Dt-28/02/2023

FUNCTIONS:

Function is a paradigm

Types of Paradigm:	**Wasting Resources:**
Procedural Paradigm	Programming Time
Functional Paradigm	Programmer Time
Modular Paradigm	Interpreter
Oops	Memory
Logical Paradigm	Enhancement is difficult
	Debugging is difficult

Functional Programming:

- No wastage of resources
- Enhancement is easy
- Debugging is easy

Types of Function:

1. Built-in Functions
   These functions are pre-defined by python.
   [print(),len(),input()…..]

2. User Defined Functions
   - These functions are defined and used by the programmer as per project needs.
3. Recursive Function

4. Lambda Function

Functions:

- A function is a repeated block of code which consist of business logic.
- This block can be called “n” number of times based on the requirement.

Functions consists of two parts,

- Function Definition
  - Function Declaration
    It is first line of the definition with “def” keyword.
  - Function Implementation
    Block of code under function definition is called Function Implementation.

Syntax:

Def<Function Name>():

- ✓ Function definition should start with “def” keyword followed by function name which identifies the function definition uniquely.
- ✓ Function definition can exist without calling.
- ✓ Function definition should be defined before function calling.
- ✓ We can define multiple function definition within the same program.
- ✓ Function definition is executed when it is called.

- Function Calling

Syntax:

<Function Name>()

- ✓ Function Name followed by paranthesis is considered as function calling.
- ✓ Function calling should be always mentioned after definition of the function.
- ✓ Function calling can be done ‘n’ number of time.
- ✓ Function calling cannot exsit without function definition, It triggers error.

Types of Variables:

❖ **Local Variables:**

- Variables which are declared within a function and can be used within the same function only.
- These variable cannot be used outside the declared function.

❖ **Global Variables:**

- Global Variables which can used anywhere within the program is called Global Variable.
- Global variables are declared in two ways,

- Declaring a variable outside all the functions is called or considered as global variable
- A variable which declared with global keyword inside a function is also considered global variable.

Types of Arguments:

- Positional Arguments
  - Values are assigned to argument at function definition based on position is called Positional Arguments.
- Keyword Arguments
  - Initializing arguments with relevant values at function calling is called Keyword Argument.
  - Keyword Argument should always follow positional argument.
- Default Arguments
  - Initializing argumentswith alternative values at function definition is called default arguments.
  - KA should always follow last argument to declared.

Variable-Length Arguments

Keyword Variable Length Argument

Dt-28/02/2023

- Variable-Length Arguments:
  - It is a special type of variable which accept 'n' number of positional values.
  - These values are stored in the form of 'Tuple'.

  Recommended: Variable length argument name should be 'args'.
  *args → Variable length argument

- Keyword Variable Length Argument:
  - It is a special type of variable which accept 'n' number of keyword arguments.
  - These keyqord arguments are stored in the form of dictionary.

  Recommended: Keyword variable argument name should be 'kwargs'.
  **kwargs→ Keyword length argument

Return:

- A function definition can return a value/values to the function calling part using "return" keyword.
- These values can be collect at function calling place.
- Return should be the last line of the function.

Function Reference:

- Function reference refers to address of the function.
- Which ever variable holds the address of the function is authorized to call the function.
- This address can be passed from one variable to another variable.

Nested Functions:

- Nested function can be executed within the parent function.
- A Function within another function is called Nested Function.
- These nested function cannot be executed

Closure:

- Passing address of nested function to main program to main program and executing nested function to main program is called as Closure Property.

- Function Reference, Nested Function, Closure Property → Decorators

Func→ Name recommended name variable which holds function reference.

Df→ DataFrame

Tf→ Tensorflow

Dt-01/03/2023

LAMBDA FUNCTION(ANONYMOUS FUNCTION):

Recursion Function:

- A function which calls itself until a particular condition is satisfied.
- Condition which control the flow of execution is called base condition.

LAMBDA FUNCTION(ANONYMOUS FUNCTION):

- Lambda function is called anonymous function.
- Anonymous means not identify for this function.
- It is an anonymous inline function which is defined with “lambda” keyword and this function accepts “n” number of arguments but return only one value based on the expression.

Properties of Lambda Function:

- It doesn’t have any name to identify.
- No “def” keyword is used to define it.
- It accepts “n” number of arguments.
- It return only one value without any “return” keyword.
- It is an inline function.
- It is considered as light weight function.
- Its execution is very fast compared to that of traditional function.
- It is single use function(no resuability).

Use Of Lambda Function:

- We can invoke the lambda into another python objects.
  - List, Dictionray…etc
- It can act as source of input to higher order functions.
  - Higher order Function: Any function which accepts other function reference as argument is called higher order function.
  - Map, Filter, Reduce

Syntax of Lambda Function:

❖ Lambda <Input Arguments> : <Output Expression>

- Input Argument(Optional)→ Values passed into function
- Output Expression→ Bussiness logic for output

Higher order Function:

- Higher order Function: Any function which accepts other function reference as argument is called higher order function.
- Map
    - It maps elements of iterable with function to implement logic.
    - Map(func , <iterable>)
    - Map() implements a business logic to each and every element of the iterable.
    - No. of input elements = No. output elements
- Filter,
- Reduce

Dt-02/03/2023

- Filter
    - It implements conditions each and every elements of iterable and return only satisfied elements.
    - Syntax:
        - filter(func,<Iterable>)
        - filter(lambda(),<Iterable>)
    - No. of output element either equal or less than the input number based on the condition.

- Reduce
    - It is used to reduce the final output to a single output.
    - It is a part of “functools” library