ASSIGNMENT 6

1.What is functions? State its type and explain the syntax to define a Function.

In Python, a function is a reusable block of code that performs a specific task. Functions help organize code, promote code reusability, and make programs easier to understand and maintain.

Types of Functions

1. **Built-in Functions**: Functions that are provided by Python (e.g., print(), len(), sum()).
2. **User-defined Functions**: Functions created by the user to perform specific tasks.
3. **Lambda Functions**: Anonymous functions defined using the lambda keyword for simple operations, often used for short-term use.
4. **Higher-order Functions**: Functions that take other functions as arguments or return them as results (e.g., map(), filter(), reduce()).

Syntax to Define a Function

The basic syntax to define a function in Python is as follows:

Code:-

```
def function_name(parameters):
    """Docstring (optional): Description of the function."""
    # Function body: code to execute
    return value  # Optional return statement
```

Breakdown of the Syntax

- **def**: A keyword that indicates the start of a function definition.
- **function_name**: The name of the function. It should be descriptive and follow naming conventions (snake_case).
- **parameters**: (Optional) A comma-separated list of parameters (inputs) the function can accept. If there are no parameters, you can leave the parentheses empty.
- **Docstring**: A string (enclosed in triple quotes) that describes what the function does. It's optional but recommended for documentation.
- **Function Body**: The block of code that defines what the function does. This code runs when the function is called.
- **return**: (Optional) A statement that exits the function and can return a value to the caller. If omitted, the function returns None.

Example

Here's a simple example of a user-defined function:

Code:-

```
def add_numbers(a, b):
    """Returns the sum of two numbers."""
return a + b
# Calling the function
result = add_numbers(3, 5)
print(result)
```

2.What is lambda function?

A **lambda function** in Python is a small, anonymous function defined using the lambda keyword. It can take any number of arguments but can only have a single expression. Lambda functions are often used for short-term operations and are especially useful in situations where you need a simple function for a brief period, such as with functions like map(), filter(), and sorted().

Characteristics of Lambda Functions

- **Anonymous**: They do not need a name (hence "anonymous").
- **Single Expression**: Lambda functions can only contain one expression; they cannot have multiple statements or a return statement.
- **Return Value**: The result of the expression is automatically returned.

Syntax

The syntax for defining a lambda function is as follows:

Code:-

```
lambda arguments: expression
```

Example

Here are a few examples to illustrate how lambda functions work:

1.Basic Usage:

Code:-

```
add = lambda x, y: x + y
print(add(3, 5))  # Output: 8
```

2.Using with map():

Code:-

```
numbers = [1, 2, 3, 4]
squares = list(map(lambda x: x ** 2, numbers))
print(squares)  # Output: [1, 4, 9, 16]
```

3.Using with filter():

Code:-

```
numbers = [1, 2, 3, 4, 5, 6]

even_numbers = list(filter(lambda x: x % 2 == 0, numbers))

print(even_numbers)  # Output: [2, 4, 6]
```

4.Using with sorted():

Code:-

```
tuples = [(1, 'one'), (2, 'two'), (3, 'three')]

sorted_tuples = sorted(tuples, key=lambda x: x[1])

print(sorted_tuples)  # Output: [(1, 'one'), (3, 'three'), (2, 'two')]
```

When to Use Lambda Functions

- When you need a simple function for a short duration.
- In cases where defining a full function is unnecessary, like in functional programming constructs (e.g., map(), filter(), sorted()).
- For simple operations that can be expressed in a single line.