Dhirubhai Ambani University

(Formerly known as DA-IICT)

Topic: Data Types

Course: Programming Lab

Course Code- PC503

Dr. Ankit Vijayvargiya

Assistant Professor

Room No. 4205, Faculty Block 4

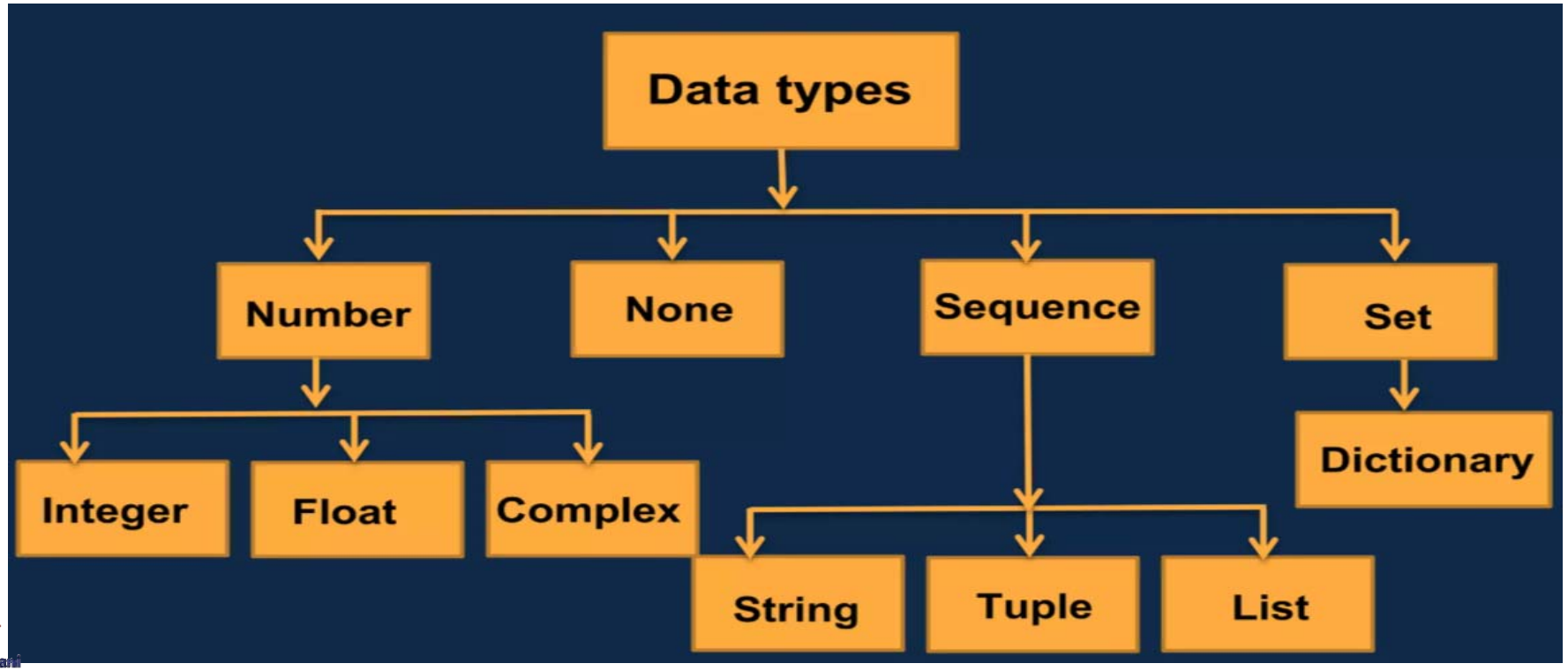
Email: ankit_Vijayvargiya[at]dau.ac.in

Phone: 079-68261628(O), 7877709590(M)

Dhirubhai Ambani University
Formerly known as DA-IICT

Data types

The term "data types" refers to classifications of data that tell a computer what kind of value is being handled and what operations can be performed on it.



Dhirubhai Ambani University
Formerly known as DA-IICT

Data types: Number/Numeric

Number data types store numeric values. They are immutable data types, which means that changing the value of a number data type results in a newly allocated object.

```
x = 5
print(id(x))
# Let's print the memory address of x

x = x + 1
print(id(x))
# Memory address changes because integers are immutable
```

```
140732584805288
140732584805320
```

How to check the datatype:

```
X=2
print(type(X))
```

```
<class 'int'>
```

```
y=3.2
print(type(y))
```

```
<class 'float'>
```

```
z=3+4j
print(type(z))
```

```
<class 'complex'>
```

Integers (default for numbers)
Z=5

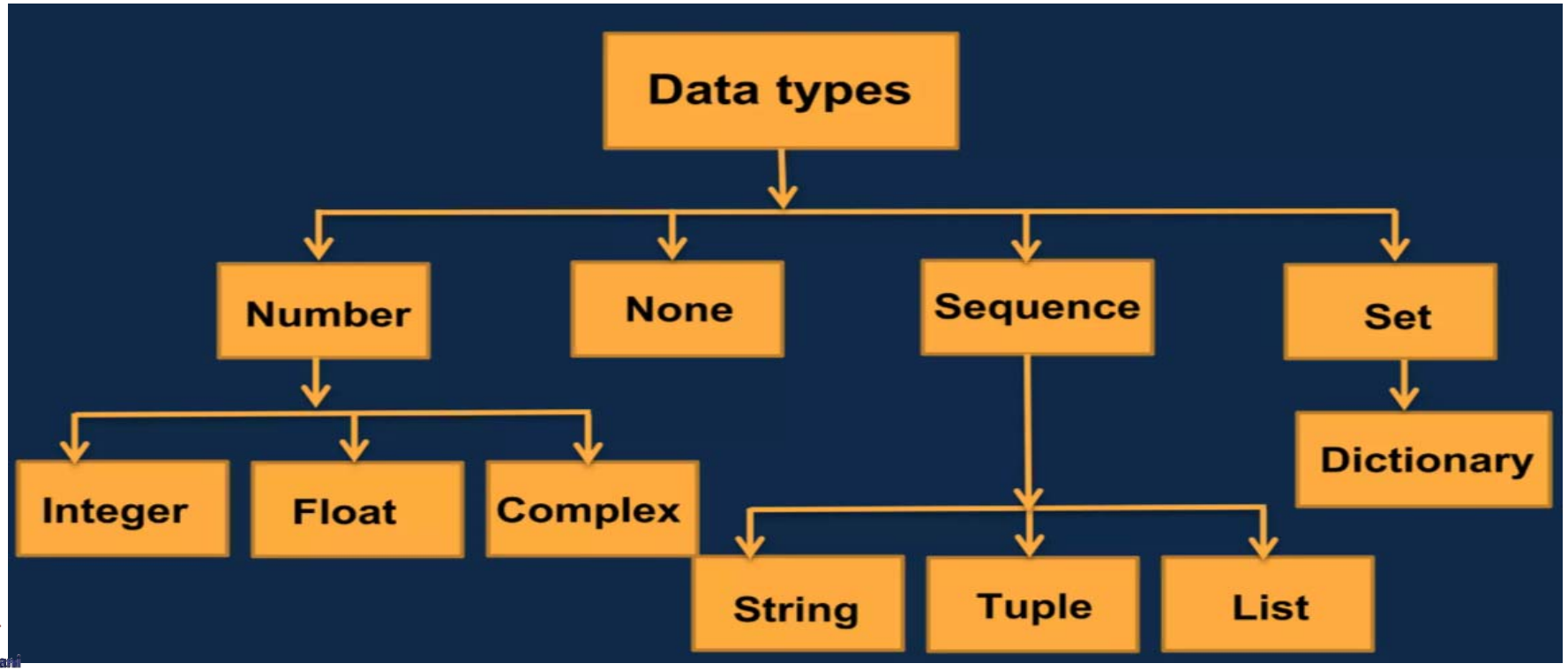
Floats
Z=5.1

Complex Number
Z=5+2j

Dhirubhai Ambani University
Formerly known as DA-IICT

Data types

A data type is a classification that specifies the kind of value a variable can hold, such as numbers, text, or boolean values.



Data types: None

None:

- **None is a special constant in Python that represents the absence of a value or a null value.**

$$Z = \text{None}$$

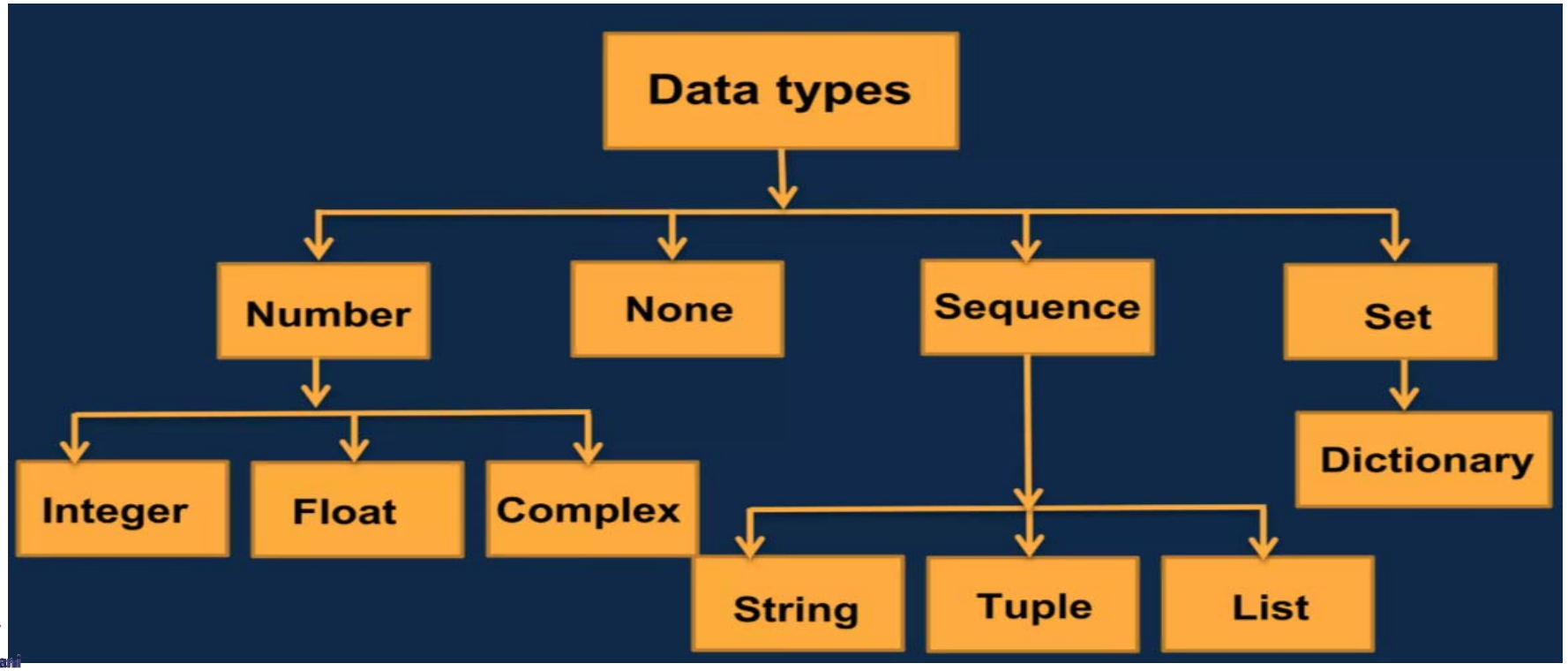
```python
def find_min(numbers):
    min_value = None  # Initially, no value is set

    for num in numbers:
        if (min_value is None) or (num < min_value):
            min_value = num  # Update max_value when it's None or a larger number is found

    return min_value

# Test case 1
nums = [10, 5, 20, 7]
print(find_min(nums))  # Output: 20

# Test case 2 (Empty list)
nums = []
print(find_min(nums))  # Output: None
```

```
5
None
```

Dhirubhai Ambani University
Formerly known as DA-IICT

Data types

A data type is a classification that specifies the kind of value a variable can hold, such as numbers, text, or boolean values.



Data types: Sequence

Strings:

- A string is a collection of one or more characters put in a single quote, double quote, or triple quote.
- In Python, there is no character data type; a character is a string of length one.

Z = ‘python’

Z = “python”

Z = “‘python”’

Dhirubhai Ambani University
Formerly known as DA-IICT

Data types: Sequence

List:

A list is an ordered sequence of items. It is one of the most used data types in Python and is very flexible. All the items in a list do not need to be of the same type.

How to create the list:

```
# empty list
        mylist = []
# list of integers
        mylist = [1, 2, 3]
# list with mixed datatypes
        mylist = [1, "Hello", 3.4]
# nested list
        mylist = ["mouse", [8, 4, 6], ['a']]
```

Data types: Sequence

User Input:

```
# User inputs numbers separated by spaces
user_input = input("Enter numbers separated by space: ")

# Split the string and convert each to integer
numbers = list(map(int, user_input.split()))

print("You entered:", numbers)
```

```
Enter numbers separated by space:  2 3 4 5 6 7
You entered: [2, 3, 4, 5, 6, 7]
```

```
# User inputs numbers separated by spaces
user_input = input("Enter numbers separated by space: ")

# Split the string and convert each to integer
numbers = list(map(str, user_input.split()))

print("You entered:", numbers)
```

```
Enter numbers separated by space:  ankit vijay vargiya matlab python
You entered: ['ankit', 'vijay', 'vargiya', 'matlab', 'python']
```

```
# Ask the user how many elements they want to enter
n = int(input("How many numbers do you want to enter? "))

# User inputs numbers separated by spaces
user_input = input(f"Enter {n} numbers separated by space: ")

# Split the string and convert each to integer
numbers = list(map(int, user_input.split()))

# Optional: Check if user entered exactly 'n' numbers
if len(numbers) != n:
    print(f"Warning: You entered {len(numbers)} numbers instead of {n}.")

print("You entered:", numbers)
```

```
How many numbers do you want to enter?  3
Enter 3 numbers separated by space:  1 2 3
You entered: [1, 2, 3]
```

```
# User inputs numbers separated by spaces
user_input = input("Enter numbers separated by space: ")

# Split the string and convert each to integer
numbers = list(map(str, user_input.split()))

print("You entered:", numbers)
```

```
Enter numbers separated by space:  2 ankit vijay
You entered: ['2', 'ankit', 'vijay']
```

```
print(numbers[0]+2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[5], line 1
----> 1 print(numbers[0]+2)

TypeError: can only concatenate str (not "int") to str
```

Data types: Sequence

Accessing List Element:

```
mylist = [1, 2 , 3, 4]
```

```
print(mylist[0])
```

```
1
```

```
mylist = [1, "Hello", 3.4]
```

```
print(mylist[1])
```

```
Hello
```

```
print(mylist[1][1])
```

```
e
```

```
mylist = ["mouse", [8, 4, 6], ['a']]
```

```
print(mylist[1][2])
```

```
6
```

Dhirubhai Ambani University
Formerly known as DA-IICT

Data types: Sequence

Slicing Operator for Python List:

```
mylist=['a','b','c','d','e','f','g','h']
```

```
# print element 3rd to 5th
print(mylist[2:5])
```

```
['c', 'd', 'e']
```

```
# print element begining to 4th
print(mylist[:4])
```

```
['a', 'b', 'c', 'd']
```

```
# print element 4th to end
print(mylist[3:])
```

```
['d', 'e', 'f', 'g', 'h']
```

```
print(mylist[:-4])
```

```
['a', 'b', 'c', 'd']
```

```
print(mylist[-4:])
```

```
['e', 'f', 'g', 'h']
```

Dhirubhai Ambani University
Formerly known as DA-IICT

Data types: Sequence

Add an element to a Python List:

1. append

```
In [27]: x= [1,'Shyam',25]

In [28]: print(x)
         [1, 'Shyam', 25]

In [29]: x.append("Male")

In [30]: print(x)
         [1, 'Shyam', 25, 'Male']
```

```
In [1]: x= [1,'Shyam',25]

In [2]: print(x)
        [1, 'Shyam', 25]

In [3]: y=['Male', 25000]

In [4]: print(y)
        ['Male', 25000]

In [5]: x.append(y)
        print(x)
        [1, 'Shyam', 25, ['Male', 25000]]
```

Dhirubhai Ambani University
Formerly known as DA-IICT

Data types: Sequence

Add an element to a Python List:

2. Insert

```
In [1]: x= [1,'Shyam',25]

In [2]: print(x)
        [1, 'Shyam', 25]

In [3]: y=['Male', 25000]

In [4]: print(y)
        ['Male', 25000]

In [5]: x.insert(3,y)

In [6]: print(x)
        [1, 'Shyam', 25, ['Male', 25000]]
```

3. extend

```
In [1]: x= [1,'Shyam',25]

In [2]: print(x)
        [1, 'Shyam', 25]

In [3]: y=['Male', 25000]

In [4]: print(y)
        ['Male', 25000]

In [5]: x.extend(y)

In [6]: print(x)
        [1, 'Shyam', 25, 'Male', 25000]
```

Data types: Sequence

Add an element to a Python List:

4. + operator

```
In [1]: x= [1,'Shyam',25]

In [2]: print(x)
        [1, 'Shyam', 25]

In [3]: y=['Male', 25000]

In [4]: print(y)
        ['Male', 25000]

In [5]: print(x+y)
        [1, 'Shyam', 25, 'Male', 25000]
```

```
In [3]: x= [1,'Shyam',25]

In [5]: print(x + ['hi']*3)
        [1, 'Shyam', 25, 'hi', 'hi', 'hi']
```

Data types: Sequence

Delete Element from Python List:

```
In [9]: mylist=['a','b','c','d','e','f','g','h']

In [10]: #del one value
         del mylist[2]

In [11]: print(mylist)
['a', 'b', 'd', 'e', 'f', 'g', 'h']
```

```
In [9]: mylist=['a','b','c','d','e','f','g','h']

In [12]: #del multiple value
         del mylist[2:4]

In [13]: print(mylist)
['a', 'b', 'f', 'g', 'h']
```

```
In [9]: mylist=['a','b','c','d','e','f','g','h']

In [14]: #del entire list
         del mylist

In [15]: print(mylist)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-15-aca29853c9f0> in <module>
----> 1 print(mylist)

NameError: name 'mylist' is not defined
```

Dhirubhai University
Formerly known as DA-IICT

Exercise

numbers = [12, 7, 5, 64, 14]

Your tasks are:

- Print the first and last elements of the list.
- Add the number 100 at the end of the list.
- Insert the number 50 at the second position in the list.
- Delete the 3rd element from the list.
- Print the length of the updated list.
- Print all elements using a for loop.

```
numbers = [12, 7, 5, 64, 14]

# 1. Print the first and last elements
print("First Element:", numbers[0])
print("Last Element:", numbers[-1])

# 2. Add the number 100 at the end of the list
numbers.append(100)

# 3. Insert the number 50 at the second position (index 1)
numbers.insert(1, 50)

# 4. Delete the 3rd element from the list
del numbers[2]

# 5. Print the length of the updated list
print("Length of list:", len(numbers))

# 6. Print all elements using a for loop
print("All elements in the list:")
for num in numbers:
    print(num)
```

```
First Element: 12
Last Element: 14
Length of list: 6
All elements in the list:
12
50
5
64
14
100
```

Exercise

numbers = [12, 7, 9, 20, 33, 18, 42, 55, 61, 28]

Your tasks are:

- Write a program that iterates through the list using a for loop.
- If a number is greater than 50, print: "Large number found: <number>".
- After processing, print:
  - The list of even numbers.
  - The list of odd numbers.
  - The total count of even and odd numbers.

Exercise

Write a Python program that builds a list where each element can have a different data type (int, float, or string).

Program Requirements:

- Ask the user how many elements they want to add to the list.
- For each element, prompt the user to:
  - Specify the data type (int, float, or str).
  - Enter the value.
- Convert the value to the specified data type.
- Append the converted value to the list.
- Finally, display the complete list.

```
n = int(input("Enter how many elements you want: "))

elements = []

for i in range(n):
    datatype = input(f"Enter datatype for element {i+1} (int / float / str): ")
    value = input(f"Enter value for element {i+1}: ")

    # Convert value based on datatype
    if datatype == "int":
        value = int(value)
    elif datatype == "float":
        value = float(value)
    elif datatype == "str":
        value = str(value)
    else:
        print("Invalid datatype entered. Treating as string by default.")
        value = str(value)

    elements.append(value)

print("Final List:", elements)
```

```
Enter how many elements you want:  3
Enter datatype for element 1 (int / float / str):  str
Enter value for element 1:  ankit
Enter datatype for element 2 (int / float / str):  int
Enter value for element 2:  3
Enter datatype for element 3 (int / float / str):  float
Enter value for element 3:  3.4
Final List: ['ankit', 3, 3.4]
```

Dhirubhai Ambani University
Formerly known as DA-IICT

Data types: Sequence

Some Python List Methods:

- append(): Add an element to the end of the list.
- extend(): Add all elements of a list to the another list
- insert(): insert an item at the defined index
- remove(): only remove first occurrence of value
- pop(): remove and returns an element at the given index
- clear(): remove all items from the list
- index(): return the index of the first match item
- count(): return the count of number of item passed as an argument
- sort(): sort items in a list in ascending order
- reverse(): reverse the order of items in the list
- len(): length of the list

Data types: Sequence

Some Built-in function with Lists:

- sum(): sum up the number in the list
- ord(): returns an integer representing the Unicode code point of the given Unicode character
- max(): return maximum element of the given list
- min(): return minimum element of the given list
- All(): returns true if all elements of the list is true or list is empty otherwise false
- Any(): return true if any element of the list is true. If list is empty, return false
- len(): Return length of the list or size of the list

Data types: Sequence

Tuple:

- A tuple is similar to a list. The difference between the two is that we cannot change the elements of a tuple once it is assigned whereas in a list, elements can be changed.
- Advantages of Tuple over List
  - Since tuples are immutable, iterating through a tuple is faster than with a list. So there is a slight performance boost.

Data types: Sequence

Creating a Tuple:

```
# Empty Tuple
my_tuple=()
print(my_tuple)
```

```
()
```

```
# Tuple having integer
my_tuple=(1,2,3)
print(my_tuple)
```

```
(1, 2, 3)
```

```
# Tuple with mixed datatypes
my_tuple= (1, "Hello" , 3.5)
print(my_tuple)
```

```
(1, 'Hello', 3.5)
```

```
# Nested Tuple
my_tuple= ("Hello" , [1,2,3,4] , (1,2,3))
print(my_tuple)
```

```
('Hello', [1, 2, 3, 4], (1, 2, 3))
```

Data types: Sequence

Assessing element in a Tuple:

```
: my_tuple=('a','b','c','d','e','f','g','h')

: print(my_tuple[0])

a

: print(my_tuple[5])

f

: print(my_tuple[8])

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-29-82cd0f8a39f9> in <module>
----> 1 print(my_tuple[8])

IndexError: tuple index out of range

: print(my_tuple[1:3])

('b', 'c')
```

Data types: Sequence

Changing in a Tuple:

- Tuples are immutable, meaning you cannot update or change the values of tuple elements. You can take portions of existing tuples to create new tuples.

```
my_tuple1=(1,2,3)
my_tuple1[0]="hello"
print(my_tuple1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-37-cc49132ecbe0> in <module>
      1 my_tuple1=(1,2,3)
----> 2 my_tuple1[0]="hello"
      3 print(my_tuple1)

TypeError: 'tuple' object does not support item assignment
```

```
my_list1=[1,2,3]
my_list1[0]="hello"
print(my_list1)
```

```
['hello', 2, 3]
```

Data types: Sequence

Buil in function with List and Tuple:

Built in function	List	Tuple
append(): Add an element to the end of the list.	✓	X
extend(): Add all elements of a list to the another list	✓	X
insert(): insert an item at the defined index	✓	X
remove(): only remove first occurrence of value	✓	X
pop(): remove and returns an element at the given index	✓	X
clear(): remove all items from the list	✓	X
index(): return the index of the first match item	✓	✓
count(): return the count of number of item passed as an argument	✓	✓
sort(): sort items in a list in ascending order	✓	X
reverse(): reverse the order of items in the list	✓	X

Data types: Sequence

Buil in function with List and Tuple:

Built in function	List	Tuple
sum(): sum up the number in the list	✓	✓
ord(): returns an integer representing the Unicode code point of the given Unicode character	✓	✓
max(): return maximum element of the given list	✓	✓
min(): return minimum element of the given list	✓	✓
All(): returns true if all elements of the list/tuple is true or list is empty otherwise false	✓	✓
Any(): return true if any element of the list/tuple is true. If list is empty, return false	✓	✓
len(): Return length of the list/tuple or size of the list/tuple	✓	✓

Data types: Dictionary

Dictionary:

- A dictionary is an unordered collection of key-value pairs. It is generally used when we have a huge amount of data. Dictionaries are optimized for retrieving data.
  - Keys
  - Values

Creating a Dictionary:

```
# Empty Dictionary
my_dict={}
```

```
# Dictionary with Integer Keys
my_dict={1:"Apple", 2:"Ball"}
print(my_dict)
```

```
{1: 'Apple', 2: 'Ball'}
```

```
# Dictionary with Mixed Keys
my_dict={"Name":"Shyam", "Age":25}
print(my_dict)
```

```
{'Name': 'Shyam', 'Age': 25}
```

```
student_record = {
    101: "Alice",                       # Integer key: Student ID
    "course": "Machine Learning",       # String key: Course name
    (2024, "Spring"): "Enrolled",       # Tuple key: Semester info
    "grades": {"Math": 90, "Python": 95} # String key with a nested dict as value
}
```

Data types: Dictionary

Access Element from a Dictionary:

```
# Dictionary with Mixed Keys
my_dict={"Name":"Shyam", "Age":25}
print(my_dict)
```

```
{'Name': 'Shyam', 'Age': 25}
```

```
print(my_dict["Name"])
```

```
Shyam
```

Change /Add and Delete in a Dictionary:

```
In [16]: my_dict["Age"]=28
         print(my_dict)
```

```
{'Name': 'Shyam', 'Age': 28}
```

```
In [17]: my_dict["Gender"]="Male"
         print(my_dict)
```

```
{'Name': 'Shyam', 'Age': 28, 'Gender': 'Male'}
```

```
In [18]: del my_dict["Age"]
         print(my_dict)
```

```
{'Name': 'Shyam', 'Gender': 'Male'}
```

Data types: Dictionary

Python Nested Dictionary:

```
students = {
    101: {"name": "Alice", "course": "Machine Learning", "GPA": 3.8},
    102: {"name": "Bob", "course": "Data Science", "GPA": 3.5},
    103: {"name": "Charlie", "course": "AI", "GPA": 3.9},
    104: {"name": "Diana", "course": "Cybersecurity", "GPA": 3.6},
    105: {"name": "Ethan", "course": "Cloud Computing", "GPA": 3.7}
}
```

```
students[101]
```

```
{'name': 'Alice', 'course': 'Machine Learning', 'GPA': 3.8}
```

```
students[101]={"name": "Ram", "course": "Python programming", "GPA": 4.9}
```

```
students[101]
```

```
{'name': 'Ram', 'course': 'Python programming', 'GPA': 4.9}
```

Exercise- 9

Employee Database Management:

- **Create a list of dictionaries representing 5 employees in the company. Each employee's dictionary should contain:**
  - ID: Employee ID (integer)
  - name: Name of the employee (string)
  - department: Department the employee belongs to (string)
  - salary: Salary of the employee (float)
- **add a new employee to the list**
- **remove an employee by their ID**
- **update the salary of an employee based on their ID**
- **find an employee by their ID and print their details**

Exercise- 9

```
employees = [
    {"ID": 101, "name": "John", "department": "HR", "salary": 50000},
        {"ID": 102, "name": "Alice", "department": "IT", "salary": 60000},
    {"ID": 103, "name": "Bob", "department": "Finance", "salary": 55000},
    {"ID": 104, "name": "Diana", "department": "Marketing", "salary": 45000},
    {"ID": 105, "name": "Ethan", "department": "IT", "salary": 70000} ]
```

```
new_employee=[{"ID": 106, "name":"Ankit","department": "IT", "salary": 25000}]
employees.extend(new_employee)
```

```
for emp in employees:
    if emp['ID']==101:
        employees.remove(emp)
```

```
for emp in employees:
    if emp["ID"]==103:
        emp["salary"]=28000
```

```
for emp in employees:
    if emp["ID"]==105:
        print(emp)
```

```
{'ID': 105, 'name': 'Ethan', 'department': 'IT', 'salary': 70000}
```

Exercise- 9

Employee Database Management:

- **Create a list of dictionaries representing 5 employees in the company. Each employee's dictionary should contain:**
  - ID: Employee ID (integer)
  - name: Name of the employee (string)
  - department: Department the employee belongs to (string)
  - salary: Salary of the employee (float)

- **add a new employee to the list**
- **remove an employee by their ID**
- **update the salary of an employee based on their ID**
- **find an employee by their ID and print their details**

Create a function

- to add a new employee to the list, where information is provided by the user input.
- Remove an employee by their ID, which is given by the user input in a single line, separated by a space.
- Update the salary of an employee based on their ID, which is given by the user input.
- Find an employee by their ID and print their detail; the ID should be the user input.

Dhirubhai Ambani University
Formerly known as DA-IICT

Exercise- 9

```
employees = [
    {"ID": 101, "name": "John", "department": "HR", "salary": 50000},
        {"ID": 102, "name": "Alice", "department": "IT", "salary": 60000},
    {"ID": 103, "name": "Bob", "department": "Finance", "salary": 55000},
    {"ID": 104, "name": "Diana", "department": "Marketing", "salary": 45000},
    {"ID": 105, "name": "Ethan", "department": "IT", "salary": 70000} ]

def add_employee(employees, new_employee):
    employees.append(new_employee)

def remove_employee(employees, ID):
    for emp in employees:
        if emp["ID"] == ID:
            employees.remove(emp)

def update_salary(employees, ID, new_salary):
    for emp in employees:
        if emp["ID"] == ID:
            emp["salary"] = new_salary
            break

def find_employee_by_id(employees, ID):
    for emp in employees:
        if emp["ID"] == ID:
            print(emp)
```

Exercise- 9

```
user_input_for_add= input("Enter the details of new employee as ID, name, department, salary seperated by space")
new_employee= list(map(str, user_input_for_add.split()))
new_employee={"ID":int(new_employee[0]), "name":new_employee[1], "department":new_employee[2], "salary":int(new_employee[3])}
new_employee

add_employee(employees, new_employee)
```

```
user_input_for_remove= int(input("Enter the ID of employee to remove"))
remove_employee(employees, user_input_for_remove)
```

```
emp_id, New_salary=int(input("Enter the employee ID: ")), int(input("Enter the new salary: "))
update_salary(employees, emp_id, New_salary)
```

```
emp_id=int(input("Enter the employee ID to search"))
find_employee_by_id(employees, emp_id)
```