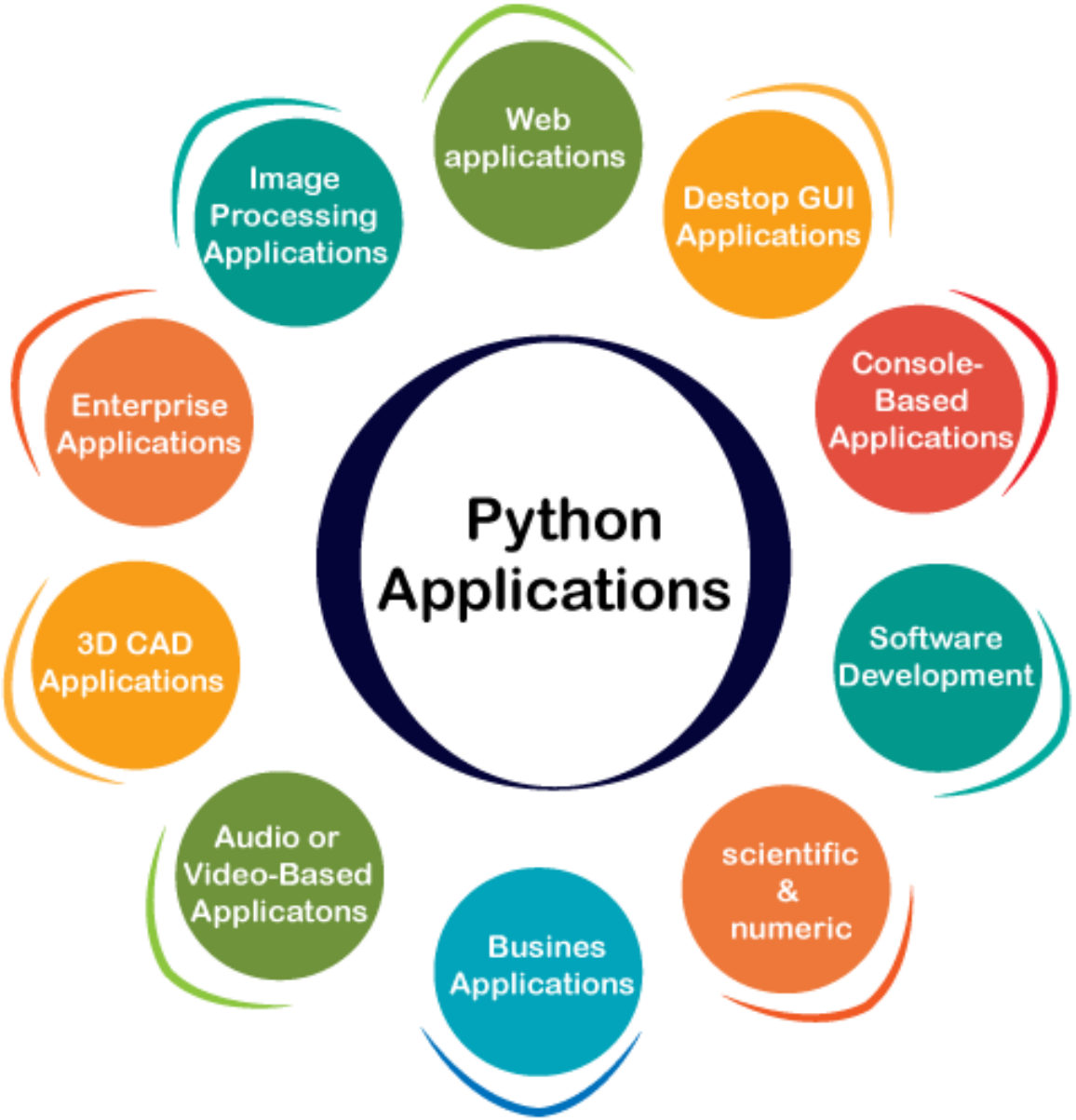
# Introduction to Python

**Eng. Fatma Gamal**

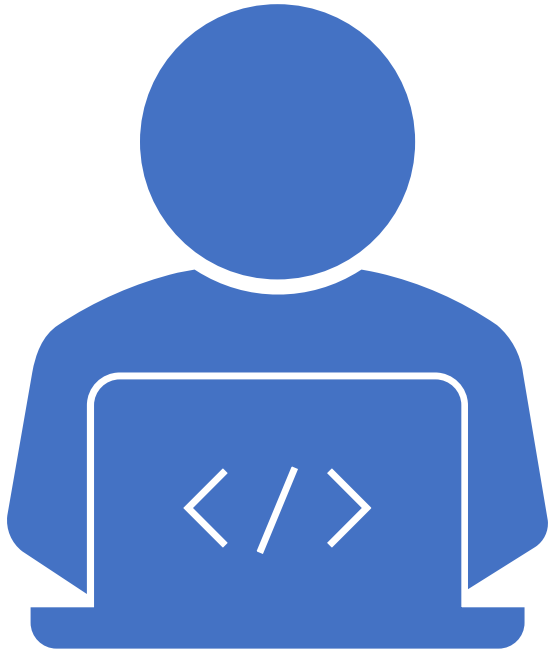
# What is Python?

- **Python** is a popular programming language. It was created by **Guido van Rossum**, released in **1991**.



# Python Features

- Easy to Learn and Use
- Expressive Language
- Interpreted Language
- Cross-platform Language
- Free and Open Source
- Object-Oriented Language
- Extensible
- Large Standard Library
- GUI Programming Support
- Integrated
- Dynamic Memory Allocation



# Python Installation

- **Anaconda distribution**

The **Anaconda distribution** is a repackaging of Python aimed at developers who use Python for **data science**. It provides a management GUI, a slew of scientifically oriented work environments, and tools to simplify the process of using Python for data crunching.

https://www.anaconda.com/products/individual

# Hello Python

**print(“hello Python”)**

# Python Comments

- Comments starts with a #
- Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (**triple quotes**) in your code

# Python Variables

- Python has **no command** for declaring a variable. A variable is created the moment you **first assign a value to it.**
- Variables do not need to be declared with any **particular type** and can even **change** type after they have been set.
- You can get the data type of a variable with the **type()** function.
- String variables can be declared either by using **single** or **double** quote
- Variable names are **case-sensitive**

# Python Variables cont.

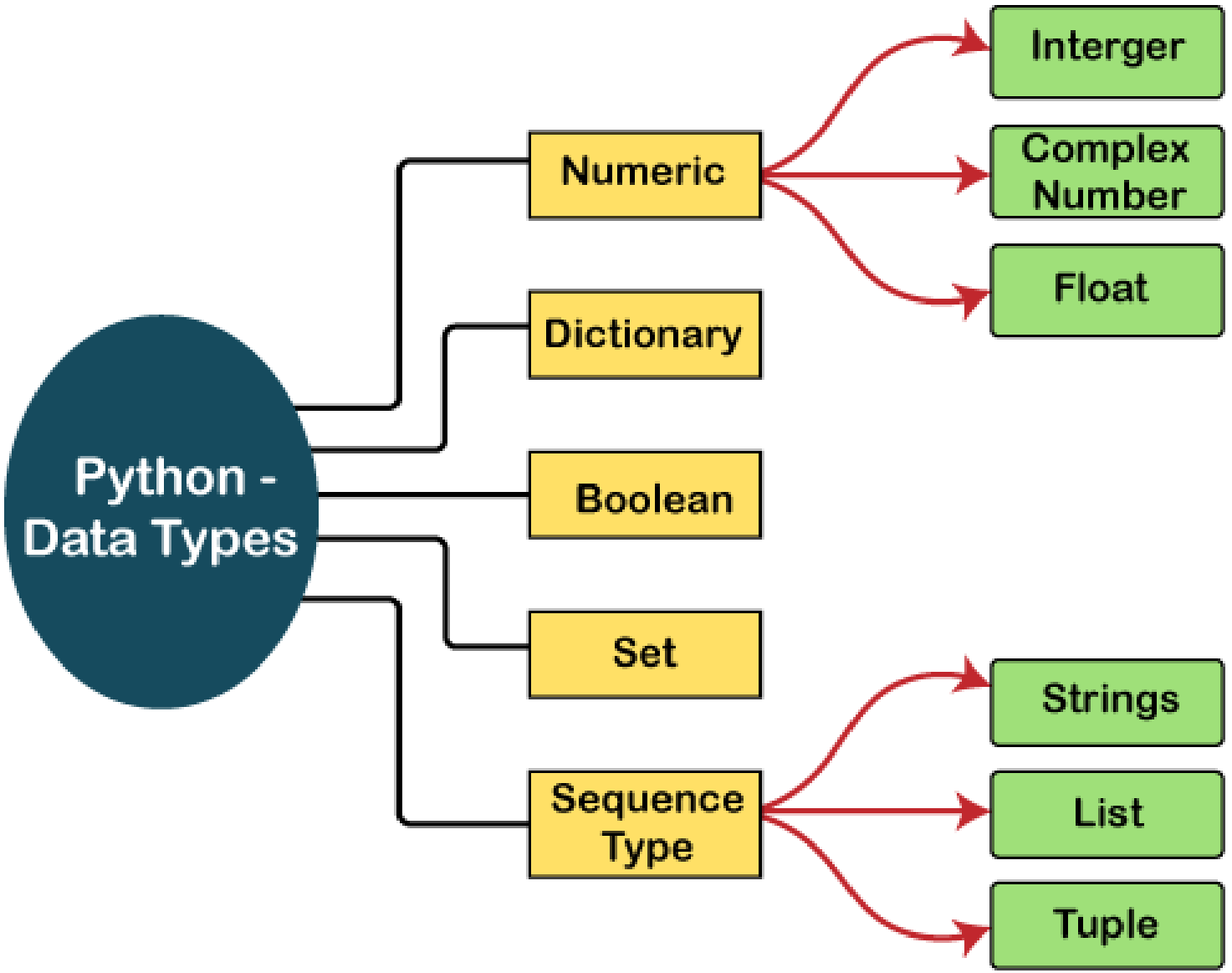
- Python allows you to assign values to **multiple variables in one line**, and you can assign the same value to multiple variables in one line

```
x, y, z = "Orange", "Banana", "Cherry"
```

- If you have a collection of values in a list, tuple etc. Python allows you extract the values into variables. This is called **unpacking**.

```
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits
```

# Python Data Types



# Python Lists

- A list in Python is used to store the sequence of various types of data.
- The items in the list are separated with the comma (,) and enclosed with the square brackets [].

```
L1 = ["John", 102, "USA"]
```

# List indexing and splitting

- The index starts from 0 and goes to length - 1. The first element of the list is stored at the 0th index, the second element of the list is stored at the 1st index, and so on.

List = [ 0, 1, 2, 3, 4, 5]

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|

List[0] = 0

List[1] = 1

List[2] = 2

List[3] = 3

List[4] = 4

List[5] = 5

List[0:] = [0,1,2,3,4,5]

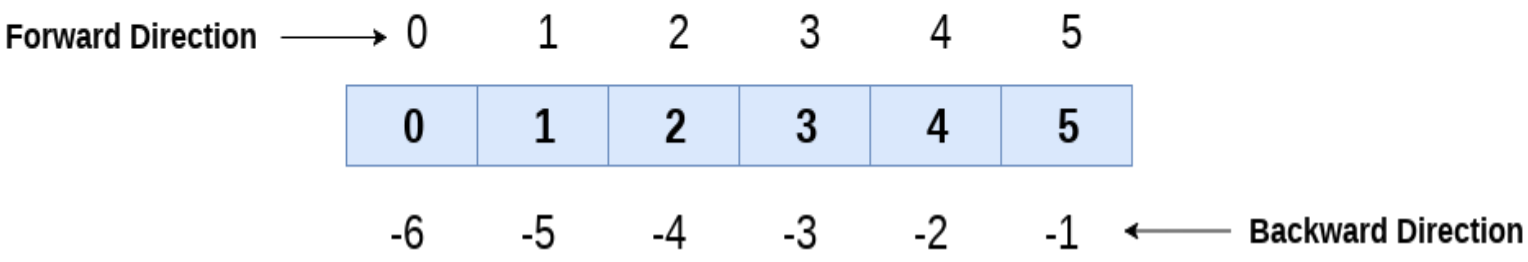
List[:] = [0,1,2,3,4,5]

List[2:4] = [2, 3]

List[1:3] = [1, 2]

List[:4] = [0, 1, 2, 3]

List = [ 0, 1, 2, 3, 4, 5]



# Updating List values

- **Lists** are the most versatile data structures in Python since they are **ordered** and **mutable**, and their values can be updated by using the **slice** and **assignment** operator.
- Python also provides **append()** and **insert()** methods, which can be used to add values to the list.
- The list elements can also be deleted by using the **del** keyword. Python also provides us the **remove()** method if we do not know which element is to be deleted from the list.

# List Operations

| Operator | Description | Example |
|---|---|---|
| Repetition | The repetition operator enables the list elements to be repeated multiple times. | l1*2 = [1, 2, 3, 4, 1, 2, 3, 4] |
| Concatenation | It concatenates the list mentioned on either side of the operator. | l1+l2 = [1, 2, 3, 4, 5, 6, 7, 8] |
| Membership | It returns true if a particular item exists in a particular list otherwise false. | print(2 in l1) prints True. |
| Iteration | The for loop is used to iterate over the list elements. | for i in l1:<br>print(i) |
| Length | It is used to get the length of the list | len(l1) = 4 |

# Python Tuples

- Python **Tuple** is used to store the sequence of **immutable** Python objects
- A tuple can be written as the collection of comma-separated (,) values enclosed with the small () brackets. The parentheses are optional, but it is good practice to use.

```
T= (101, "Banana", "Orange")
```

# Python Set

- A Python **set** is the collection of the **unordered** items. Each element in the set must be **unique**, and the sets **remove the duplicate** elements. Sets are **mutable** which means we can modify it after its creation.
- there is **no index** attached to the elements of the set, i.e., we cannot directly access any element of the set by the index.

```
Days = {"Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday", "Sunday"}
```

# Python Set Cont.

- Python also provides the **set()** method, which can be used to create the set by the passed sequence.
- Python provides the **add()** method and **update()** method which can be used to add some particular item to the set. The add() method is used to add a **single element** whereas the update() method is used to add **multiple elements** to the set.
- Python provides the **discard()** and **remove()** method which can be used to remove the items from the set, using discard() function if the item does not exist in the set, then the set remain unchanged whereas remove() method will through an error.
- Python provides the **clear()** method to remove all the items from the set

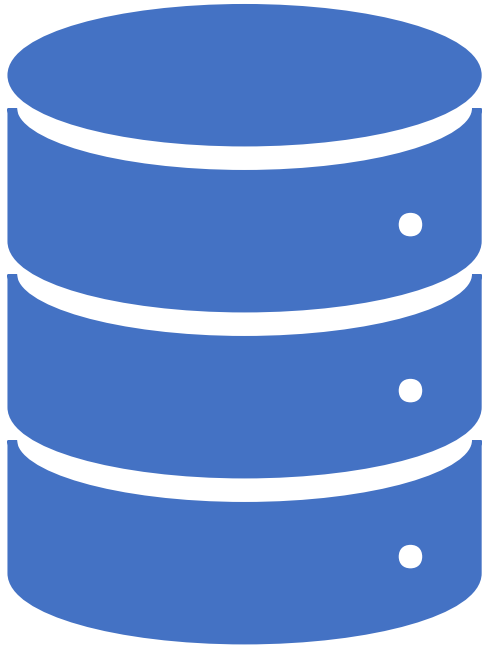
# Python Set Operations

- Set can be performed mathematical operation such as **union**, **intersection**, **difference**, and **symmetric difference**. Python provides the facility to carry out these operations with operators or methods.
- The union of two sets is calculated by using the **pipe (|)** operator. The union of the two sets contains all the items that are present in both the sets.
- Python also provides the **union()** method which can also be used to calculate the union of two sets.

```
Days1.union(Days2)
```

# Python Set Operations Cont.

- The **intersection** of two sets can be performed by the **and &** operator or the **intersection()** function. The intersection of the two sets is given as the set of the elements that common in both sets.
- The **intersection_update()** method removes the items from the original set that are not present in both the sets.
- The **difference** of two sets can be calculated by using the **subtraction (-)** operator or **intersection()** method.



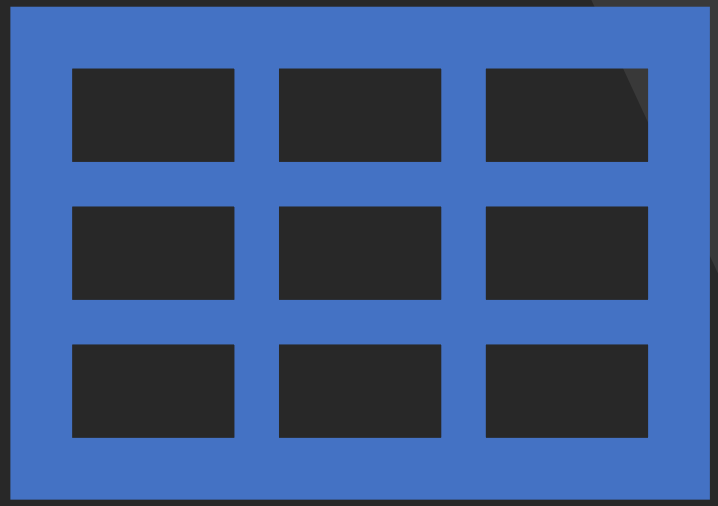
# Python Dictionary

- Python Dictionary is used to store the data in a key-value pair format.
- It is the mutable data-structure
- Keys must be a single element
- Value can be any type such as list, tuple, integer, etc.

```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
```

# Python Arrays

```
from array import *
arrayName = array(typecode, [initializers])
```



# Python Arithmetic Operators

| Operator | Description |
|---|---|
| **+ (Addition)** | It is used to add two operands. For example, if a = 20, b = 10 => a+b = 30 |
| **- (Subtraction)** | It is used to subtract the second operand from the first operand. If the first operand is less than the second operand, the value results negative. For example, if a = 20, b = 10 => a - b = 10 |
| **/ (divide)** | It returns the quotient after dividing the first operand by the second operand. For example, if a = 20, b = 10 => a/b = 2.0 |
| *** (Multiplication)** | It is used to multiply one operand with the other. For example, if a = 20, b = 10 => a * b = 200 |
| **% (reminder)** | It returns the reminder after dividing the first operand by the second operand. For example, if a = 20, b = 10 => a%b = 0 |
| **** (Exponent)** | It is an exponent operator represented as it calculates the first operand power to the second operand. |
| **// (Floor division)** | It gives the floor value of the quotient produced by dividing the two operands. |

# Python Comparison operator

| Operator | Description |
| --- | --- |
| == | If the value of two operands is equal, then the condition becomes true. |
| != | If the value of two operands is not equal, then the condition becomes true. |
| <= | If the first operand is less than or equal to the second operand, then the condition becomes true. |
| >= | If the first operand is greater than or equal to the second operand, then the condition becomes true. |
| > | If the first operand is greater than the second operand, then the condition becomes true. |
| < | If the first operand is less than the second operand, then the condition becomes true. |

# Python Logical Operators

| | |
|---|---|
| and | If both the expression are true, then the condition will be true. If a and b are the two expressions, a → true, b → true => a and b → true. |
| or | If one of the expressions is true, then the condition will be true. If a and b are the two expressions, a → true, b → false => a or b → true. |
| not | If an expression **a** is true, then not (a) will be false and vice versa. |

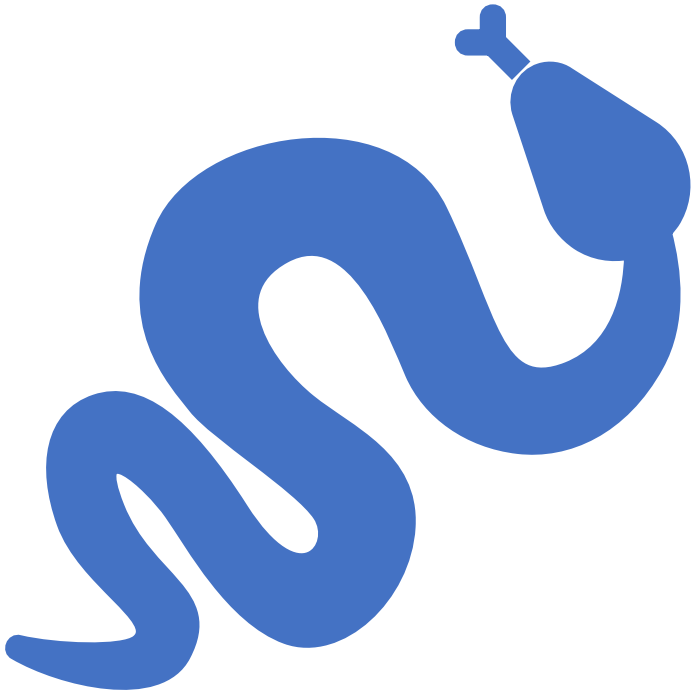
# Python If-else statements

```
if expression 1:
    # block of statements

elif expression 2:
    # block of statements

elif expression 3:
    # block of statements
 else:
    # block of statements
```

# Python for loop

```
for iterating_var in sequence:
    statement(s)
```

```
for i in range(1,11):
    c = n*i
```



# Python While loop

```
while expression:
    statements
```

# Python Function

```
def my_function(parameters):
    function_block
return expression
```

# Python Math Module

- math.log()
- math.exp()
- math.pow(x,y)
- math.factorial()

# Python Sheet

1. Write a Python program to find the area of a triangle.
2. Write a Python Program to Find the Factorial of a Number.
3. Write a Python Program to Check if a Number is Odd or Even.
4. Write a Python Program to Display the Multiplication Table.

# Python Case Study

1. Write an Employee Management System in Python. The script will contain the following operations:
   - Add Employee
   - Remove Employee
   - Promote Employee
   - Display Employees

*Python 3.8.5 Shell*

File Edit Shell Debug Options Window Help

```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\HP\AppData\Local\Programs\Python\Python38\employ management system.py
Welcome to Employ Management Record
Press
1 to Add Employ
2 to Remove Employ
3 to Promote Employ
4 to Display Employees
5 to Exit
Enter your Choice 1
```



Ln: 12 Col: 18



00:31
09-04-2021