#### **Enumerate function**

Takes a collection and returns a two-item tuple which contains a count (index) and the item at that position.

#### enumerate( sequence, start=0)

Returns as an <enumerator> object which we can convert to list.





# Creating a Function

def add(x,y):
 return x+y

- Function are Objects
- 'return' statement is optional (default is None).
- Any object type may be returned.



## Parameters or Arguments?

The terms parameter and argument are used for the same thing: information that are passed into a function.

**Parameter** - is the variable listed inside the parentheses in the function definition.

**Argument** - is the value that is sent to the function when it is called.



# Default Arguments

Assign the default when defining the function:

def myFunc(x,y,z=8)

Default one, then you must default those to the right



# Passing parameters

By position: myFunc('one','two','three')

By default: myFunc('one','two')

Or by name: myFunc(x='one', y='two')

#### **Enforcing named arguments**

Used a bare \* to force a user to supply named arguments def myFunc(\*,x,y):





#### **Arbitrary positional arguments, \*args**

Allow you to pass a varying number of values during a function call, add a \* before the parameter name in the function definition.

This way the function will receive a tuple of arguments, and can access the items by 'for' statement or by t[i]



#### **Arbitrary Keyword Arguments, \*\*kwargs**

Allow you to pass multiple keyword arguments to a function. Use add a \*\* before the parameter name in the function definition. This way the function will receive a dictionary of arguments, and can access the items by using key-value pair same as dictionary





\*args /\*\*kwargs have a default ()/{}

Use \*/\*\* to unpack caller's arguments from a tuple/dictionary



# Docstrings

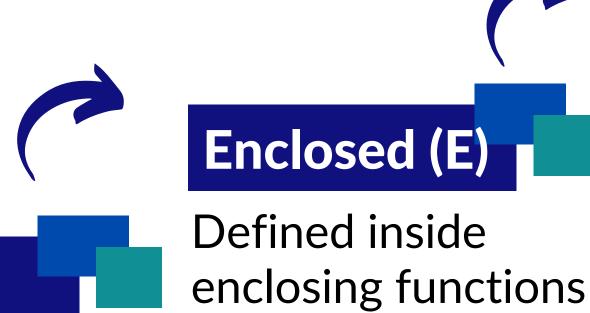
We write docstring in source code and define it immediately after module, class, function, or method definition.

It is being declared using triple single quotes (""") or triple-double quote("""").

We can access docstring using doc attribute (\_\_doc\_\_) or by help() function



# Scope and Lifetime of Variables



concept)

(Nested function

Defined at the uppermost level

Built-in (B)

Reserved names in Python builtin modules





Local (L)

Defined inside

function/class



#### **Global**

Global keyword used to declare a variable to global.



#### Nonlocal

Nonlocal keyword used to declare a variable that acts as a global variable for a nested function



### Function Attributes

As an object-oriented programming language, Python functions are objects too. It means that it has its own attributes, just like a regular object.

#### **Built-in attributes:**

Format: \_\_the\_attr\_\_, (two underlines before the name and two underlines after the name.)

func.\_\_doc\_\_

func.\_\_name\_\_







Create a file object:

myFile = open('fileName', mode,....)

#### Modes options:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists



Reading the content of the file

- f.read()
- f.read(10)
- f.readline()
- f.readlines()



Close the file when you are finish with it:

f.close()

Saved the file without closing:

f.flush()

A safer way to open files:

with open(path,mode) as f:

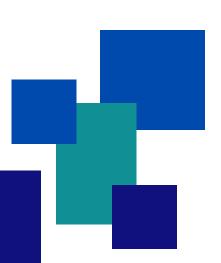
do something





The file object provides the following methods to write to a file:

- write(string)
- writelines([list])



### delete file

To delete a file, you must import the OS module:

- os.remove(fileName)
- os.path.exists(fileName)



# One liner functions

map(function, iterables)



Used to apply some functionality for every element present in the given sequence and generate a new series with a required modification.

result = map(mySumFunc, [1,2,3,4)



#### filter(function, iterables)

Used to filter value.

Returns an iterator were the items are filtered through a function to test if the item is accepted or not.

songs = filter(is\_mp4\_file, my\_files)



#### reduce(function, iterables)



Used to minimize sequence elements into a single value by applying the specified condition.

This function is defined in "functools" module.

import functools
functools.reduce(add, shopping\_list)



```
map(cook, [ 🐠, 🍆, 🐔, 🔪 ])
filter(is_vegeterian, [ , , , , , , , , , ))
=>[ , ]
reduce(eat, [ (a), (b), (b), (iii))
```

### Lambda function

A small anonymous function.

using for:

one-time usage.

arguments for functions

in list of functions

lambda arguments: expression

(lambda x, y: x + y , shopping\_list)



# List Comprehension

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

#### newlist = [expression for item in iterable if condition == True]

- **expression**: The current item in the iteration, which you can manipulate before it ends up.
- item: A variable that represents the item of the input List.
- iterable: Can be any iterable object, (like a list, tuple, set etc...
- condition: Optional. Filter conditions for the output List items.

