

# Chapter 3

- Mrs. Apurwa A. Barve

# Functions in Python

- ❖ A function is a block of code which contains a group of logically linked statements which are executed in the given sequence and serve some common business goal.
- ❖ It can accept data (arguments), process that data and can return some value after execution of the statements within it.
- ❖ It has to be defined and then called (as many times as the functionality needs to be completed) to be able to get the desired output.
- ❖ Syntax :

**def** function\_name([parameter1,parameter2,parameterN]):

→ Code statement1  
→ Code statement2  
→ Code statement N

- Starts with the keyword def
- Ends with :

- The function body consisting of one or more code statements.
- Each code statement is indented at the same level

# Activity 3.1

1. Define a function `message()` Within the method display the message passed as an argument. Take the message string from the user.

Final code should look like:

```
def message(msg):  
    print('Your message is:',msg)  
  
m = input('Enter a message which you want to print')  
message(m)
```

- Creating a function
- `msg` is the **parameter**

- Indented message body

- Invoking the function by using its exact name and passing **arguments** to it.

What is the difference between PARAMETER and ARGUMENT?

# Activity 3.1

2. Define a class Calculator. Within that class add a method add taking two numbers. Within the method perform addition of those two numbers and print the result.

Create object of Calculator class and invoke addition() on that object. Take the two numbers from the user.  
RETURN the result of addition

3. Look at the code below and discuss the code changes which you see.

```
class Calculator:
    def __init__(self):
        pass
    def addition(self,num1,num2):
        res = num1+num2
        print(f'The addition of {num1} and {num2} is {res}')
    def addition(self,*num):
        res=0
        for n in num:
            res=res+n
        print('Addition of all the numbers is',res)

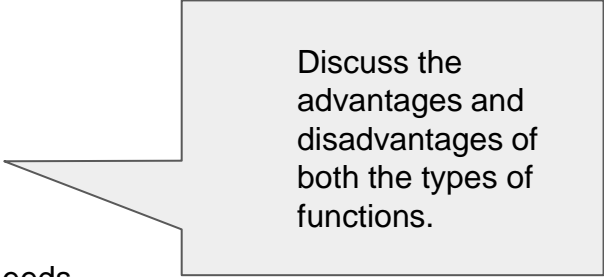
calObj = Calculator()
calObj.addition(2,3)
calObj.addition(1,2,3,4)
```

addition() taking two parameters

addition() taking 'n' number of parameters using \*

# Functions in Python

- ❖ Advantages of functions : Code reuse, code modularity and code readability
- ❖ Types of functions :
  - **In built functions** : These are Standard functions in Python that are available to use.
  - **User defined functions** : User added functions based on business needs



Discuss the advantages and disadvantages of both the types of functions.

# Functions in Python

## ❖ Arbitrary Arguments :

- In the above example, it is possible to pass 'n' number of arguments to the addition() by using var ARBITRARY arguments syntax, **\*varNm**
- These arbitrary arguments are handled as a TUPLE and accessed using the index of each individual element.

## ❖ Default Parameter Value : If we call the function without argument, it uses the default value.

```
def no_of_present_students(count=0):  
    print('Total number of students present for the lecture is:',count)  
  
no_of_present_students(120)  
no_of_present_students(175)  
no_of_present_students()
```

What will be the output of the above code? And why?

# Lambda Functions in Python

- ❖ An anonymous function in Python is a function without a name. Such function is called LAMBDA function
- ❖ A lambda function can take any number of arguments, but can only have one expression.
- ❖ They are used when an anonymous function is required for a short period of time.
- ❖ Check the modified example of calculator class to understand the use of lambda functions

## Activity 3.2

Step 1 : Copy the below given part of code

Create a new python file. Call it LambdaCalculator.py

Getting numbers and operation from the user

main.py

LambdaCalculator.py

+

```
1 val1 = int(input('Enter first number'))
2 val2 = int(input('Enter second number'))
3 op = input("Enter add if you want to add the two numbers or sub if you want to subtract")
4
```



## Activity 3.2

Step 2: Copy the below given part of code

```
5 #defining lambda function and its handler for addition
6 addFunction = lambda val1,val2:val1+val2
7
8 #defining lambda function and its handler for addition
9 ▾ if(val1>val2):
10     subFunction = lambda val1,val2:val1-val2
11 ▾ else:
12     subFunction = lambda val1,val2:val2-val1
```

Syn :  
**lambda** argument(s):expression

**addFunction** and **subFunction** are the **HANDLERS** which are used to invoke specific lambda function linked to them

A given lambda function can have any number of arguments but **ONLY ONE** expression

## Activity 3.2

Step 3: Copy the below given part of code

```
14 #invoking appropriate function based on user's choice of operation
15 ▾ if (op=='add'or op=='ADD'or op=='Add'):
16     print("Result of addition is",addFunction(val1,val2))
17 ▾ else:
18     print("Your choice can not be matched")
19 ▾ if (op=='sub'or op=='SUB'or op=='Sub'):
20     print("Result of subtraction is",subFunction(val1,val2))
21 ▾ else:
22     print("Your choice can not be matched")
```

If the operation is Add we call add lambda function HANDLER, addFunction and pass the two values to that handler.

Else we call subFunction which is the handler to subtraction lambda function.

# Lambda Functions v/s user defined functions

<b><u>Lambda Functions</u></b>	<b><u>User Defined Functions</u></b>
Defined using the lambda keyword	Defined using the def keyword
Typically single lined	Typically multilined
No return statement required to return the value	Return statement required to return any value
These are ANONYMOUS functions	These are NAMED functions
They can be involved immediately without being defined beforehand	They can be invoked only when they are defined somewhere.

# Modules

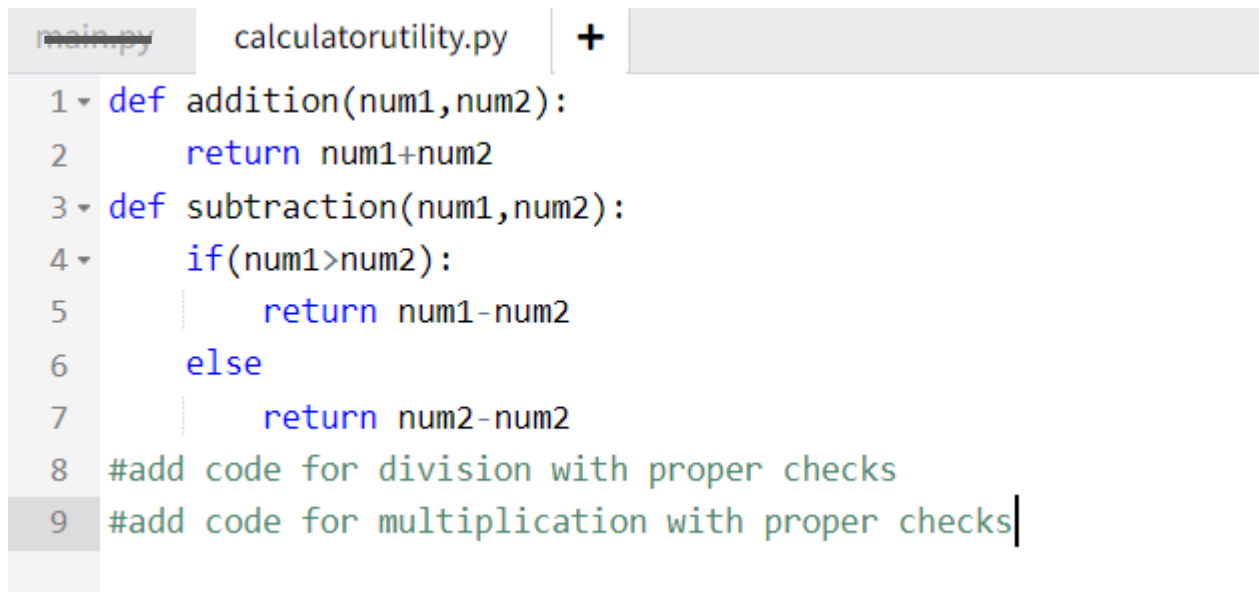
- ❖ A MODULE is a file containing other Python constructs like class, functions, variables, statements which together perform some specific task(s)
- ❖ Each module needs to have some name and the file should be stored with .py extension.
- ❖ A module acts as a CODE LIBRARY.
- ❖ There are over 200 Standard Library Modules in Python
- ❖ We can create our own USER DEFINED modules and use those in other Python programs to achieve that functionality.
- ❖ Having modules means separating a bigger, lengthy code into manageable files each containing some independent part of that code.
- ❖ Modules promote modularization of code, code reuse, code maintainability and better organisation.

## Activity 3.3

1. Define a MODULE called calculatorutility.py

To this module add methods such as addition(),subtraction(),division() and multiplication()

Use the below given code snippet to start and build your module further:



```
main.py calculatorutility.py +
1 def addition(num1,num2):
2     return num1+num2
3 def subtraction(num1,num2):
4     if(num1>num2):
5         return num1-num2
6     else
7         return num2-num2
8 #add code for division with proper checks
9 #add code for multiplication with proper checks|
```

## Activity 3.3

2. To USE the calculatorutility module, we need to import it into another file first and then invoke its methods.

Use the below given code snippet to start and build your module further:

```
main.py  calculatorutility.py  MyCal.py  +  
1  import calculatorutility  
2  
3  n1 = int(input('Enter one number'))  
4  #add code to take another number from the user in n2 variable  
5  
6  #invoking addition() from calculatorutility module  
7  res= calculatorutility.addition(n1,n2)  
8  print('Result of addition is:',res)  
9  
10 #invoking subtraction() from calculatorutility module  
11 res= calculatorutility.subtraction(n1,n2)  
12 print('Result of subtraction is:',res)  
13  
14 # add code to invoke multiplication() from calculatorutility module  
15 # add code to invokde division() from calculatorutility module  
16
```

Using keyword  
import to add  
another module(s)  
in the current file.

Using  
modulename.methodname()  
syntax to invoke methods from  
the module

Complete the pending  
code and execute your  
example.

# Variables in Modules

- ❖ A MODULE can contain functions, classes and even VARIABLES. Check the example in the next slide to understand how to use these variables from a module into another file.

## Activity 3.4

1. Define a module `iplteams.py` to contain details of 2 teams, CSK and MumbaiIndians. Use the code snippet given below to start and build your code further.

```
main.py  iplteams.py  checkIPLDetails.py  +  
1  #details of CSK team  
2  CSK = {  
3      #observe how attributes and values are paired and comma separated within the variable CSK  
4      "captain": "Ruturaj Gaikwad",  
5      "state": "Chennai",  
6      "rank": 1  
7  }  
8  #add details for MumbaiIndian team
```



## Activity 3.4

2. Now create another file which uses iplteams.py module. Use the code snippet below to start and build your code further

```
main.py  iplteams.py  checkIPLDetails.py  +
1  import iplteams as ipt
2
3  print('Details of CSK team are as given below:')
4  print(ipt.CSK['captain'])
5  print(ipt.CSK['state'])
6  print(ipt.CSK['rank'])
7
8  #add code for printing details of mumbai indians
```

Using another file to import iplteams module

Giving iplteams module an **ALIAS**, **ipt**, in the import statement using the keyword **as**

Using the ALIAS, **ipt**, to access CSK variable and its key:value data

Complete the missing code and execute the example. Observe the output

## Activity 3.4

3. Copy the code given below and observe its output. Discuss the **dir() function** and its use.

```
print(dir(ipt))
```

```
print(dir(calculatorutility))
```

# Packages

- ❖ One can think of a folder while thinking of the concept of a PACKAGE. Like a folder, a package is way we can pack modules and other components into one wrapper.
- ❖ They boost code reuse and maintainability.
- ❖ A package is a namespace where that package's modules are bound together by the package name, by which they may be referenced in other files.
- ❖ Every package is a module — but not every module is a package.
- ❖ A package folder usually contains one file named `__init__.py` that tells Python that the given folder/namespace is a package. The init file may be empty, or it may contain code to be executed upon package initialization.
- ❖ Python comes with a big collection of pre-installed packages known as the Python Standard Library

# Fast API - Self Study

Thank you