Loops and Functions

Loops in Python are used to execute a block of code repeatedly based on a condition or iterating over a sequence. There are mainly two types of loops in Python: **`for`** loops and **`while`** loops.

1. For Loop

The `for` loop is used to iterate over items in a sequence (like a list, tuple, string, or range).

Syntax:

```
for variable in sequence:
    # code to execute for each item
```

Example:

```
# Looping through a list
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

Using `range()`:

```
# Looping with range
for i in range(5):  # Goes from 0 to 4
    print(i)
```

2. While Loop

The `while` loop executes as long as a given condition is true.

Syntax:

```
while condition:
    # code to execute while condition is true
```

Example:

```
# Printing numbers from 0 to 4
count = 0
while count < 5:
    print(count)
    count += 1
```

3. Loop Control Statements

- **break**: Exits the loop prematurely.

- **continue**: Skips the current iteration and moves to the next.

- **pass**: A null operation; it acts as a placeholder.

Example of `break` and `continue`:

```
for i in range(10):
    if i == 5:
        break  # Exits the loop when i is 5
    if i % 2 == 0:
        continue  # Skips even numbers
    print(i)  # Prints only odd numbers: 1, 3
```

When to Use Which Loop?

- For Loops: Best for iterating over a collection or a fixed range.

- While Loops: Suitable when the number of iterations is not known beforehand and depends on a condition.

FUNCTIONS

Function: Function can be defines as ” Programmers wants to break up a program into segments commonly known as Function”.

-Functions provide Modularity.

-Functions provide code reusability.

Functions in Python are a way to organize and reuse code. They allow you to encapsulate a block of code that performs a specific task, making your programs more modular and easier to manage.

-function using the def keyword, followed by the function name and parentheses.

SYNTAX:

```
def function_name(parameters):
    # code block
    return value
```

Return Statement

A function can return a value using the return statement. If no return statement is present, the function will return None.

Example:

```
def multiply(a, b):
   return a * b
result = multiply(3, 4)  # result will be 12
```

Lambda Functions

A Function without having name is known as Lambda Function.

-Lambda is a Keyword.

Syntax:

```
Lambda arguments  : expression
```

Example:

```
Sum=lambda x,y : x+y
Print(“sum=”sum(3,5))
```