Agenda:

1. Introduction to Python and its advantages.
2. Python Variables and Data Types.
3. Conditional Statements (if-else).
4. Looping Statements (for and while loops).
5. Functions in Python.

- Introduction to Python and its advantages.
  Python is an object-oriented, interpreted, high-level programming language. It is versatile and used for GUI and web applications. Python allows focusing on business logic, reducing common programming tasks.

- Python Variables and Data Types.
  In Python, variables are used to store and retrieve data in memory. They can be assigned values directly or obtained from user input using the input function. Variables should start with a letter or underscore and cannot start with numbers. Multiple assignments are also possible in Python.

  Python Variables – Assigning Values
  Single Assignment Example:
  ```
  x = 10
  name = "John"
  salary = 50000.00

  print(x)     # Output: 10
  print(name)  # Output: John
  print(salary)  # Output: 50000.0
  ```

  Multiple Assignment Example:
  ```
  a, b, c = 1, 2, 3

  print(a)  # Output: 1
  print(b)  # Output: 2
  print(c)  # Output: 3
  ```

  Python Variables – Getting User Input
  In Python, to get input from the user and assign that value to a variable, we use the input function. The input function prompts the user to enter a value. It allows for interactive input from the user.

  ```
  name = input("Enter your name: ")
  age = int(input("Enter your age: "))  # Converting input to integer
  ```

print("Hello, " + name + "! You are " + str(age) + " years old.")

## Introduction to Data Types

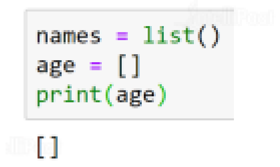
In Python, data types are used to indicate the type of data stored in a variable, such as numbers, text, or boolean values. This information helps the interpreter determine valid operations on variables. There are six types of data types in Python: Integer, Float, String, Boolean, List, Tuple.

## Lists

- In Python Lists are used to store a collection of items in a sequential order e.g. Items in a wish list of a customer.
- Since they are stored in a sequential order, a special number based on their position in the list called index is assigned to them and they are accessed using these index values.
- These indexes start from zero and are assigned in an increasing order i.e. from zero to n – 1 where n is the number of items in that list.
- These lists can store data of multiple data types, e.g. Integer, String, Float  Etc.

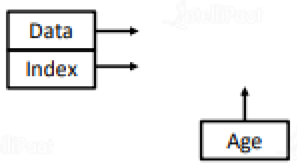
## List Operations

To create an empty list you can use the list function or use empty brackets.

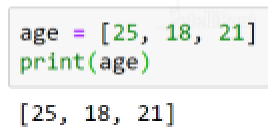


```
names = list()
age = []
print(age)
```

```
[]
```

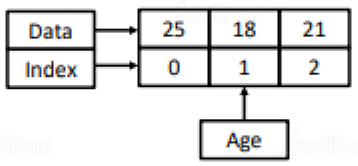


You can initialize a list with some data while creating the list, by passing values inside the list function or brackets

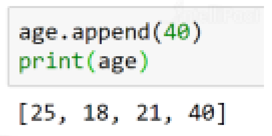


```
age = [25, 18, 21]
print(age)
```

```
[25, 18, 21]
```

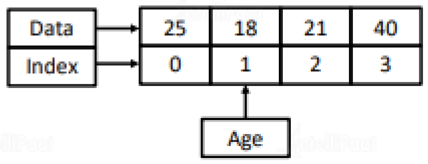


You can add data to the end of list by calling the append method on the lists .

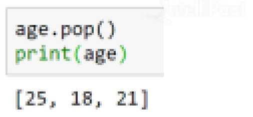


```
age.append(40)
print(age)
```

```
[25, 18, 21, 40]
```

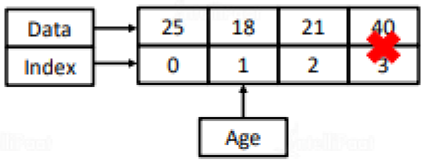


You can delete an element in a list by calling the pop method on it. Pop will delete the element at the end of the list

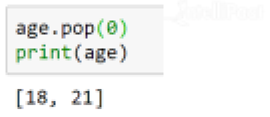


```
age.pop()
print(age)
```

```
[25, 18, 21]
```

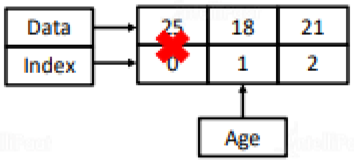


You can also delete an element at a particular index, by passing the index in as parameters in the function call.
Notice that this ends up changing the index of all the elements that occur after the deleted index.



```
age.pop(0)
print(age)
```

```
[18, 21]
```

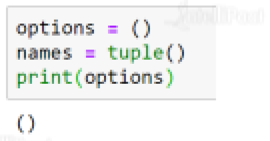


Tuples

- In Python just like Lists, Tuples are also used to store collection of data in a sequential order, the difference is that it is immutable.
- Immutable means that once you create a tuple, you can't make changes to it, i.e. add items, remove items, swap items etc.
- Tuples are especially useful when you have a collection of data which you do not wish to change in your application e.g. Days in a Week .
- Tuples will throw an error if you try to change them, so you won't be able to accidentally change the tuple.
- Like lists, elements in tuples are also accessed using their indexes.
- Tuples can also store data of multiple types such as Integer, Float Boolean etc.

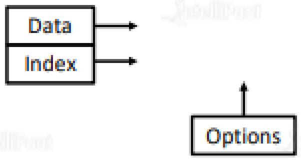
Tuple Operations

To **create** an empty tuple you can use the tuple function or use empty parenthesis .

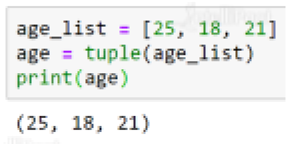


```
options = ()
names = tuple()
print(options)
```

```
()
```

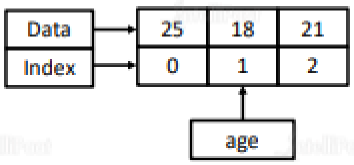


To **initialize** a tuple with some values you can either use the tuple function or use the comma syntax.
In the tuple function you need to pass in a other data structure like list and it will return a new tuple with all the values from that data structure

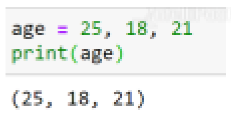


```
age_list = [25, 18, 21]
age = tuple(age_list)
print(age)
```

```
(25, 18, 21)
```

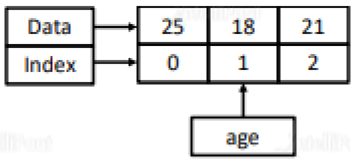


To use the comma syntax you need to have a few values separated by commas and assign them to a variable



```
age = 25, 18, 21
print(age)
```

```
(25, 18, 21)
```



You can check (**Search**) either if a particular element exists in the tuple or you can check at what index does a particular element exist
To check if a particular element exists in the tuple we use the membership operator (using the ‘in’ keyword), which returns True if element exists and False if it does not.
Eg: 21 in age - true.
You can also use the index method to find the index of a particular element in a tuple. It returns the index if the element is found if not then throws an error
age.index(21) - 2.

**Slicing** is used to get a contiguous portion of a list. For example, if wish to get a copy of elements from index 1 to 3
To slice a list we need to provide the index from where you wish to start the slice and index before which the slice ends like age[1:4]
Do note that we use 4 instead of 3 to indicate that the slice needs to stop before index 4.

Dictionaries
Dictionaries like sets are unordered collections of data but they store key value pairs, i.e. two associated values Much like sets dictionaries are also great for checking membership of keys when you have as key value pairs.

Dictionary Operations
To **create** an empty dictionary you can either use the dict function or use the curly braces syntax eg: names = {} or age = dict()
To **initialize** a dictionary with some values you can either pass some nested data structures to dict function or use the curly braces syntax You can create a dictionary with data structures like lists or tuples. These data structures need to contain either lists or tuples with 2 values each.
Notice that each value is either a list or tuple of size 2 in which index zero is the key index one is the value. You can also use curly braces syntax by separating key value pairs using commas and keys values using colon e.g. {key1 : value1, key2: value2}.
To **add** a new key value pair by using this syntax: dict_name[key] = value.
Do note that if you add values using an existing key it will overwrite the previous value.
To **remove** an element just use the del keyword with the key dictionary name e.g. del age[‘d’]
To **search** a value you need to use the membership operator with the key, If the key is found it returns true else it returns false

- Conditional Statements

  Conditional statements are used in Python to control the flow of execution based on certain conditions. There are three types of conditional statements:
  - if: Executes code when a condition is true.
  - if-else: Executes one block of code if the condition is true and some other code if statement evaluates to false.
  - if-elif: Used to check multiple conditions and execute different blocks of code based on the first condition that is true.

  Sometimes in an application we have to perform certain tasks if a given condition is true e.g. Load profile if user is logged in etc.
  To accomplish this in code we use conditional statements

- Looping Statements

  Looping is the process in which we have a code that gets executed repeatedly until a particular condition is satisfied.
  There are two kinds of loops used in python:
  - For - The for statement is used to loop over a group or collection of data.
  - While - The while statement simply loops until a condition is evaluated to False.

Creating a Function

To create a function we first need to understand the syntax of a function. A function definition can have multiple parts to it: Name, Parameters, Return statement.

**Name** - When a function is being defined we give it a name so that it can be referenced later. The name of the function is like naming a variable and follows all the same rules.
**Parameters** - Parameters or Function Arguments are the variables or data that we want our function to work on, e.g. numbers to be added , You can have any number of parameters be accepted in your function or no parameters if your function does not need it.
**Return** - A return statement is used to return the answer computer by your function.You can have no return value at the end by simply omitting the return statement in case you do not wish your function to return a value.

**Types of Functions**

There are two types of functions:

- User Defined Function - These functions are defined and used by the developers who are writing the code and wish to solve a problem for which there isn’t a function in the standard library .
- Built In Defined Function - These functions come built into the language as part of the standard library for example, functions like print, input etc.