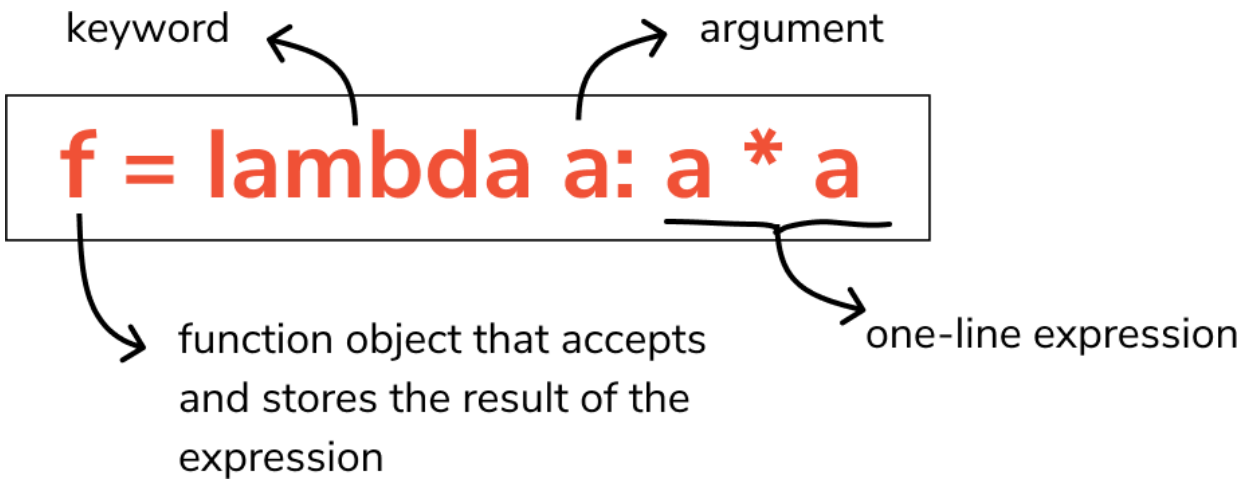
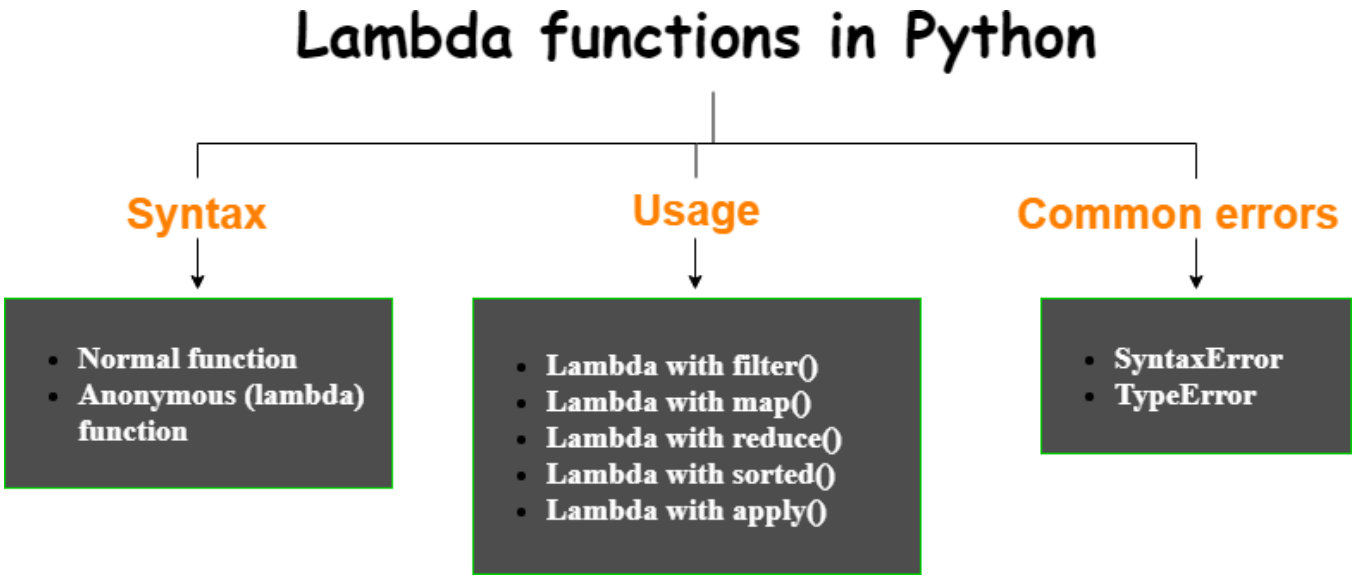
Lambda Functions in Python

- A **lambda** function is a small anonymous function.
- A **lambda** function can take any number of arguments, but can only have **one expression**.
- The expression is evaluated and returned.
- **Lambda** functions can be used wherever function objects are required.
- Lambda expressions (or lambda functions) are essentially blocks of code that can be assigned to variables, passed as an argument, or returned from a function call, in languages that support high-order functions.
- They have been part of programming languages for quite some time.
- The **main role of the lambda function** is better described in the scenarios when we employ them anonymously inside another function.
- In Python, **the lambda function** can be utilized as an argument to the higher order functions as arguments.





In [1]:

```
# Define a function using 'def'
def f(x):
    return x + 6
print(f(3.14))

# Define the same function using 'lambda'
f = lambda x: x+6
print(f(3.14))
```

```
9.14
9.14
```

In [2]:

```
# Define a function using 'def'
def f(x, y):
    return x + y

print('The sum of {} and {} is'.format(3.14, 2.718), f(3.14, 2.718))

# Define the same function using 'lambda
f = lambda x, y: x+y
print(f'The sum of pi number and euler number is {f(3.14, 2.718)}.')
```

```
The sum of 3.14 and 2.718 is 5.8580000000000005
The sum of pi number and euler number is 5.8580000000000005.
```

In [3]:

```
# Calculate the volume of a cube using def and lambda functions
# def function
def cube_volume_def(a):
    return a*a*a

print(f'The volume of a cube using def function is {cube_volume_def(3.14)}.')

# lambda function
cube_volume_lambda = lambda a: a*a*a
print(f'The volume of a cube using lambda function is {cube_volume_lambda(3.14)}.')
```

The volume of a cube using def function is 30.959144000000002.
The volume of a cube using lambda function is 30.959144000000002.

Multiplication table

In [16]:

```
def mult_table(n):
    return lambda x:x*n

n = int(input('Enter a number: '))
y = mult_table(n)

print(f'The entered number is {n}, which is a perfect number.')
for i in range(11):
    print(('%d x %d = %d' %(n, i, y(i))))
```

The entered number is 6, which is a perfect number.
6 x 0 = 0
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60

filter()

In [5]:

```
# This program returns a new list when the special numbers in the list are divided by 2 and the remainder is equal to 0
special_nums = [0.577, 1.618, 2.718, 3.14, 6, 28, 37, 1729]
new_list = list(filter(lambda x:(x%2==0), special_nums))
new_list
```

Out[5]:

[6, 28]

map()

In [6]:

```
# This program will multiplicate each element of the list with 5 and followed by power of 2.
special_nums = [0.577, 1.618, 2.718, 3.14, 6, 28, 37, 1729]
non_special_nums_1 = list(map(lambda x: x*5, special_nums))
print(non_special_nums_1)
non_special_nums_2 = list(map(lambda x: pow(x, 2), special_nums))
print(non_special_nums_2)
```

```
[2.885, 8.09, 13.59, 15.700000000000001, 30, 140, 185, 8645]
[0.332929, 2.6179240000000004, 7.387524, 9.8596, 36, 784, 1369, 2989441]
```

List comprehensions

In [7]:

```
nums = [lambda a=a:a+3.14 for a in range(5)]
for num in nums:
    print(num())
```

```
3.14
4.140000000000001
5.140000000000001
6.140000000000001
7.140000000000001
```

lambda function with if/else

In [8]:

```
age = int(input('Enter an age: '))
print(f'The entered age is {age}.')
vote = lambda age: print('Therefore, you can use a vote.') if (age>=18) else print('Therefore, you do not use a vote.')
vote(age)
```

```
The entered age is 18.
Therefore, you can use a vote.
```

lambda function usage with multiple statements

In [9]:

```
special_nums = [[0.577, 1.618, 2.718, 3.14], [6, 28, 37, 1729]]
special_nums_sorted = lambda a: (sorted(i) for i in a)

# Get the maximum of special numbers in the list
special_nums_max = lambda a, f: [y[len(y)-1] for y in f(a)]
print(f'The maximum of special numbers in each list is {special_nums_max(special_nums, special_nums_sorted)}.')

# Get the minimum of special numbers in the list
special_nums_min = lambda a, f: [y[len(y)-len(y)] for y in f(a)]
print(f'The minimum of special numbers in each list is {special_nums_min(special_nums, special_nums_sorted)}.')

# Get the second maximum of special numbers in the list
special_nums_second_max = lambda a, f: [y[len(y)-2] for y in f(a)]
print(f'The second maximum of special numbers in each list is {special_nums_second_max(special_nums, special_nu
```

```
The maximum of special numbers in each list is [3.14, 1729].
The minimum of special numbers in each list is [0.577, 6].
The second maximum of special numbers in each list is [2.718, 37].
```

Some examples

In [10]:

```
def func(n):
    return lambda x: x*n

mult_pi_number = func(3.14)
mult_euler_constant = func(0.577)

print(f'The multiplication of euler number and pi number is equal to {mult_pi_number(2.718)}.')
print(f'The multiplication of euler number and euler constant is equal to {mult_euler_constant(2.718)}.')
```

```
The multiplication of euler number and pi number is equal to 8.53452.
The multiplication of euler number and euler constant is equal to 1.5682859999999998.
```

In [20]:

```
text = 'Python is a programming language.'
print(lambda text: text)
```

```
<function <lambda> at 0x000001CF331ECEE0>
```

In [12]:

```
text = 'Python is a programming language.'
(lambda text: print(text))(text)
```

```
Python is a programming language.
```

In [13]:

```
# Multiplication of pi number and 12 in one line using lambda function
print((lambda x:x*3.14) (12))
# Division of pi number and 12 in one line using lambda function
print((lambda x: x/3.14) (12))
# Addition of pi number and 12 in one line using lambda function
print((lambda x: x+3.14) (12))
# Subtraction of pi number and 12 in one line using lambda function
print((lambda x: x-3.14) (12))
# Remainder of pi number and 12 in one line using lambda function
print((lambda x: x%3.14) (12))
# Floor division of pi number and 12 in one line using lambda function
print((lambda x: x//3.14) (12))
# Exponential of pi number and 12 in one line using lambda function
print((lambda x: x**3.14) (12))
```

```
37.68
3.821656050955414
15.14
8.86
2.5799999999999996
3.0
2446.972635086879
```

In [14]:

```
# Using the function reduce() with lambda to get the sum abd average of the list.
# You should import the library 'functools' first.
import functools
from functools import *
special_nums = [0.577, 1.618, 2.718, 3.14, 6, 28, 37, 1729]
total = reduce((lambda a, b: a+b), special_nums)
print(f'The sum and average of the numbers in the list are {total} and {total/len(special_nums)}, respectively.')
```

```
The sum and average of the numbers in the list are 1808.0529999999999 and 226.00662499999999, res
pectively.
```

In [15]:

```
1 help(functools)
```

```
Help on module functools:

NAME
    functools - functools.py - Tools for working with functions and callable objects

MODULE REFERENCE
    https://docs.python.org/3.10/library/functools.html (https://docs.python.org/3.10/library/functools.html)

    The following documentation is automatically generated from the Python
    source files.  It may be incomplete, incorrect or include features that
    are considered implementation detail and may vary between Python
    implementations.  When in doubt, consult the module reference at the
    location listed above.

CLASSES
    builtins.object
        cached_property
        partial
```