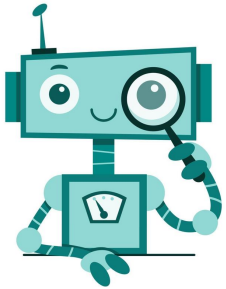
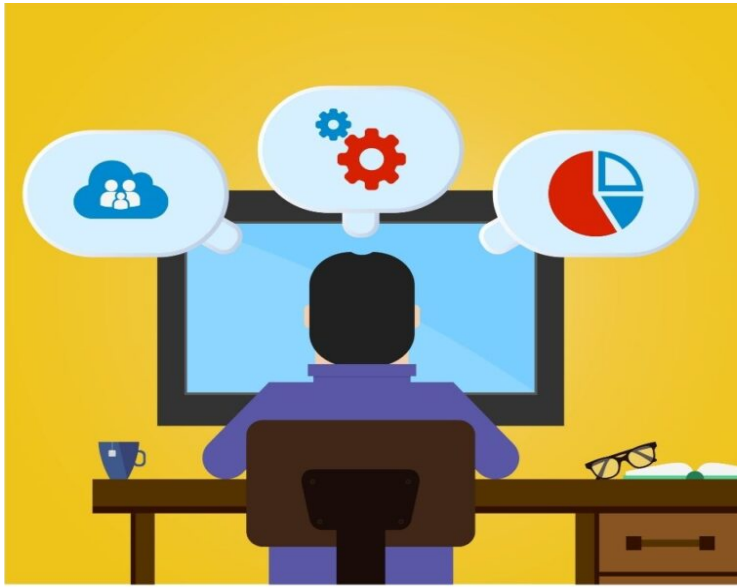
Artificial Intelligence (AI)



Week-5

Agenda

- Loops
- Functions
- String Handling

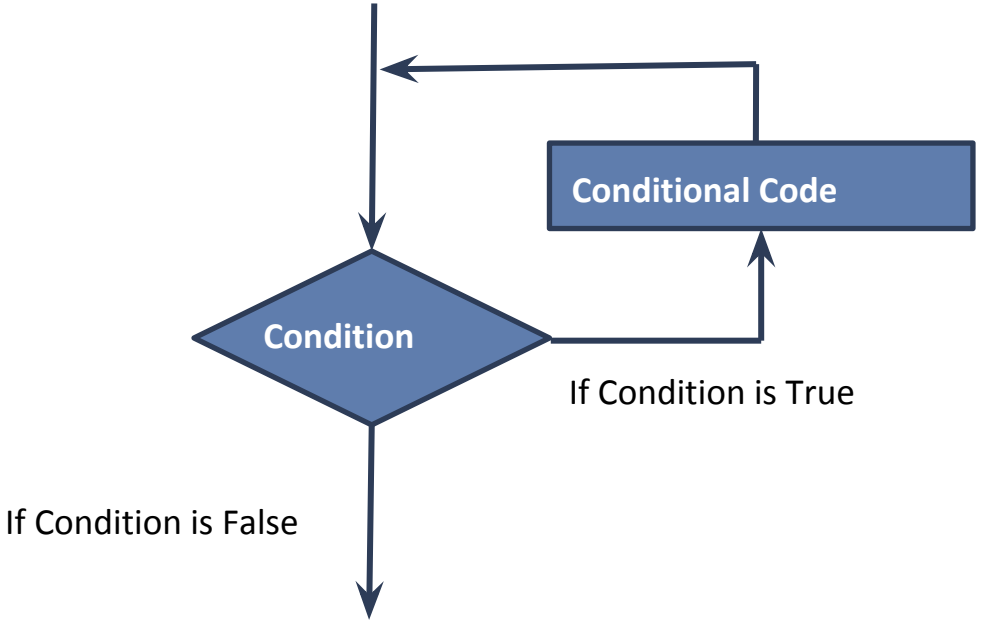


LOOPS :

- A loop is a **sequence of instructions** that are repeated over and over until a particular condition is satisfied.
- Python provide **Three** types of loops with similar basic functionality but different syntax and condition checking time.

Types of Loops:

- while loop
- for loop
- nested loop



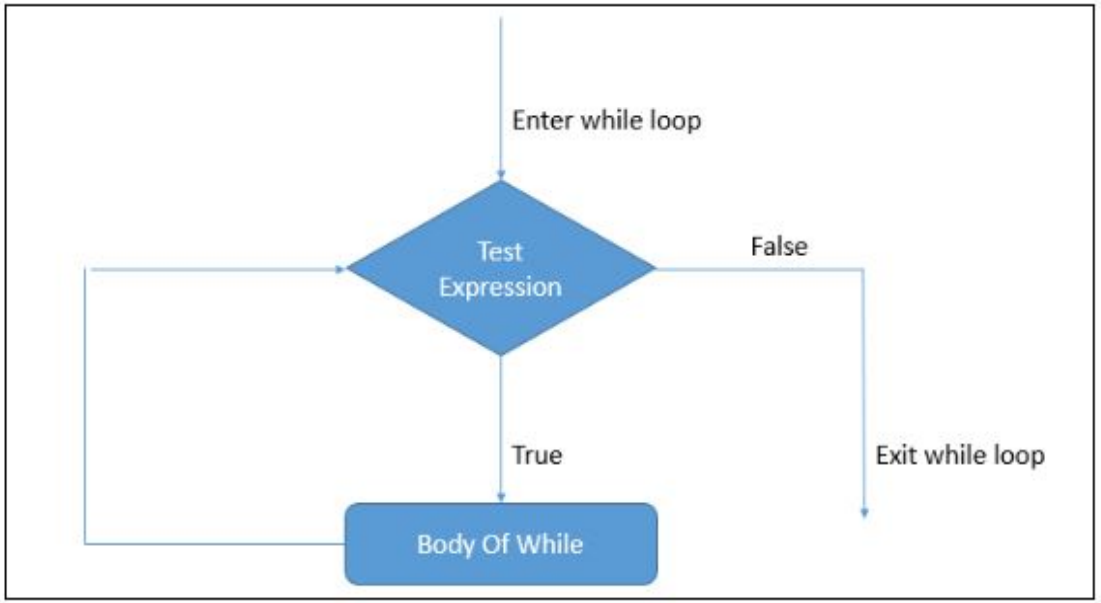
While Loop :

while loop is used to execute a block of statements as long as the given condition is **True**.

Syntax :

```
while condition:
    Statements
```

Flow Chart :


Enter while loop
Test Expression
False
True
Exit while loop
Body Of While

Example :

```
num = 1
while num <= 5:
   print("Hello World!")
   num += 1
```

OUTPUT:

Hello World!
Hello World!
Hello World!
Hello World!
Hello World!

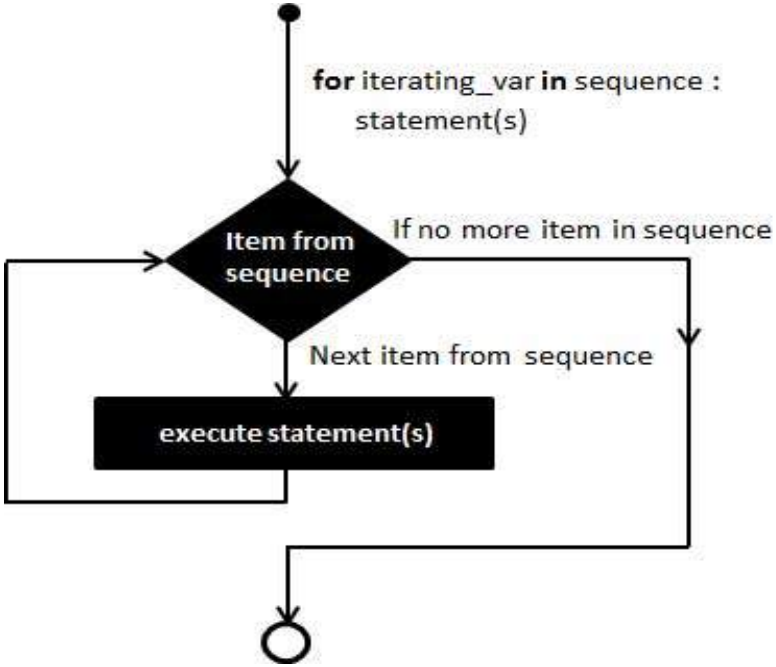
For Loop :

A for loop in Python is used to iterate over a sequence that is either a list, a tuple, a dictionary, a set, or a string

Syntax:

```
for item in sequence:
    Statements(code)
```

Flow Chart :



https://editor.analyticsvidhya.com/uploads/62260python_for_loop.jpg

Example :

```
list = [1, 2, 3, 4, 5]
for num in list:
    print(num)
```

Nested loops:

Nested Loop is a loop inside another loop. We can use any loop inside any type of loop.

The inner loop will be executed one time for each iteration of outer loop.

Example: Create a list of even numbers between 1 to 10

```
even_num = [ ]
for item in range(1, 11):
    while item % 2 == 0:
        even_num.append(item)
        break
print("Even Numbers: ", even_num)
```

Loop Control Statements:

There are three types of loop control statements:

- **break statement:** The break statement stops the execution before it has looped through all the items:
- **continue statement:** It is used to skip the current iteration & continue with the next one.
- **pass statement:** A null statement generally used as a placeholder.

Example : Break statement

Create a list of the odd numbers from 1 to 20

```
num = 1
odd_nums = []
while num:
    if num % 2 != 0:
        odd_nums.append(num)
    if num >=20:
        break
    num += 1
print("Odd numbers: ", odd_nums)
```

Example : continue statement

```
num = 0
while num < 10:
    num += 1
    if num == 5:
        continue
    print(num)
```

Skip the iteration if the current number is 5

```
for num in range(1, 11):
    if num == 5:
        continue
    print(num)
```

FUNCTIONS:

Function is a **block of reusable code** that performs a specific task. It help to break program into **small modules**.

In **Python** we have many built in functions, but we can also create our own functions known as **User-defined Functions.**

Syntax:

```
def function_name():
        Statements
```

def is a keyword to declare a function that specifies the starting of function block. It is followed by a **function_name** followed by **parenthesis().**

Example :

Creating a simple function to print Hello World in Python.

```
def Hello():
    print("Hello World")
```

Calling a Function:

To execute a Function, it needs to be called. In order to call a function, we simply type the function name.

Calling a Function

Calling a defined function in Python:

```
Hello()
```

Docstring in a function:

Docstrings stands for Documentation strings used to describe what the function does.

Inclusion of docstring in a function is optional.

Example :

```
def Hello():
    " " "This is a very first function created
in Python " " "
    print("Hello World")
```

if we run this code, we will see exact same output as that of a function Without Docstring.

Function with return Statement

Statement that allows a function to return a **specific value**. We can use **Expression** with the **return** keyword.

```
return expression
```

If there is **no expression** in the statement, then the function will return the **None** object

Example :

```
print(func1("March"))
Hello, March. Good morning!
None
```

None is the returned value since func1() directly prints the name and **no** return statement is used.

Example :

```
def func2(a,b):
    return a+b
func2(5,2)    #Calling function with a return statement
```

- Our function is accepting two parameters **a** and **b**. As a result, it is returning the **sum** of **a + b.**
- Here, our function has accepted two **arguments** 5 and 2, and return the addition of these two arguments which is **7**(output).

Function Parameters and Arguments

There are two types of data passed in the function:

Parameters:

The data recieved in the function defination is called Parameters.

Parameters must be a variable to hold incoming values.

Arguments:

The data Passed in the function call is called Arguments. It can be literals, variables or expressions.

Built In Functions:

Functions that have pre-defined Functionalities are called **Built In Functions**

Here are some built in functions in python:

- **all() :** Returns True if all items in an iterable object (such as list, dictionary, etc.) are true.
- **any() :** Returns True if any item in an iterable object is true
- **bool() :** Returns the boolean value of the specified object.

Built In Functions:

- **dict():** Returns a dictionary (Array).
- **float():** Returns a floating-point number.
- **format():** Formats a specified value.
- **help():** Executes the built-in help system.
- **id():** Returns the id of an object.
- **input():** Allowing user input.
- **int():** Returns an integer number.

Built In Functions:

- **isinstance():** Returns True if a specified object is an instance of a specified object.
- **len():** Returns the length of an object.
- **list():** Returns a list.
- **max():** Returns the largest item in an iterable.
- **min():** Returns the smallest item in an iterable.
- **next():** Returns the next item in an iterable.

Built In Functions:

- **open():** Opens a file and returns a file object
- **pow():**Returns the value of x to the power of y
- **range():** Returns a sequence of numbers, starting from 0 and increments by 1 (by default)
- **print():** Prints to the standard output device.
- **round():** Rounds a numbers.
- **set():** Returns a new set object.

Built In Functions:

- **slice():** Returns a slice object
- **object():** Returns a new object
- **sorted():** Returns a sorted list
- **str():** Returns a string object
- **sum():** Sums the items of an iterator
- **type():** Returns the type of an object
- **super():** Returns an object that represents the parent class

String Handling:

The **string** is a sequence of characters written in single quotes or in double quotes.

These characters can be **letters, numbers or symbols**.

```
string1= 'This is single quoted string'
string2= "This is double quoted string"
a ="hello123"
b ="*#%$"
c = "12*&%$"
```

How to use quotation marks in Strings?

By using Forward Slash (/) in string we use quotation marks, For example:

```
string1 =’Hey Don\’t’
```

String Operations:

String length: len(string)

It gives the Length of the string.

```
string1 =”Hello Python”
print (len(string1))
```

String Concatenation:

Concatenation is the joining together of two strings

Example:

```
string1 =”Hello “
string2=”Python”
print (string1 + string2)
```

```
string1 =”Hello “
num=100
print (string1 + str(num))
```

Print a String multiple times

```
string1 =”Hello “;
print (string1 *10);
```

String Formatting:

String formatting is the process of **formatting** string into a nicer output

There are four different ways to perform **string formatting**:

- Formatting with % Operator.
- Formatting with format() string method.
- Formatting with string literals, called f-strings.
- Formatting with String Template Class

Formatting with % Operator:

The **oldest** method of string formatting. **Inject** multiple strings at a time and also use variables to insert objects in the string.

```
print('There are %d students in a class.' %25)

print('She stood up and %s to the class.'
%'spoke')
```

Output: There are 25 students in a class.
She stood up and spoke to the class.

'%s' is used to inject strings,
'%d' for integers, **'%f'** for floating-point values,
'%b' for binary format.

Example:

You can use **multiple** format conversion types in a single print statement

```
variable = 10

string = "Variable as integer = %d \n\
Variable as float = %f" %(variable, variable)

print (string)
```

Output: Variable as integer = 10
Variable as float = 10.000000

Formatting with format() string method.

Syntax:

```
'String here {} then also {}'.format('something1','something2')
```

The**.format()** method has many advantages. **Like** we can insert **object** by using index-based position,by using assigned **keywords**, **reuse** the inserted objects to avoid duplication

Formatting with format() string method.

```
print('We all are {}.'.format('equal'))

print('{2} {1} {0}'.format('directions','the', 'Read'))

print('a: {a}, b: {b}, c: {c}'.format(a = 1,b = 'Two',c = 12.3))

print('The first {p} was alright, but the {p} {p} was tough.'.format(p =
'second'))
```

Output: We all are equal.
Read the directions.
a: 1, b: Two, c: 12.3
The first second was alright, but the second second was tough.

F-strings

F-strings provide a concise and convenient way to embed python expressions inside string literals for formatting.

```
name = 'Alee'
 print(f"My name is {name}.")

a = 5
b = 10
print(f"He said his age is {2 * (a + b)}.")
```

Output: My name is Alee.
He said his age is 30.

String Template Class:

- Python string template class is used to create a **simple** template string.
- It is created by passing template strings as **argument**, Where string formatting **operators** used for the percentage sign **%** for substitutions and the template object uses dollar signs **$**.

String Methods:

Python has a **set of built-in** methods. All string methods returns new values **without** changing the original string.

- **capitalize():** Converts the first character to upper case
- **center():** Returns a centered string
- **count():** Returns the number of times a specified value occurs in a string
- **endswith():** Returns true if the string ends with the specified value

String Methods:

- **expandtabs():** Sets the tab size of the string
- **find():**Searches the string for a specified value and returns the position of where it was found
- **index():** Searches the string for a specified value and returns the position of where it was found
- **title():**Converts the first character of each word to upper case
- **translate():** Returns a translated string
- **upper():**Converts a string into upper case

String Methods:

isalnum(): Returns True if all characters in the string are alphanumeric

isalpha(): Returns True if all characters in the string are in the alphabet

isdecimal(): Returns True if all characters in the string are decimals

isdigit(): Returns True if all characters in the string are digits

islower(): Returns True if all characters in the string are lower case

String Methods:

isprintable(): Returns True if all characters in the string are printable

isspace(): Returns True if all characters in the string are whitespaces

istitle(): Returns True if the string follows the rules of a title

isupper(): Returns True if all characters in the string are upper case

join(): Joins the elements of an iterable to the end of the string

String Methods:

lower(): Converts a string into lower case

replace(): Returns a string where a specified value is replaced with a specified value

split(): Splits the string at the specified separator, and returns a list

startswith(): Returns true if the string starts with the specified value

strip(): Returns a trimmed version of the string

swapcase(): Swaps cases, lower case becomes upper case and vice versa

Lab Tasks

Number Pattern Recognition with Loops:

- Implement a Python program using a for loop.
- Print numbers from 1 to 100.
- Replace multiples of 3 with ‘AI’, multiples of 5 with ‘Lab’, and multiples of both with ‘AILab’.

Text Analysis Function:

- Develop a Python function named analyze_text.
- The function should count uppercase letters, lowercase letters, and digits in a given string.
- Return the counts in a dictionary format.

Word Reversal in Strings:

- Write a Python program to reverse word order in a sentence.
- For the input ‘Artificial Intelligence Lab’, the output should be ‘Lab Intelligence Artificial’.

Back to main