எங்கள் வாழ்வும் எங்கள் வளமும்
மங்காத தமிழ் என்று சங்கே முழங்கு ... *புரட்சிக்கவி*

NOTICE

- We support open-source products to spread Technology to the mass.
- This is completely a FREE training course to provide introduction to Python language
- All materials / contents / images/ examples and logo used in this document are owned by the respective companies / websites. We use those contents for FREE teaching purposes only.

- We take utmost care to provide credits whenever we use materials from external source/s. If we missed to acknowledge any content that we had used here, please feel free to inform us at info@DataScienceInTamil.com.
- All the programing examples in this document are for FREE teaching purposes only.

Thanks to all the open-source community and to the below websites from where we take references / content /code example. definitions, please use these websites for further reading:

- Python Notes For Professionals.pdf – this is the book we follow
- https://docs.python.org
- https://www.datacamp.com/community/tutorials/functions-python-tutorial
- https://www.w3schools.com
- https://data-flair.training/blogs/python-function/

Today's class

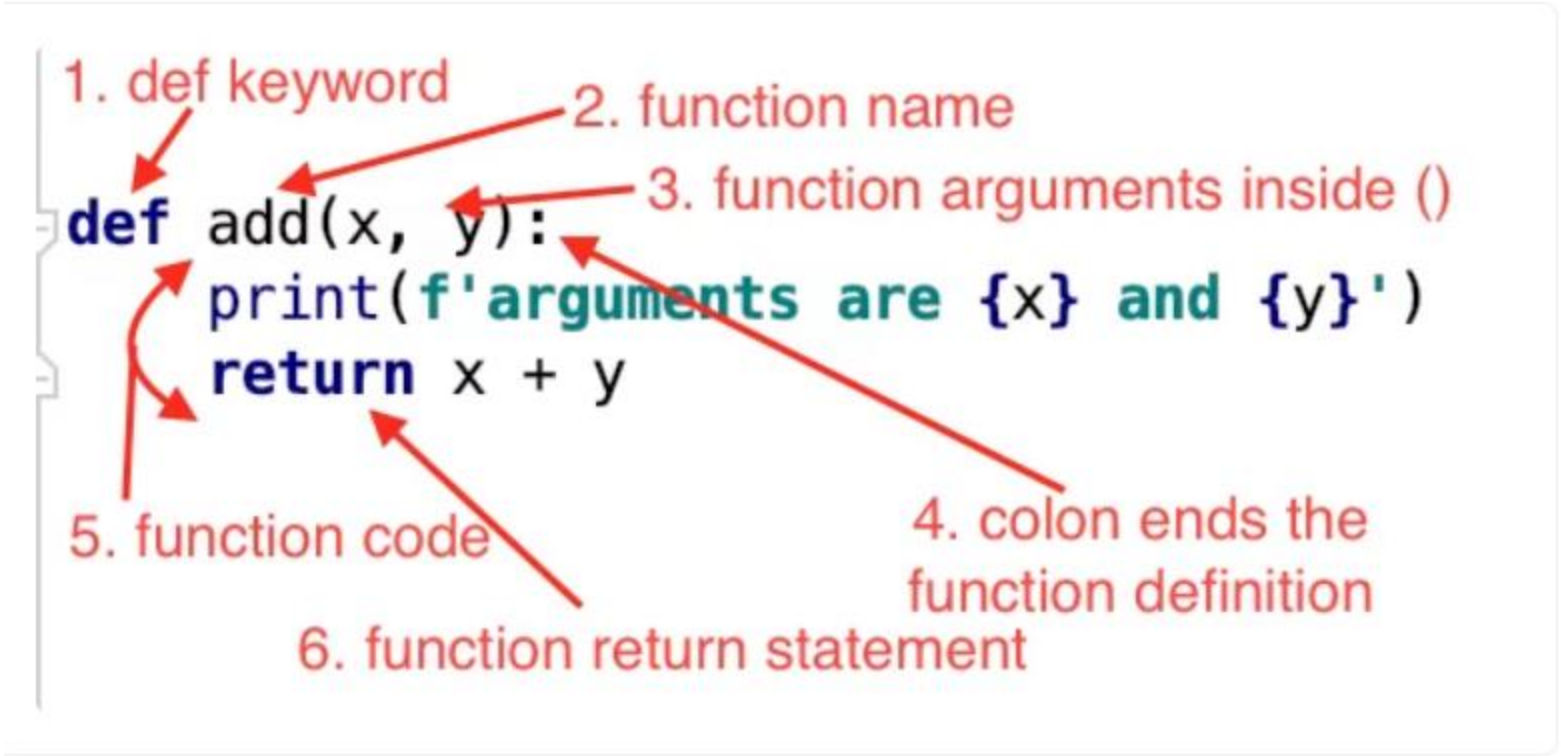
Data Science in Tamil

007 TOPIC: PYTHON FUNCTIONS

- What is function
- what is function signature in python?
- Rules for naming python function (identifier)
- The pass Statement in a function
- **Types of Functions in Python**
    1. Built Functions
    2. User defined functions
    3. Python Function with no argument and no return value.

4. Function with no argument and with a Return value.
5. Python Function with argument and No Return value.
6. Function with argument and return value.
7. Returning Multiple Values in a function
8. Function with default arguments. **- Default arguments are optional arguments**
9. The pass Statement
10. Function with arbitrary positional arguments.
11. Function with arbitrary keyword arguments.
12. High Order Functions
13. Anonyms Functions / Lamda Functions
14. Recursion functions

What is function



1. A function is a block of code which only runs when it is called.
2. You can pass data, known as parameters, into a function.
3. A function can return data as a result.

Python function in any programming language is a sequence of statements in a certain order, given a name. When called, those statements are executed. So we

don’t have to write the code again and again for each [type of] data that we want to apply it to. This is called code re-usability

function is a piece of code written to carry out a specified task. To carry out that specific task, the function might or might not need multiple inputs.

- A function is a block of code with a name.
- We can call a function by its name.
- The code inside a function only runs when it’s called.
- A function can accept data from the caller program, it’s called as function parameters.
- The function parameters are inside parentheses and separated by a comma. A function can accept any number of arguments.
- A function can return data to the caller program. Unlike other popular programming languages, Python functions definition doesn’t specify the return type.
- We can’t use reserved keywords as the function name. A function name must follow the Python identifiers definition rules.

Rules to follow to naming python function

1. Same rules of declaring variables
2. It can begin with either of the following: A-Z, a-z, and underscore(_).
3. The rest of it can contain either of the following: A-Z, a-z, digits(0-9), and underscore(_).
4. A reserved keyword may not be chosen as an identifier.

What is function signature in python?

1. parameters and their types
2. a return value and type, return can returns MULTIPLE VALUES

Function with no argument and with a Return value.

Defining and call a function

```
def my_function():
  print("Say Hello from a function")

result = my_function()
```

Python Function with argument and No Return value.

- Information can be passed into functions as arguments.
- Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
- A parameter is the variable listed inside the parentheses in the function definition.
- An argument is the value that is sent to the function when it is called.

```
def my_function(mySubject):
  print("I am studying ", mySubject)

result = my_function("Python")
result = my_function("Numpy")
result = my_function("Pandas")
```

Function with argument and return value.

```
def add_Fucntion(a, b):
    result = a + b
    return result

add_Fucntion(2, 2)
```

Returning Multiple Values

```
def add_Fucntion(a, b):
    addResult = a + b
    subResult = a * b
    multiResult = a * b

    return addResult,subResult,multiResult

result = add_Fucntion(10,2)
print(result)
---------------
```

Built Functions

Built-in functions, such as help() to ask for help, min() to get the minimum value, print() to print an object to the terminal

The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.

Built-in Functions				
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

https://docs.python.org/3/library/functions.html

https://www.w3schools.com/python/ref_func_abs.asp

Function	Description
abs()	Returns the absolute value of a number
all()	Returns True if all items in an iterable object are true
any()	Returns True if any item in an iterable object is true
ascii()	Returns a readable version of an object. Replaces none-ascii characters with escape character
bin()	Returns the binary version of a number
bool()	Returns the boolean value of the specified object
bytearray()	Returns an array of bytes
bytes()	Returns a bytes object
callable()	Returns True if the specified object is callable, otherwise False
chr()	Returns a character from the specified Unicode code.

classmethod()	Converts a method into a class method
compile()	Returns the specified source as an object, ready to be executed
complex()	Returns a complex number
delattr()	Deletes the specified attribute (property or method) from the specified object
dict()	Returns a dictionary (Array)
dir()	Returns a list of the specified object's properties and methods
divmod()	Returns the quotient and the remainder when argument1 is divided by argument2
enumerate()	Takes a collection (e.g. a tuple) and returns it as an enumerate object
eval()	Evaluates and executes an expression
exec()	Executes the specified code (or object)
filter()	Use a filter function to exclude items in an iterable object
float()	Returns a floating point number

format()	Formats a specified value
frozenset()	Returns a frozenset object
getattr()	Returns the value of the specified attribute (property or method)
globals()	Returns the current global symbol table as a dictionary
hasattr()	Returns True if the specified object has the specified attribute (property/method)
hash()	Returns the hash value of a specified object
help()	Executes the built-in help system
hex()	Converts a number into a hexadecimal value
id()	Returns the id of an object
input()	Allowing user input
int()	Returns an integer number
isinstance()	Returns True if a specified object is an instance of a specified object
issubclass()	Returns True if a specified class is a subclass of a specified object

iter()	Returns an iterator object
len()	Returns the length of an object
list()	Returns a list
locals()	Returns an updated dictionary of the current local symbol table
map()	Returns the specified iterator with the specified function applied to each item
max()	Returns the largest item in an iterable
memoryview()	Returns a memory view object
min()	Returns the smallest item in an iterable
next()	Returns the next item in an iterable
object()	Returns a new object
oct()	Converts a number into an octal
open()	Opens a file and returns a file object
ord()	Convert an integer representing the Unicode of the specified character

pow()	Returns the value of x to the power of y
print()	Prints to the standard output device
property()	Gets, sets, deletes a property
range()	Returns a sequence of numbers, starting from 0 and increments by 1 (by default)
repr()	Returns a readable version of an object
reversed()	Returns a reversed iterator
round()	Rounds a numbers
set()	Returns a new set object
setattr()	Sets an attribute (property/method) of an object
slice()	Returns a slice object
sorted()	Returns a sorted list
staticmethod()	Converts a method into a static method
str()	Returns a string object

sum()	Sums the items of an iterator
super()	Returns an object that represents the parent class
tuple()	Returns a tuple
type()	Returns the type of an object
vars()	Returns the __dict__ property of an object
zip()	Returns an iterator, from two or more iterators

TASK TO DSIT - TECH MEMBERS

- Create code for each builtin functions
- Task is assigned to
  - Names here

User defined functions

Python lets us group a sequence of statements into a single entity, called a function. A Python function may or may not have a name.

1. This Python Function help divide a program into modules. This makes the code easier to manage, debug, and scale.
2. It implements code reuse. Every time you need to execute a sequence of statements, all you need to do is to call the function.
3. This Python Function allow us to change functionality easily, and different programmers can work on different functions.

```
def add_Fucntion(a, b):
    addResult = a + b
    subResult = a * b
    multiResult = a * b

    return addResult,subResult,multiResult

result = add_Fucntion(10,2)
print(result)
```
