 99]
```

```
scores = [55, 75, 45, 38, 58, 67, 81, 92]
# add 7 if score is >70 else leave as it is

map_scores = list(map(lambda x: x+7 if x>70 else x, scores))

print(map_scores)
```

```
[55, 82, 45, 38, 58, 67, 88, 99]
```

```
def statement(msg, times = 1):
  print(msg* times)

statement("Hello")
statement("World", 5)
```

```
Hello
WorldWorldWorldWorldWorld
```

```
def a(b):
  print("id of b before adding 5 : ", id(b))
  b = b+[5] # create a new list
  print("id of b after adding 5 : ", id(b))


c = [1,2,3,4]
print("id c : -->", id(c))
a(c)
print(len(c))
```

```
id c : --> 138945410319808
id of b before adding 5 :  138945410319808
id of b after adding 5 :  138945410325184
4
```

```
a = [1,2,3,4]
print("id : ", id(a))
a.append(5) # Q1 # [1,2,3,4,5] in the same id
print("id post Q1: ", id(a))
a = a+[5]  # Q2 # new list id will get created and a will point to it
print('id post Q2: ', id(a))
```

```
id :  138945409800000
id post Q1:  138945409800000
id post Q2:  138945408243776
```

```
def a(b):
  print("id of b before adding 5 : ", id(b))
  b.append(5)
  print("id of b after adding 5 : ", id(b))


c = [1,2,3,4]
print("id c : -->", id(c))
a(c)
print(len(c))
```

```
id c : --> 138946275896064
id of b before adding 5 :  138946275896064
id of b after adding 5 :  138946275896064
5
```

```
def foo(i, x=[]):
  x.append(i)
  return x

for i in range(3):
  print(foo(i))
```

```
[0]
[0, 1]
[0, 1, 2]
```

```
l = [1,2,3]
l.append(4)
l
```

```
[1, 2, 3, 4]
```

```
l = [1,2,3]
a = l.append(4)
print(a)
print(l)
```

```
None
[1, 2, 3, 4]
```

```
l = []
print(l)
for i in range(3):
  l.append(l.append(i))
  print(l)

print(l)
```

```
[]
[0, None]
[0, None, 1, None]
[0, None, 1, None, 2, None]
[0, None, 1, None, 2, None]
```

```
def foo(i, x= []):
  x.append(i)
  return x

for i in range(3):
  print(foo(i))
```

```
[0]
[0, 1]
[0, 1, 2]
```

Double-click (or enter) to edit

```
def foo(i, x= []):
  x.append(i)
  return x

for i in range(3):
  print(foo(i))
```

```
[0]
[0, 1]
[0, 1, 2]
```