**Day-05**

**27-02-2025**

**================**

**Python Programming Concepts:**

**=====================**

**Keywords:**

**========**

-> Python Library,

 packages

 modules

 Python files

 data, functions, classes, methods etc.

-> Keywords are "pre-defined words"

-> 35 keywords

-> There is a module named as "keyword" has included with all the list of 35 keywords.

-> All 35-keyword's list in "keyword" module can be uniquely identified with a name "kwlist".

-> When we want to display all the 35-keywords:

 we need to import the keyword module

Q: WRITE A PROGRAM IN PYTHON TO DISPLAY ALL THE KEYWORDS OF PYTHON.

import keyword

print(keyword.kwlist) # kwlist is the member of keyword module

['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

-> The python keywords are classified into two categories:

 1) Reserved words ==> 32

 2) Reserved Literals/Values ==> 3

 True, False, None

**Identifiers**

**=======**

-> That every program can define with some elements like:

 functions, classes, methods, data etc.

-> While defining these in the python, we need to define a name to uniquely identify those elements. In this case, we can use "Identifiers".

-> Identifiers are names can be used to name any entity while the programming.

Ex:

def greetings():

 print("Good Evening")

 print("Welcome To Ashok IT!")

class Employee:

 eid = 101

 ename = "Prakask"

 designation = "Software Developer"

-> In the above example:

 greetings is the name for the function

 Employee is the name for the class etc.

to define these type of names, we can use "identifiers".

Rules for Identifiers:

-----------------------

1) The identifiers are allowed to define with:

 alphabets

 digits (0 to 9)

 underscore sign (\_)

2) If the identifier with any other characters like: space, . etc., ==> Invalid identifiers.

Ex: abc def = 23 ==> Syntax error

3) Identifier never begin with digit but it can allow to begin either alphabet or \_.

Ex: 9abc = 100 ==> Syntax error

\_abc = 121

abc = 211 ==> Valid

4) Keywords are not allowed as identifiers.

5) Identifiers in python are case sensitive.

**Different cases for Identifiers:**

**===================**

-> there are 5-different cases are available to define identifiers:

 1) Lower case

 2) Upper case

 3) Title case

 4) Capitalize case

 5) Camel case

-> When all the letters of an identifier with lower case alphabets that is called as "Lowercase Identifier".

Ex: employeename = "Rajesh"

-> When all the letters of an identifier with upper case alphabets that is called as "Uppercase Identifier".

Ex: EMPLOYEENAME = "Rakesh"

-> In the entire identifier, only the starting letter with capital and remaining all in lower case that identifier is called as "Title case identifier".

Ex: Employeename = "Rahul"

-> In the Entire Identifier, the first letter of each word with capital and remaining all in lowercase such type of identifiers are called as "Capitalize case Identifiers".

Ex: EmployeeName = "Keerthi"

-> Camel case identifiers can represent that:

 from the second word that every word's beginning letter is capital and remaining all in lowercase.

Ex: employeeName = "Kishore"

pythonProgrammingLanguage = 121

**Variables**

**=======**

-> Variable is a name

 can be used to store a value of any type.

-> Python is Dynamically Typed programming Language.

High-level Programming languages:

 classified into three types:

 1) Statically Typed

 2) Strongly Typed

 3) Dynamically Typed

-> Statically Typed Programming Languages are consider the data definition based on the type of the data. That means, the data what we can assign that can be change according to the type of the data what we can specify.

Ex: C

#include<stdio.h>

void main(){

 int a;

 a = 123;

 printf("%d\n",a);

 a = 12.234;

 printf("%d",a);

}

-> The strongly Typed programming languages, can allow the data definition based on the Type of the data as like the statically typed. But the value what we have assigned to the variable must be the same or equivalent type of the specified datatype of the variable.

Ex: Java

class Variables{

 public static void main(String[] args){

 int a;

 a = 100;

 System.out.println(a);

 a = 102.234;

 }

}

-> Dynamically Typed programing languages:

 1) not requires the type of the data before the variable name.

 2) The Python can understand the type of the value based on the assignment.

-> The variables in python can be defined by using the following syntax:

 data-name/variable-name/identifier = value

**type():**

**=====**

-> can give the type of the value what we have assigned to the variable.

Syntax:

 type(data/variable-name)

a = 121

print(type(a)) # int

a = 1.023

print(type(a)) # float