UNIT 4

1. Define function in python. List its types.

In Python, a function is a named block of code that performs a specific task. It helps in organizing code and making it reusable.
Functions can have parameters, which are variables that we can pass values into when we call the function.
Functions can also return values using the return statement.

There are different types of functions in Python:

1. Built-in Functions: These are functions that are already available in Python. Examples include print(), len(), and range().

2. User-defined Functions: These are functions that we create ourselves. We can define our own functions using the def keyword and give them a name. We can then call these functions whenever we need to perform a specific task.

3. Lambda Functions: The lambda function, which is also called anonymous function. A lambda function can take any number of arguments, but can only have one expression.

2. Explain built in function in detail with example.(type/data conversion,math, string etc

Built-in functions in Python are pre-defined functions that are available for use without the need for any additional imports or installations. These functions serve different purposes and are grouped into various categories. Here are a few examples:

1. Type Conversion Functions:These functions are used to convert values from one data type to another.
   - int(): Converts a value to an integer.
   - float(): Converts a value to a floating-point number.
   - str(): Converts a value to a string.
   - bool(): Converts a value to a boolean.

   Example:
```
num = "10"
num_int = int(num)
print(num_int)  # Output: 10
```

2. Mathematical Functions:These functions provide various mathematical operations.
   - abs(): Returns the absolute value of a number.
   - pow(): Raises a number to a specific power.
   - round(): Rounds a number to a specified precision.

```
  Example:
  x = -3.7
  abs_x = abs(x)
  print(abs_x)  # Output: 3.7
```

3. String Functions: These functions are used to manipulate strings
  - len(): Returns the length of a string.
  - upper(): Converts a string to uppercase.
  - lower(): Converts a string to lowercase.
  - split(): Splits a string into a list of substrings.

```
  Example:
  message = "Hello, World!"
  length = len(message)
  print(length)  # Output: 13
```

3. List Functions: These functions are used to manipulate lists.

```
 Example:
 my_list = [1, 2, 3]
 my_list.append(4)
```

3. What is user defined function? How to declare, define & call function in python with syntax & example.

In Python, def keyword is used to declare user defined functions.
The function name with parentheses (), which may or maynot include parameters and arguments and a colon:
An indented block of statements follows the function name and arguments which contains the body of the function.
Syntax: def function_name(): statements
Example:

```
def fun():
print("User defined function")fun()
output: User defined function
```

Parameterized function: The function may take arguments(s) also called parameters as input within the opening and closing parentheses, just after the function name followed by a colon.
Syntax:
def function_name(argument1, argument2, ...): statements
Example:

```
def square( x ): print("Square=",x*x)
Output: Square= 4
```

4. Explain function argument and parameter passing with syntax and example.

Functions can accept parameters, which are placeholders for the values that the function will receive when it is called. These parameters are defined in the function's definition. When you call the function and pass actual values to these parameters, they are called arguments

Parameters: Parameters are the variables listed inside the parentheses in the function definition.

SYNTAX : def function_name(parameter1, parameter2, ...):

Example

```
def greet(name):
 print("Hello,", name)
```

Arguments: Arguments are the actual values that are passed to the function when it is called. They are provided inside the parentheses of the function call.

Syntax : function_name(argument1, argument2, ...)

Example:

```
greet("Alice")
```

5. Explain scope of variable with example.

The scope of a variable refers to the region of the code where the variable is accessible.

Global Scope: Variables defined outside of any function or class have a global scope. They can be accessed from anywhere in the code, both inside and outside functions.

```
global_var = 10
def func():
 print(global_var)
func()
print(global_var) # Output: 10
```

Local Scope: Variables defined inside a function have a local scope. They are accessible only within that function.

```
def func():
 local_var = 20
 print(local_var)
func()
```

6. Explain function with return statement with syntax and example.

A return statement is used within a function to send a result or value back to the code that called the function. The return statement ends the function's execution immediately and passes control back to the caller along with the specified value.

Syntax : def function_name(parameters): return value_to_return

Example:

```
def add_numbers(a, b):
 sum = a + b
 return sum
```

```
result = add_numbers(3, 5)
print(result)
```

7. What is module? How does module defined in python?

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference. Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.
Example
The Python code for a module named aname normally resides in a file named aname.py.
Here's an example of a simple module, support.py

```
def print_func( par ):
 print "Hello : ", par
 return
```

To create a module just save the code you want in a file with the file extension .py:
Example
Save this code in a file named mymodule.py

```
def greeting(name):
 print("Hello, " + name)
```

Now we can use the module we just created, by using the import statement:
Example
Import the module named mymodule, and call the greeting function:

```
import mymodule
mymodule.greeting("ABC")
```

8. Explain in detail how module & object from module can be import with example.

Importing modules and objects from modules in Python allows you to access functionality defined in other Python files. You can import entire modules, specific objects (functions, classes, variables), or even rename them during import.
Importing Entire Modules:
You can import an entire module using the import statement. This allows you to access all the functions, classes, and variables defined in that module.
Example:

```
import math
print(math.sqrt(16))
print(math.pi)
```

Importing Specific Objects from Modules:
Instead of importing the entire module, you can import specific objects (functions, classes, variables) from the module using the from ... import ... syntax.

```
from math import sqrt, pi
 print(sqrt(25))
print(pi)
```

9. Explain built in modules (numeric,mathematical,functional programming module) with example.

1.Numeric Module (numbers)
The numbers module provides a hierarchy of numeric abstract base classes which are used to define numeric types. It includes abstract base classes such as Number, Complex, Real, Integral, etc.
EXAMPLE

```
import numbers
print(isinstance(5, numbers.Number))
print(isinstance(3.14, numbers.Number))
print(isinstance(2 + 3j, numbers.Number))
```

2. Mathematical Module (math)
The math module provides mathematical functions for tasks such as trigonometry, logarithms, exponentiation, and more.
Example import math

```
print(math.sqrt(16)) # Output: 4.0
print(math.factorial(5)) # Output: 120
```

3. Functional Programming Module (functools)
The functools module provides higher-order functions and operations for working with functions and callable objects.
Example (using functools.partial to create a new function with default arguments):

```
import functools
def greet(greeting, name):
 print(f"{greeting}, {name}!")
greet_hello = functools.partial(greet, "Hello")
greet_hello("Alice")
```

9. Explain the concept of namespace & scoping in detail with example.

Namespaces are containers that hold a collection of identifiers (such as variable names, function names, class names, etc.) and their corresponding objects. Namespaces are used to avoid naming conflicts and to provide a way to organize and manage identifiers within a Python program A namespace is a mapping from names to objects. Each scope in Python has its own namespace, which stores the names defined within that scope
EXAMPLE:

```
x = 10 # Global variable
def my_function():
 y = 20
 print(x)
 print(y)
my_function()
```

(Scope in above ans)

11. How python package can be created. Give steps to write package with syntax & example.

Step 1: Create Package Directory Structure
Step 2: Define __init__.py
Step 3: Add Modules
Step 4: Use the Package
Syntax:

```
my_package/
 __init__.py
 module1.py
 module2.py
```

Example:

```
import my_package
my_package.module1.greet("Alice")
my_package.module2.add(3, 5)
from my_package.module1 import goodbye
goodbye("Bob") !
```

12. Explain in detail standard packages with its uses.(math, scipy,numpy,matplotlib, pandas)

Standard Python packages are pre-built libraries that come included with Python installations.
1. Math (math): The math module provides mathematical functions and constants for numerical computations.
Uses:
• Performing basic mathematical operations like square root, exponentiation, trigonometric functions, etc.
• Constants like pi and e for mathematical calculations.

2 Scientific Computing (scipy): The scipy library builds on top of numpy and provides additional scientific computing tools and algorithms.
Uses:
• Integration, differentiation, and optimization.
• Signal processing, interpolation, and filtering.
• Statistical functions, probability distributions, and hypothesis testing.

3.Numerical Computing (numpy): The numpy library provides support for numerical operations on multi-dimensional arrays and matrices.
Uses:
• Array manipulation and mathematical operations on arrays.
• Linear algebra, Fourier transforms, and random number generation.
• Efficient computation and manipulation of

4.Data Visualization (matplotlib): The matplotlib library provides tools for creating static, animated, and interactive visualizations in Python.
Uses:
• Plotting 2D and 3D graphs, histograms, and scatter plots.
• Customizing plots with labels, titles, legends, and annotations.
• Creating publication-quality figures for scientific publications and presentations.

5.Data Analysis and Manipulation (pandas): The pandas library provides high-level data structures and tools for data manipulation and analysis.
Uses:
• Loading and manipulating structured data from various sources like CSV files, databases, and Excel sheets.
• Data cleaning, transformation, and exploration.
• Handling missing data, merging and joining datasets, and performing statistical analysis.

13. Explain user defined packages with example & syntax.

User-defined packages are custom packages created by users to organize and distribute their Python code across different modules. Creating a user-defined package involves organizing related modules into a directory structure and adding special files to define the package and its contents
Syntax:
```
my_package/
 __init__.py
 module1.py
 module2.py
```
Example:
```
import my_package
my_package.module1.greet("Alice")
my_package.module2.add(3, 5)
from my_package.module1 import goodbye
goodbye("Bob") !
```

14. Use of any four methods in math module

1. math.sqrt(x):This method returns the square root of a given number x.
```
import math
result = math.sqrt(16)
print("Square root of 16:", result)
```

2. math.pow(x, y):This method returns x raised to the power of y.
```
import math
result = math.pow(2, 3)
print("2 raised to the power of 3:", result)
```

3. math.sin(x):This method returns the sine of a given angle x (in radians).

```
import math
angle = math.radians(90)
result = math.sin(angle)
print("Sine of 90 degrees:", result)
```

4.math.floor(x):This method returns the largest integer less than or equal to a given number x.

```
import math
result = math.floor(3.7)
print("Floor of 3.7:", result)
```

15. Use of NumPy.

NumPy (Numerical Python) is a powerful library in Python used for numerical computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

16. Describe module in Python with its advantages?

A module is a file containing Python definitions, statements, and functions. It serves as a reusable unit of code that can be imported and used in other Python scripts or modules. Modules allow you to organize related functionality into separate files, making your code more modular, maintainable, and easier to manage.

Advantages:

1.Encapsulation: Modules provide a way to encapsulate related code into separate units, allowing you to group similar functionality together.

2. Reusability: you can write code once and reuse it in multiple places within the same project or across different projects

3.Namespace Separation: Each module has its own namespace, which acts as a container for the names defined within the module.

4.Modularity: Modules allow you to break down your code into smaller, more manageable units, making it easier to understand and maintain.